# Target Type Identification for Entity-Bearing Queries

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

Faegheh Hasibi
Norwegian University of Science and Technology
faegheh.hasibi@ntnu.no

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

## ABSTRACT

Identifying the target types of entity-bearing queries can help improve retrieval performance as well as the overall search experience. In this work, we address the problem of automatically detecting the target types of a query with respect to a type taxonomy. We propose a supervised learning approach with a rich variety of features. Using a purpose-built test collection, we show that our approach outperforms existing methods by a remarkable margin.

## KEYWORDS

Query understanding; query types; entity search; semantic search

**ACM Reference format:**
Darío Garigliotti, Faegheh Hasibi, and Krisztian Balog. 2017. Target Type Identification for Entity-Bearing Queries. In *Proceedings of SIGIR'17, August 7–11, 2017, Shinjuku, Tokyo, Japan,* 4 pages.
DOI: http://dx.doi.org/10.1145/3077136.3080659

## 1 INTRODUCTION

A significant portion of information needs in web search target entities [18]. Entities, such as people, organizations, or locations are natural units for organizing information and for providing direct answers. A characteristic property of entities is that they are typed, where types are typically organized in a hierarchical structure, i.e., a *type taxonomy*. Previous work has shown that entity retrieval performance can be significantly improved when a query is complemented with explicit *target type* information, see, e.g., [1, 14, 17]. Most of this work has been conducted in the context of TREC and INEX benchmarking campaigns, where target types are readily provided (by topic creators). Arguably, this is an idealized and unrealistic scenario. Users are accustomed to the "single search box" paradigm, and asking them to annotate queries with types might lead to a cognitive overload in many situations. A more realistic scenario is that the user first issues a keyword query, and then (optionally) uses a small set of (automatically) recommended types as facets, for filtering the results. Target types may also be used, among others, for direct result displays, as it is seen increasingly often in modern web search engines.

Motivated by the above reasons, our main objective is to generate target type annotations of queries automatically. Following the *hierarchical target type identification* task proposed in [2], we wish to identify the most specific target types for a query, from a given type taxonomy, such that they are sufficient to cover all relevant results. One important assumption made in [2] is that each query must have a *single* target type; queries without a clearly identifiable type were discarded. This limits the potential for usefulness in practice. Therefore, we introduce a relaxation to the task definition, by allowing for a query to have multiple target types (or none).

One main contribution of this work is a test collection we build for the revised hierarchical target type identification task. We use the DBpedia ontology as our type taxonomy and collect relevance labels via crowdsourcing for close to 500 queries. As our second main contribution, we develop a supervised learning approach with a rich set of features, including term-based, linguistic, and distributional similarity, as well as taxonomic features. Out of these, we find the distributional similarity features to be the most effective. Our supervised learning approach outperforms existing baselines by a large margin, and does consistently so across all query categories. All resources developed within this study (i.e., the test collection, pre-computed features, and final rankings) are made publicly available at http://bit.ly/sigir2017-querytypes. An extended version of this paper is available at https://arxiv.org/abs/1705.06056.

## 2 RELATED WORK

Most of the research related to the usage of type information in *ad hoc entity ranking* has been conducted in the context of the INEX Entity Ranking [9] and TREC Entity [4] tracks. There, it is assumed that the user complements the keyword query with one or more *target types*. Several works have reported consistent and significant performance improvements when a type-based component is incorporated into the (term-based) retrieval model, see, e.g., [1, 8, 14, 17, 19]. In the lack of explicit target type information, one might attempt to infer types from the keyword query. Vallet and Zaragoza [20] introduce the *entity type ranking* problem, where they consider the types associated with the top-ranked entities using various weighting functions. Balog and Neumayer [2] address a hierarchical version of the *target type identification* task using the DBpedia ontology and language modeling techniques. Sawant and Chakrabarti [19] focus on telegraphic queries and assume that each query term is either a type hint or a "word matcher," i.e., strongly assuming that every query contains a type hint. 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. Our work also falls within the broad area of *query understanding*, which, according to [7], refers to process of "identifying the underlying intent of the queries, based on a particular representation." This includes, among many others, recognizing entity mentions in queries [10] and linking them to knowledge base entries [11, 13].

