On Type-Aware Entity Retrieval

Darío Garigliotti
University of Stavanger
dario.garigliotti@uis.no

Krisztian Balog
University of Stavanger
krisztian.balog@uis.no

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 an idealized "oracle" setting, assuming that we know the distribution of target types of the relevant entities for a given 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 find that type information proves most useful when using large type taxonomies that provide very specific types. We provide further insights on the extensional coverage of entities and on the utility of target types.

CCS CONCEPTS

•**Information systems** →**Retrieval Models and Ranking;**

KEYWORDS

Entity retrieval, entity types, semantic search

ACM Reference format:
Darío Garigliotti and Krisztian Balog. 2017. On Type-Aware Entity Retrieval. In *Proceedings of ICTIR'17, October 1–4, 2017, Amsterdam, Netherlands., ,* 8 pages.
DOI: https://doi.org/10.1145/3121050.3121054

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 [30]. Indeed, entities play a key role in transforming search engines into "answer engines" [24]. The pivotal component that sparked this evolution is the increased availability of structured data published in knowledge bases, such as Wikipedia, DBpedia, or the Google Knowledge Graph, now primary sources of information for entity-oriented search. 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* [30], 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 is that entities are typed, this is, grouped into more general classes, i.e., *types*, of entities. 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, in DBpedia, the type of the entity `Albert Einstein` is `Scientist`; according to Wikipedia, 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; see Figure 1 for an illustration. The main high-level research question we are concerned with in this study is: *How to use entity type information in ad-hoc entity retrieval?*

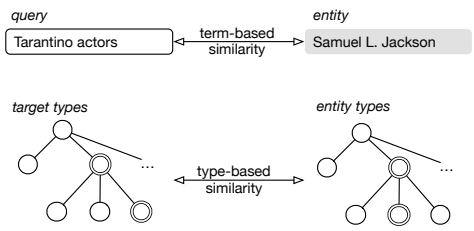


Figure 1: Entity retrieval using entity type information.

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 [9]. 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 [1, 8, 18, 29]. 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 matter of representing type information, i.e., to what extent the hierarchy of the taxonomy should be preserved. Yet another piece of the puzzle is how to combine type-based and text-based

ICTIR'17, October 1–4, 2017, Amsterdam, Netherlands.
 ISBN 978-1-4503-4490-6/17/10…$15.00
DOI: https://doi.org/10.1145/3121050.3121054

similarity 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 to represent hierarchical entity type information for entity retrieval?
RQ3 How to combine term-based and type-based information?

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; see Section 3.
ii) The representation of hierarchical entity type information; see Section 4.
iii) The choice of the type taxonomy; see Section 5.

In summary, our work is the first comprehensive study on the usage of entity type information for entity retrieval. Our main contributions are twofold. First, we present methods for representing types in a hierarchy, establishing type-based similarity, and combining term-based and type-based similarities. Second, we perform a thorough experimental comparison and analysis of all possible configurations across the above identified three dimensions. Our overall finding is that type information has the most benefits in case of large, deep type taxonomies that provide very specific types.

2 RELATED WORK

The task of entity ranking has been studied in different flavors, including ad-hoc entity retrieval [26, 30], list search [5, 10], related entity finding [2], and question answering [23]. Our interest in this work lies in the usage of type information for general-purpose entity retrieval against a knowledge base (KB), where queries may belong to either of the above categories.

Retrieval models. Early works represented type information as a separate field in a fielded entity model [40]. 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 using (i) a linear interpolation [1, 18, 29] or (ii) in a multiplicative manner, where the type-based component essentially serves as a filter [6]. Raviv et al. [32] 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 Section 3, and then compare them in Section 7.

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 [1, 6, 8, 18, 32]. 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 [3, 35], YAGO types [8, 25, 33, 35], Freebase [21], and schema.org [35]. We are not aware of any work that compared different type taxonomies for entity retrieval.