*SIGIR'17, August 7–11, 2017, Shinjuku, Tokyo, Japan*
© 2017 ACM. 978-1-4503-5022-8/17/08...$15.00
DOI: http://dx.doi.org/10.1145/3077136.3080659

# 3 TARGET TYPE DETECTION

We begin by providing a detailed explanation of the task we are addressing, and then present various approaches for solving it.

## 3.1 Problem Definition

Our objective is to assign target 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 [2]: "find 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 paths in the taxonomy. Second, it can happen—and in fact it does happen for 33% of the queries considered in [2]—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 [2]. 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.

*Definition 3.1 (HTTIv2).* Find 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 path 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.

## 3.2 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., [2, 15, 20]. This model also fits the late fusion design pattern for object retrieval [21]. 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:

$$score_{EC}(t,q) = \sum_{e \in R_K(q)} score(q,e) \times w(e,t),$$

where $R_K(q)$ is the set of top-$K$ ranked entities for query $q$. The retrieval score of entity $e$ is denoted by $score(q,e)$. We consider both Language Modeling (LM) and BM25 as the underlying entity retrieval model. For LM, we use Dirichlet prior smoothing with the smoothing parameter set to 2000. For BM25, we use $k1 = 1.2$ and $b = 0.75$. The rank-cutoff threshold $K$ is set empirically. The entity-type association weight, $w(e,t)$, is set uniformly across entities that are typed with $t$, i.e., $w(e,t) = 1/\sum_{e'} \mathbb{1}(e',t)$, and is 0 otherwise. $\mathbb{1}(e,t)$ is an indicator function that returns 1 if $e$ is typed with $t$, otherwise returns 0.

## 3.3 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 of that type. Then, those type representations can be ranked much like documents. This model has been presented in [2] using Language Models, and has been generalized to arbitrary retrieval models (and referred to as the early fusion design pattern for object retrieval) in [21]. The (pseudo) frequency of a word for a type is defined as: $\tilde{f}(w,t) = \sum_e f(w,e) \times w(e,t)$, where $f(w,e)$ is the frequency of the term $w$ in (the description of) entity $e$ and $w(e,t)$, as before, denotes the entity-type association weight. The relevance score of a type for a given query $q$ is then calculated as the sum of the individual query term scores:

$$score_{TC}(t,q) = \sum_{i=1}^{|q|} score(q_i, \tilde{f}, \varphi)$$

where $score(q_i, \tilde{f}, \varphi)$ is the underlying term-based retrieval model (e.g., LM or BM25), parameterized by $\varphi$. We use the same parameter settings as in §3.2. This model assigns a score to each query term $q_i$, based on the pseudo word frequencies $\tilde{f}$.

## 3.4 Our Approach

To the best of our knowledge, we are the first ones to address the target type detection task using a learning-to-rank (LTR) approach. The entity-centric and type-centric models capture different aspects of the task, and it is therefore sensible to combine the two (as already suggested in [2]). In addition, there are other signals that one could leverage, including taxonomy-driven features and type label similarities. Table 1 summarizes our features. Due to space limitations, we only highlight our distributional similarity features, which were found to perform best (cf. Fig. 2); a detailed description of all features may be found in the extended version of the paper.

We use pre-trained word embeddings provided by the *word2vec* toolkit [16]. 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:

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

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:

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

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

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.

# 4 BUILDING A TEST COLLECTION

We base our test collection on the DBpedia-Entity collection [3]. This dataset contains 485 queries, synthesized from various entity-related benchmarking evaluation campaigns, ranging from short keyword queries to natural language questions. The DBpedia-Entity

**Table 1: Features for learning to rank target types.**

| # | Feature | Description | Kind | Value |
|---|---|---|---|---|
| *Baseline features* | | | | |
| 1-5 | $EC_{BM25,K}(t,q)$ | Entity-centric type score (cf. §3.2) with $K \in \{5, 10, 20, 50, 100\}$ using BM25 | entity-centric | $[0..\infty)$ |
| 6-10 | $EC_{LM,K}(t,q)$ | Entity-centric type score (cf. §3.2) with $K \in \{5, 10, 20, 50, 100\}$ using LM | entity-centric | $[0..1]$ |
| 11 | $TC_{BM25}(t,q)$ | Type-centric score (cf. §3.3) using BM25 | type-centric | $[0..\infty)$ |
| 12 | $TC_{LM}(t,q)$ | Type-centric score (cf. §3.3) 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 mapped 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 | $[0..1]$ |

collection has been used in several recent works, among others, in [6, 12, 22]. We use the DBpedia Ontology (version 2015-10) as our type taxonomy, which is a manually curated and proper "is-a" hierarchy (unlike, e.g., Wikipedia categories). We note that none of the elements of our approach are specific to this taxonomy, and our methods can be applied on top of any type taxonomy.