Representations of type information. Target types are commonly considered either as a set [8, 18, 29, 32] or as a bag (weighted set) [1, 33, 36]. Various ways of measuring type-based similarity have been proposed [7, 17, 37, 38, 40]. In this work we employ a state-of-the-art probabilistic approach by Balog et al. [1] (cf. Section 3.3). Within a taxonomy, types are arranged in a hierarchy. Several approaches have attempted to expand the set of target types based on the hierarchical structure of the type system [1, 6, 8, 29]. Importantly, the investigation of type hierarchies has been limited to Wikipedia, and, even there, mixed results are reported [7, 16, 37, 40]. It remains an open question whether considering the hierarchical nature of types benefits retrieval performance. We aim to fill that gap.

Target Type Identification. The INEX Entity Ranking track [10] and the TREC Entity track [5] both featured scenarios where target types are provided by the user. In the lack of explicit target type information, one might attempt to infer types from the keyword query. This subtask is introduced by Vallet and Zaragoza [36] 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. [19] similarly use a simple entity-centric model. Manually assigned target types tend to be more general than automatically identified ones [18]. Having a hierarchical structure, therefore, makes it convenient to assign more general types. In [3], a hierarchical version of the *target type identification* task is addressed using the DBpedia Ontology and language modeling techniques. Sawant and Chakrabarti [33] 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. In our case, an oracle process generates the query target type distribution from its set of known relevant entities (cf. Section 6).

Entity Types. A further complicating issue is that the type information associated with entities in the knowledge base is incomplete, imperfect, or missing altogether for some entities. 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 [28], patterns over logical interpretations of the deeply parsed natural language definitions [13], or linguistic hypotheses about category classes [12]. Several works have addressed entity typing over progressively larger taxonomies with finer-grained types [11, 15, 22, 31, 39]. Regarding the task of detecting and typing *emerging entities*, having fine-grained types for new entities is of particular importance for informative knowledge [21, 25].

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 Section 1. Our contributions do not lie in this part; the techniques we present were shown to be effective in prior research.

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 Section 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 Sections 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 [6] (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 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 below in Section 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:

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

Examples of this approach include [1, 18, 29, 32].

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., [1, 4, 6, 18]. Specifically, we employ the Mixture of Language Models method from [4] with two fields, title and content. Following [27], the weights are set to 0.2 and 0.8, respectively. This is a simple, yet solid baseline approach. 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 [1], 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. Section 6). 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 Section 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 $P(t)$ is the background type model obtained by a maximum-likelihood estimate, and μ is the smoothing parameter, which we set to the average number of types assigned to an entity.

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

$$P(q_t|e) = z(\max_{e'} KL(\theta_q \| \ \theta_{e'}) - KL(\theta_q \| \theta_e)) \ , \tag{5}$$

where z is a normalization factor. Note that the smaller the divergence the more similar the distributions are, therefore in Eq. (5) we subtract it from the maximum KL-divergence, in order to obtain a probability distribution. For further details we refer to [1].

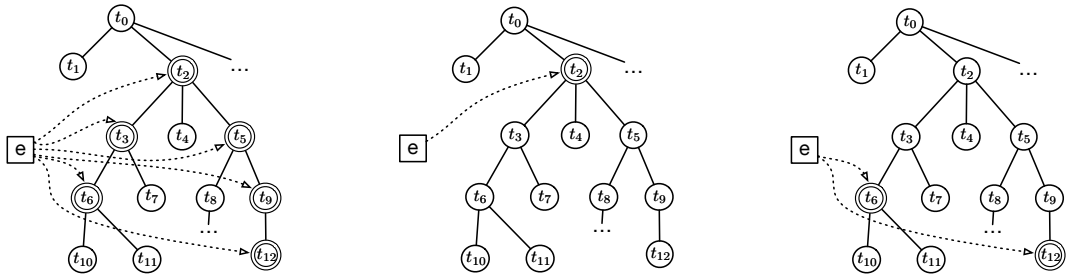
4 REPRESENTING HIERARCHICAL ENTITY TYPE INFORMATION

This section presents various ways of representing hierarchical entity type information. That is, 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.

- T is a type taxonomy that consists of a set of hierarchically organized entity types, and $t \in T$ is a specific entity type.
- E is the set of all entities in the knowledge base, and $e \in E$ is a specific entity.
- $T(e)$ is the set of types that are assigned to entity e in the knowledge base. We refer to this as a set of *assigned types*. Note that $T(e)$ 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: $t \in T(e) \wedge \pi(t) \neq t_0 \implies \pi(t) \in T(e)$.



(a) Path-to-top type(s) (b) Top-level type(s) (c) Most specific type(s)

Figure 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.

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 $T(e)$ 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 or weight with respect to the given entity. This weight 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 Figure 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.

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 $T(e)$ contains all the types to the top-level node):

$$n(t, e) = \mathbb{1}\Big(t \in T(e)\Big) .$$

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

$$n(t, e) = \mathbb{1}\Big(t \in T(e) \wedge \pi(t) = t_0\Big) .$$

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

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

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

5 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 Section 4. Table 1 presents an overview of the type systems (after normalization). The number of type assignments are counted according to the path-to-top representation. Properties of the four type systems and details of the normalization process are discussed below.

Table 1: 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
#types	713	1,719	423,636	568,672
#top-level types	22	92	27	61
#leaf-level types	561	1,626	303,956	549,754
height	7	2	35	19
#types used	408	1,626	359,159	314,632
#entities w/ type	4.87M	3.27M	3.52M	2.88M
avg #types/entity	2.8	4.4	20.8	13.4
mode depth	2	2	11	4

5.1 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 [19]. Also, categories often represent only loose relatedness between articles; category assignments are neither consistent nor complete [9].

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

[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).

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. 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 712 classes organized in a hierarchy of 7 levels.

YAGO taxonomy. YAGO is a huge semantic knowledge base, derived from Wikipedia, WordNet, and GeoNames [34]. Its 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 1,719 types.

5.2 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 path-to-top representation. We treat all entity-type assignments transitively, adhering to constraint (iii) in Section 4.

6 EXPERIMENTAL SETUP

We base our experiments on the DBpedia knowledge base (version 2015-10). DBpedia [20], as a central hub in the Linked Open Data cloud, provides a large repository of entities, which are mapped—directly or indirectly—to each of the type taxonomies of interest.

[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.

Test Collection. Our experimental platform is based on the test collection developed in [4]. The dataset contains 485 queries, synthesized from various entity-related benchmarking evaluation campaigns. These range from short keyword queries to natural language questions.

Target Types Oracle. Throughout all our 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 (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 [9] and the TREC Entity track [5]), where target types are provided explicitly as part of the topic definition. We need 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 the union of all types of all entities that are judged relevant for that query. Each of these types t becomes a target type, and its probability $P(t|\theta_q)$ is set proportional to the number of relevant entities that have that type.

Retrieval Models. As our baseline, we use a term-based approach, specifically the Mixture of Language Models [4], which we described in Section 3.2. We compare three type-aware retrieval models (cf. Section 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.)