*Generating the pool.* A *pool* of target entity types is constructed from four baseline methods, taking the top 10 types from each: entity-centric (cf. §3.2) and type-centric (cf. §3.3), using $K$=100, and both BM25 and LM as retrieval methods. Additionally, we included all types returned by an *oracle* method, which has knowledge of the set of relevant entities for each query (from the ground truth). Specifically, the oracle score is computed as: $score_O(t,q) = \sum_{e \in Rel(q)} \mathbb{1}(e,t)$, where $Rel(q)$ indicates the set of relevant entities for the query. We employ this oracle to ensure that all reasonable types are considered when collecting human annotations.

*Collecting judgments.* 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 [2]). 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.

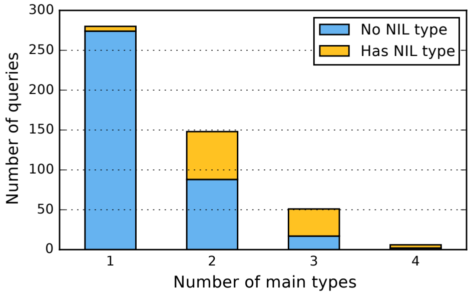

**Figure 1: Distribution of the number of main target types.**

*Results.* 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 generic type received all the "votes" of the more specific one). This affected 120 queries. Figure 1 shows the distribution of queries according to the number of main types. 280 of all queries (57.73%) have a single target type, while the remainder of them have multiple target types. Notice that as the number of main types increases, so does the proportion of NIL-type annotations.

## 5 EVALUATING TARGET TYPE DETECTION

Next, we present our evaluation results and analysis.

### 5.1 Evaluation Methodology

Following [2], we approach the task as a ranking problem and report on NDCG at rank positions 1 and 5. The relevance level ("gain") of a type is set to the number of assessors that selected that type. Detecting NIL-type queries is a separate problem on its own, which we are not addressing in this paper. Therefore, the NIL-type labels are ignored in our experimental evaluation (affecting 104 queries). Queries that got only the NIL-type assigned to them are removed (6 queries in total). No re-normalization of the relevance

Table 2: Target type detection performance.

| Method | NDCG@1 | NDCG@5 |
|---|---|---|
| EC, BM25 ($K = 20$) | 0.1490 | 0.3223 |
| EC, LM ($K = 20$) | 0.1417 | 0.3161 |
| TC, BM25 | 0.2015 | 0.3109 |
| TC, LM | 0.2341 | 0.3780 |
| LTR | **0.4842** | **0.6355** |

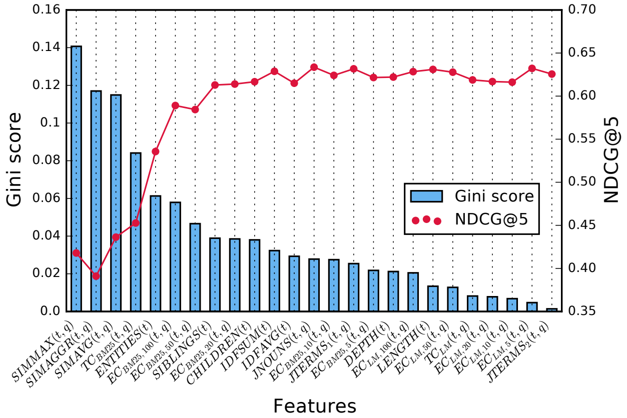

**Figure 2: Performance of our LTR approach when incrementally adding features according to their information gain.**

levels for NIL-typed queries is performed (similar to the setting in [5]). For the LTR results, we used 5-fold cross-validation.

## 5.2 Results and Analysis

Table 2 presents the evaluation results. We find that our supervised learning approach significantly and substantially outperforms all baseline methods (relative improvement over 43% according to any metric, with $p < 0.001$ using a two-tailed paired T-test).

*Feature analysis.* We analyze the discriminative power of our features, by sorting them according to their information gain, measured in terms of Gini importance (shown as the vertical bars in Fig. 2). The top 3 features are: $SIMMAX(t,q)$, $SIMAGGR(t,q)$, and $SIMAVG(t,q)$. 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 (shown as the line plot in Fig. 2). In each iteration, we set the $m$ parameter of the Random Forests algorithm to 10% of the size of the feature set.

*Query category analysis.* In Figure 3, we break performance down into different query categories, following the grouping scheme of Zhiltsov et al. [22]. A first observation is about robustness: our proposed method clearly outperforms the baselines in every query category, i.e., it succeeds in automatically detecting target types for a wide variety of queries. We find the biggest improvements for QALD-2; these queries are mostly well-formed natural language questions. On the other hand, SemSearch ES, which contains short (and ambiguous) keyword queries, has the lowest performance.

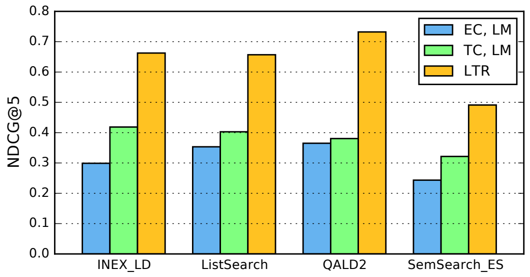

**Figure 3: Performance across different query categories.**

## 6 CONCLUSIONS

In this paper, we have addressed the problem of automatically detecting target types of a query with respect to a type taxonomy. We have proposed a supervised learning approach with a rich set of features. We have developed test collection and showed that our approach outperforms previous methods by a remarkable margin.

## 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] Krisztian Balog and Robert Neumayer. 2012. Hierarchical Target Type Identification for Entity-oriented Queries. In *Proc. of CIKM*. 2391–2394.

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

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

[5] Hannah Bast, Björn Buchhold, and Elmar Haussmann. 2015. Relevance Scores for Triples from Type-Like Relations. In *Proc. of SIGIR*. 243–252.

[6] Jing Chen, Chenyan Xiong, and Jamie Callan. 2016. An Empirical Study of Learning to Rank for Entity Search. In *Proc. of SIGIR*. 737–740.

[7] W Bruce Croft, Michael Bendersky, Hang Li, and Gu Xu. 2010. Query Representation and Understanding Workshop. In *SIGIR Forum*. 48–53.

[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 (2010), 534–567.

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

[10] Jiafeng Guo, Gu Xu, Xueqi Cheng, and Hang Li. 2009. Named Entity Recognition in Query. In *Proc. of SIGIR*. 267–274.

[11] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. 2015. Entity Linking in Queries: Tasks and Evaluation. In *Proc. of ICTIR*. 171–180.

[12] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. 2016. Exploiting Entity Linking in Queries for Entity Retrieval. In *Proc. of ICTIR*. 209–218.

[13] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. 2017. Entity Linking in Queries: Efficiency vs. Effectiveness. In *Proc. of ECIR*. 40–53.

[14] Rianne Kaptein and Jaap Kamps. 2013. Exploiting the Category Structure of Wikipedia for Entity Ranking. *Artificial Intelligence* 194 (2013), 111–129.

[15] 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.

[16] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. In *Proc. of NIPS*. 3111–3119.

[17] 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.

[18] Jeffrey Pound, Peter Mika, and Hugo Zaragoza. 2010. Ad-hoc Object Retrieval in the Web of Data. In *Proc. of WWW*. 771–780.

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

[20] David Vallet and Hugo Zaragoza. 2008. Inferring the Most Important Types of a Query: a Semantic Approach. In *Proc. of SIGIR*. 857–858.

[21] Shuo Zhang and Krisztian Balog. Design Patterns for Fusion-Based Object Retrieval. In *Proc. of ECIR*. 684–690.

[22] Nikita Zhiltsov, Alexander Kotov, and Fedor Nikolaev. 2015. Fielded Sequential Dependence Model for Ad-Hoc Entity Retrieval in the Web of Data. In *Proc. of SIGIR*. 253–262.