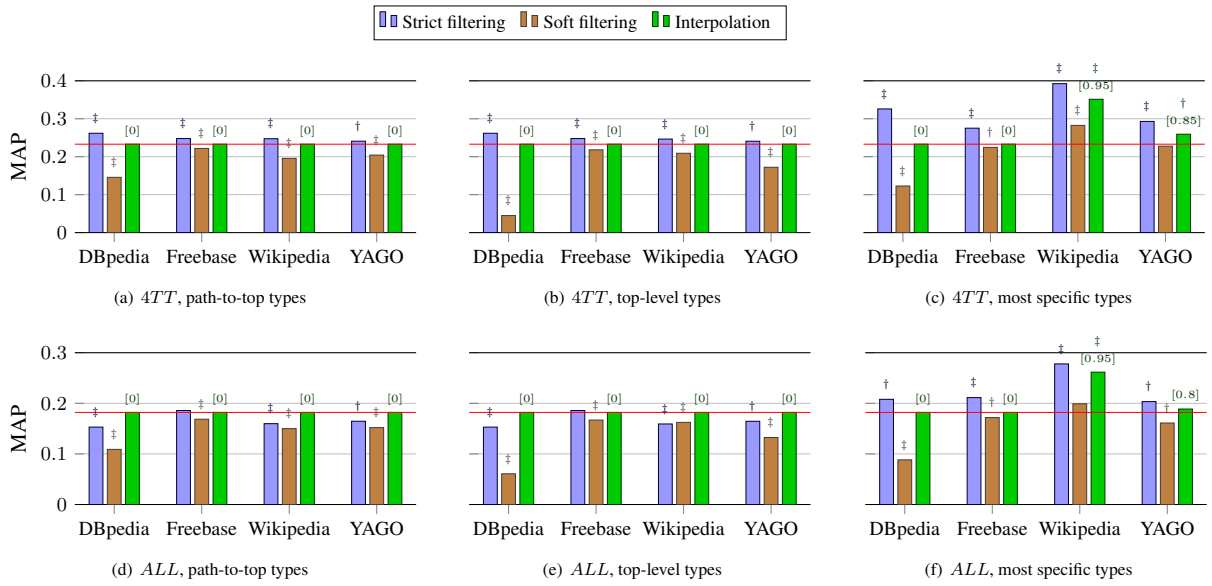
Type Assignments. To ensure that the differences we observe are not a result of missing type assignments, we distinguish between two settings in our experiments.

- **4TT** We restrict our set of entities to those that have types assigned to them from all four type systems (1.51M entities in total). This ensures that the results we obtain are comparable across the different type systems. We also restrict the set of queries to those that have target types in all four type systems; queries without any relevant results (as a consequence of these restrictions) are filtered out. This leaves us with a total of 419 queries.
- **ALL** We include all entities from the knowledge base and use the original set of relevance assessments without any modifications. Hence, some entities and queries do not have types assigned from one or more taxonomies.

7 RESULTS

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.

Figure 3 shows the results, corresponding to the two settings we distinguished in Section 6: in the top histograms, we consider only entities that have types assigned to them in all four type taxonomies (4TT); in the bottom histograms, we rank all entities in the knowledge base (ALL). The red line corresponds to the term-based baseline. Our evaluation metric is Mean Average Precision (MAP).



(a) $4TT$, path-to-top types (b) $4TT$, top-level types (c) $4TT$, most specific types

(d) ALL, path-to-top types (e) ALL, top-level types (f) ALL, most specific types

Figure 3: Entity retrieval performance for all combinations of type taxonomies, type representation modes, and retrieval models. (Top): only entities with types from all four type taxonomies; (Bottom): all entities in the knowledge base. The red line corresponds to the term-based baseline. Above each bar, the symbols † and ‡ indicate statistical significance against the baseline; the numbers in brackets show the type weight (empirically found best λ_t) used by the interpolation model.

We report on statistical significance using a two-tailed paired t-test at $p < 0.05$ and $p < 0.001$, denoted by † and ‡, respectively.

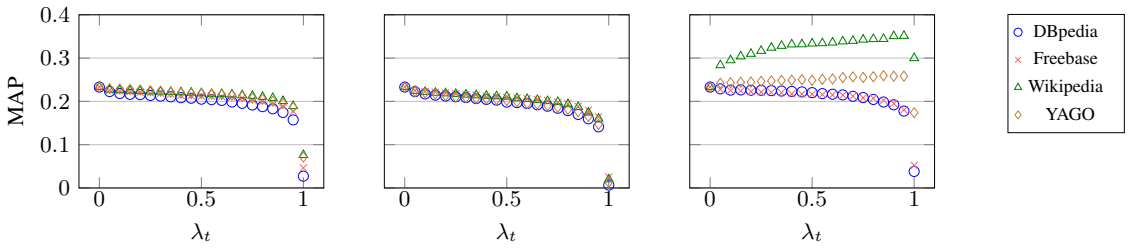
RQ1. Let us turn to our first research question, which concerns the impact of the particular choice of type taxonomy. It is clear that Wikipedia, in combination with the most specific type representation, performs best (for both 4TT and ALL). In particular, for the 4TT setting (top right plot in Figure 3), the improvements for Wikipedia are highly significant for all three retrieval models. As for the rest, there is no easy way to compare taxonomies, as the performance varies depending on the other dimensions. E.g., for 4TT using strict filtering and more general types (i.e., the purple bars in the top left and top middle histograms in Figure 3), the smaller, shallower type taxonomies (DBpedia and Freebase) tend to outperform the larger, deeper ones (Wikipedia and YAGO).

RQ2. The second research question, which is about type representation, has a clear answer: keeping only the most specific types in the hierarchy provides the best performance (right vs. left and middle histograms in Figure 3). This is also in line with findings in past work (cf. Section 2). As for the other two representations, i.e., types along path to top vs. top-level types, two things are worth pointing out. Firstly, the results are the same for both type representations when using strict filtering, which is explained by how the representations are defined in Section 4; if an entity is retained (given that the intersection between the entity's types and the target types is non-empty), this filtering does not change by adding more specific types. Secondly, for the interpolation model, we can observe that the λ_t weights are always 0 for these representations. This means that type information is not used at all. Overall, we could not find any evidence that hierarchical relationships from ancestor types would benefit retrieval effectiveness.

RQ3. Answering our final research question, concerning the type-aware retrieval model, requires a more elaborate treatment. In the 4TT setting, strict filtering is the best retrieval model for every configuration, outperforming the baseline with high significance in almost all cases. This no longer holds in the ALL setting; in fact, all MAP scores drop with respect to the corresponding 4TT configuration. This is expected, as in the more realistic setting, many relevant entities may have incomplete type assignments. Only the interpolation model can deal with this in a robust manner.

Figure 4 shows the performance of the interpolation model when varying the weight of the type-based component (value of λ_t). Due to space constraints, we present the plots only for the 4TT setting; the figures look very similar for the ALL setting. We find that for the smaller, shallower type taxonomies, DBpedia and Freebase, assigning more weight to type-based information is increasingly more harmful, independently of the type representation or type assignment setting. The same occurs for Wikipedia and YAGO using the more general type representations. On the other hand, when using only the most specific types (right plot in Figure 4), for Wikipedia and YAGO, performance increases with higher λ_t values. Yet, MAP scores peak at $\lambda_t < 1$, meaning that term-based similarity is still needed for optimal performance.

The only configurations performing worse than the baseline, even in the 4TT setting, are the ones using the soft filtering model. In particular, the MAP scores for DBpedia with soft filtering are noticeably low. We plan to perform a deeper investigation of this phenomenon in future work.



(a) Path-to-top types (b) Top-level types (c) Most specific types

Figure 4: Retrieval performance (4TT setting) using the interpolation model with different type weights, λ_t.

Table 2: Number of entities with missing type information.

Entities	DBpedia	Freebase	Wikipedia	YAGO
All entities	35,390	1,369,636	1,113,299	1,755,480
Relevant entities	3,341	1,594	2,567	2,532

8 ANALYSIS AND DISCUSSION

Now that we have presented our results, we proceed with further analysis of some of the issues we identified in the previous section.

8.1 Missing Type Information

In order to make a fair comparison between different type taxonomies, we had to account for the fact that the entity type assignments in the knowledge bases may be incomplete (cf. the 4TT setting in Section 6). Indeed, results in Section 7 have shown that the benefits of using type information are more obvious when entities are not missing type assignments. Table 2 shows, for each of the type taxonomies, the number of entities that have no types assigned to them in the KB (i.e., "non-typed" entities). Interestingly, while DBpedia has the least number of non-typed entities (only 35K out of 4.6M), it lacks types for over 25% of the relevant entities (3.3K out of 12.9K). Even for Freebase, which has the best coverage of relevant entities, over 12% of the relevant entities have no type assignments in the KB. Clearly, the problem of missing type information, frequently referred to as partial *extensional* coverage of type systems [13], is an important area of research (cf. Section 2).

8.2 Revisiting the Target Types Oracle

Another aspect of type-based information we are concerned about is the quality of target types. Previously, we have included all types associated with known relevant entities, proportional to their frequency, in the target type distribution (θ_q); we shall refer to it as the *default oracle*. Here, we consider another variant, referred to as *filtered oracle*, where a frequency threshold is applied. Specifically, we include type t as a target type iff at least 3 relevant entities have t assigned to them. As a consequence of this filtering, many queries have an empty set of types; for this experiment, we discard those from the ground truth set, leaving us with 182 queries in total.

A comparison of the two oracles is presented in Figure 5. For the more general type representations, the filtered oracle turns out to be slightly less effective for most of the configurations. Yet, the differences are barely noticeable. When using the most specific types, we find that MAP scores drop, especially for larger, deeper taxonomies (Wikipedia and YAGO); some configurations no longer outperform the term-based baseline. Hence, it is important to consider all possible target types, even those with a low probability.

Table 3: Coverage of relevant entities by top-K types, in terms of precision, recall, and F1, averaged over all queries.

Top-K types	Type Taxonomy	P	R	F1
$K = 1$	DBpedia	0.0027	0.5863	0.0046
	Freebase	0.0060	**0.7254**	0.0076
	Wikipedia	**0.1147**	0.4798	**0.1287**
	YAGO	0.0418	0.6303	0.0488
$K = 3$	DBpedia	0.0006	0.7199	0.0012
	Freebase	0.0004	**0.7805**	0.0008
	Wikipedia	**0.0402**	0.5847	**0.0614**
	YAGO	0.0036	0.7025	0.0062

8.3 What is in a Target Type?

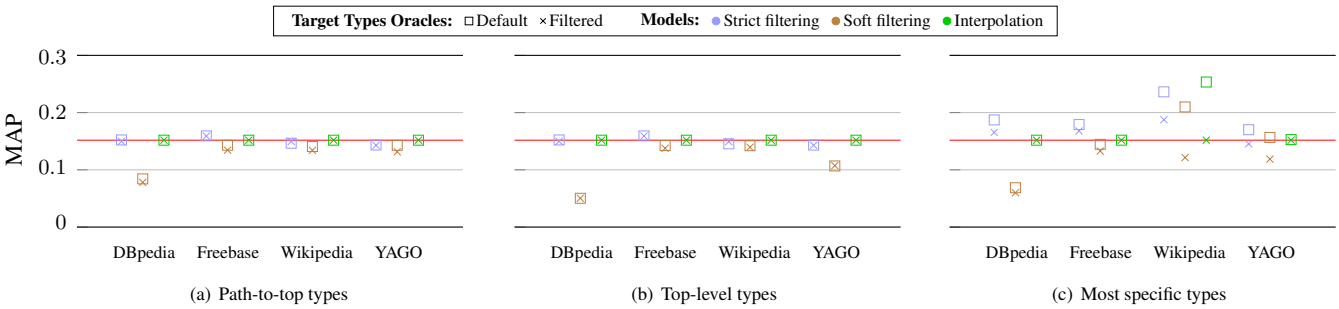
Our ultimate interest in this work is in understanding the usefulness of type information for ad-hoc entity retrieval. What portion of relevant entities can target types help to capture? To shed some light on this, we measure the coverage of relevant entities by (i) the top ranked type and (ii) the set of top 3 types.[3] Table 3 reports the results. We find that Wikipedia has, by far, the highest precision and F1-score among all type taxonomies; YAGO comes second. Notice that these are the two taxonomies that performed best, when using the most specific type representations, in Figure 3.

In summary, we have found that specific types from large, fine-grained taxonomies provide the best performance. Yet, it appears that it is not the hierarchical nature of the taxonomy that brings benefits, but rather the fact that these fine-grained types provide semantic sets or classes that can capture (some subset of) the relevant entities with high precision.

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,

[3] For this experiment, we take the type assignments "as-recorded" in Wikipedia, without enforcing the taxonomical constraints.



(a) Path-to-top types (b) Top-level types (c) Most specific types

Figure 5: Retrieval performance using the default vs. filtered target types oracle. The red line is the term-based baseline.

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. We have found that using the most specific types in a fine-grained taxonomy, like Wikipedia, leads to the best retrieval effectiveness.

We identify two directions for future work. First, we plan to report on an even deeper query-level analysis, which was not possible here due to space limitations. Second, our investigations so far have taken place in an idealized environment, assuming that an "oracle" process can provide us with the target types for each query. We wish to perform a similar analysis using automatically identified target types [14].

REFERENCES

[1] Krisztian Balog, Marc Bron, and Maarten De Rijke. 2011. Query modeling for entity search based on terms, categories, and examples. *ACM Trans. Inf. Syst.* 29, 4 (2011), 22:1–22:31.

[2] K. Balog, A. P. de Vries, P. Serdyukov, P. Thomas, and T. Westerveld. 2010. Overview of the TREC 2009 Entity Track. In *Proc. of TREC*.

[3] Krisztian Balog and Robert Neumayer. 2012. Hierarchical target type identification for entity-oriented queries. In *Proc. of CIKM*. 2391–2394.

[4] Krisztian Balog and Robert Neumayer. 2013. A Test Collection for Entity Search in DBpedia. In *Proc. of SIGIR*. 737–740.

[5] Krisztian Balog, Pavel Serdyukov, and Arjen P. De Vries. 2012. Overview of the TREC 2011 Entity Track. In *Proc. of TREC*.

[6] Marc Bron, Krisztian Balog, and Maarten de Rijke. 2010. Ranking Related Entities: Components and Analyses. In *Proc. of CIKM*. 1079–1088.

[7] Gianluca Demartini, Claudiu S. Firan, and Tereza Iofciu. 2008. Focused Access to XML Documents. Springer, Chapter L3S at INEX 2007, 252–263.

[8] Gianluca Demartini, Claudiu S. Firan, Tereza Iofciu, Ralf Krestel, and Wolfgang Nejdl. 2010. Why finding entities in Wikipedia is difficult, sometimes. *Information Retrieval* 13, 5 (may 2010), 534–567.

[9] Gianluca Demartini, Tereza Iofciu, and Arjen P. De Vries. 2010. Overview of the INEX 2009 Entity Ranking Track. In *Focused Retrieval and Evaluation, and INEX*. 254–264.

[10] Gianluca Demartini, Tereza Iofciu, and Arjen P. De Vries. 2010. Overview of the INEX 2009 Entity Ranking Track. In *Focused Retrieval and Evaluation*. 254–264.

[11] Michael Fleischman and Eduard Hovy. 2002. Fine Grained Classification of Named Entities. In *Proc. of COLING*. 1–7.

[12] Marco Fossati, Dimitris Kontokostas, and Jens Lehmann. 2015. Unsupervised Learning of an Extensive and Usable Taxonomy for DBpedia. In *Proc. of SEMANTICS*. 177–184.

[13] Aldo Gangemi, Andrea Giovanni Nuzzolese, Valentina Presutti, Francesco Draicchio, Alberto Musetti, and Paolo Ciancarini. 2012. Automatic Typing of DBpedia Entities. In *Proc. of ISWC*. 65–81.

[14] Darío Garigliotti, Faegheh Hasibi, and Krisztian Balog. 2017. Target Type Identification for Entity-Bearing Queries. In *Proc. of SIGIR*. 845–848.

[15] Claudio Giuliano. 2009. Fine-grained Classification of Named Entities Exploiting Latent Semantic Kernels. In *Proc. of CoNLL*. 201–209.

[16] Janne Jämsen, Turkka Näppilä, and Paavo Arvola. 2008. Focused Access to XML Documents. Springer, Chapter Entity Ranking Based on Category Expansion, 264–278.

[17] Rianne Kaptein and Jaap Kamps. 2009. Finding Entities in Wikipedia using Links and Categories. In *Advances in Focused Retrieval, INEX*. 273–279.

[18] Rianne Kaptein and Jaap Kamps. 2013. Exploiting the category structure of Wikipedia for entity ranking. *Artificial Intelligence* 194 (jan 2013), 111–129.

[19] Rianne Kaptein, Pavel Serdyukov, Arjen P. De Vries, and Jaap Kamps. 2010. Entity ranking using Wikipedia as a pivot. In *Proc. of CIKM*. 69–78.

[20] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2015. DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web* 6, 2 (2015), 167–195.

[21] Thomas Lin, Mausam, and Oren Etzioni. 2012. No Noun Phrase Left Behind: Detecting and Typing Unlinkable Entities. In *Proc. of EMNLP-CoNLL*. 893–903.

[22] Xiao Ling and Daniel S. Weld. 2012. Fine-grained Entity Recognition. In *Proc. of AAAI*. 94–100.

[23] Vanessa Lopez, Christina Unger, Philipp Cimiano, and Enrico Motta. 2013. Evaluating Question Answering over Linked Data. *Web Semantics: Science, Services and Agents on the World Wide Web* 21 (aug 2013), 3–13.

[24] Peter Mika. 2013. Entity Search on the Web. In *Proc. of WWW*. 1231–1232.

[25] Ndapandula Nakashole, Tomasz Tylenda, and Gerhard Weikum. 2013. Fine-grained Semantic Typing of Emerging Entities. In *Proc. of ACL*. 1488–1497.

[26] Robert Neumayer, Krisztian Balog, and Kjetil Nørvåg. 2012. On the modeling of entities for ad-hoc entity search in the web of data. In *Proc. of ECIR*. 133–145.

[27] Robert Neumayer, Krisztian Balog, and Kjetil Nørvåg. 2012. When simple is (more than) good enough: effective semantic search with (almost) no semantics. In *Proc. of ECIR*. 540–543.

[28] Andrea Giovanni Nuzzolese, Aldo Gangemi, Valentina Presutti, and Paolo Ciancarini. 2012. Type inference through the analysis of Wikipedia links. In *Proc. of LDOW*.

[29] Jovan Pehcevski, James A Thom, Anne-Marie Vercoustre, and Vladimir Naumovski. 2010. Entity ranking in Wikipedia: utilising categories, links and topic difficulty prediction. *Information Retrieval* 13, 5 (2010), 568–600.

[30] Jeffrey Pound, Peter Mika, and Hugo Zaragoza. 2010. Ad-hoc object retrieval in the web of data. In *Proc. of WWW*. 771–780.

[31] Altaf Rahman and Vincent Ng. 2010. Inducing Fine-grained Semantic Classes via Hierarchical and Collective Classification. In *Proc. of COLING*. 931–939.

[32] Hadas Raviv, David Carmel, and Oren Kurland. 2012. A Ranking Framework for Entity Oriented Search Using Markov Random Fields. In *Proc. of JIWES*. 1:1–1:6.

[33] Uma Sawant and S Chakrabarti. 2013. Learning Joint Query Interpretation and Response Ranking. In *Proc. of WWW*. 1099–1109.

[34] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A Core of Semantic Knowledge. In *Proc. of WWW*. 697–706.

[35] Alberto Tonon, Michele Catasta, Gianluca Demartini, Philippe Cudré-Mauroux, and Karl Aberer. 2013. TRank: Ranking Entity Types Using the Web of Data. In *Proc. of ISWC*. 640–656.

[36] David Vallet and Hugo Zaragoza. 2008. Inferring the most important types of a query: a semantic approach. In *Proc. of SIGIR*. 857–858.

[37] Anne-Marie Vercoustre, Jovan Pehcevski, and James A. Thom. 2008. Focused Access to XML Documents. Springer, Chapter Using Wikipedia Categories and Links in Entity Ranking, 321–335.

[38] W. Weerkamp, K. Balog, and E. J. Meij. 2009. A Generative Language Modeling Approach for Ranking Entities. In *Advances in Focused Retrieval, INEX*. 292–299.

[39] Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart Marc Spaniol, and Gerhard Weikum. 2012. HYENA: Hierarchical Type Classification for Entity Names. In *Proc. of COLING*. 1361–1370.

[40] Jianhan Zhu, Dawei Song, and Stefan Rüger. 2008. Focused Access to XML Documents. Springer, Chapter Integrating Document Features for Entity Ranking, 336–347.