TRank: Ranking Entity Types Using the Web of Data

Alberto Tonon[1], Michele Catasta[2], Gianluca Demartini[1], Philippe Cudré-Mauroux[1], and Karl Aberer[2]

[1] eXascale Infolab, University of Fribourg—Switzerland
{firstname.lastname}@unifr.ch
[2] EPFL, Lausanne—Switzerland
{firstname.lastname}@epfl.ch

Abstract. Much of Web search and browsing activity is today centered around entities. For this reason, Search Engine Result Pages (SERPs) increasingly contain information about the searched entities such as pictures, short summaries, related entities, and factual information. A key facet that is often displayed on the SERPs and that is instrumental for many applications is the entity *type*. However, an entity is usually not associated to a single generic type in the background knowledge bases but rather to a set of more specific types, which may be relevant or not given the document context. For example, one can find on the Linked Open Data cloud the fact that Tom Hanks is a person, an actor, and a person from Concord, California. All those types are correct but some may be too general to be interesting (e.g., person), while other may be interesting but already known to the user (e.g., actor), or may be irrelevant given the current browsing context (e.g., person from Concord, California). In this paper, we define the new task of ranking entity types given an entity and its context. We propose and evaluate new methods to find the most relevant entity type based on collection statistics and on the graph structure interconnecting entities and types. An extensive experimental evaluation over several document collections at different levels of granularity (e.g., sentences, paragraphs, etc.) and different type hierarchies (including DBPedia, Freebase, and schema.org) shows that hierarchy-based approaches provide more accurate results when picking entity types to be displayed to the end-user while still being highly scalable.

1 Introduction

Many online queries are about entities [14]. Commercial search engines are increasingly returning rich Search Engine Result Pages (SERPs) that contain not just ten blue links but also images, videos, news, etc. When searching for a specific entity, users may be presented in the SERP with a summary of the entity itself. This search task is known as ad-hoc object retrieval [20], that is, finding an entity described by a keyword query in a structured knowledge base. After correctly identifying the entity described by the user query, the subsequent task is that of deciding what entity information to present on the SERP among all

potential pieces of information available in the knowledge base. It is possible, for example, to display pictures, a short textual description, and related entities.

One interesting entity facet that can be displayed in the SERP is its *type*. In public knowledge bases such as Freebase, entities are associated with several types. For example, the entity 'Peter Jackson' in Freebase[3] has 17 types, among which 'Person', 'Ontology Instance', 'Film director', and 'Chivalric Order Member' can be found. When deciding what to show on the SERP, it is important to select the few types the user would find relevant only. Some types are in most cases not compelling (e.g., 'Ontology Instance') while other types (e.g., 'Film director') may be interesting for a user who does not know much about the entity. Users who already know the entity but are looking for some of its specific facets might be interested in less obvious types (e.g., 'Chivalric Order Member', and its associated search results).

More than just for search, entity types can be displayed to Web users while browsing and reading Web pages. In such a case, pop-ups displaying contextual entity summaries (similar to the ones displayed on SERPs like in Google's Knowledge Panel) can be shown to the users who want to know more about a given entity she is reading about. In this case again, picking the types that are relevant is critical and highly context-dependent.

A third example scenario is to use selected entity types to summarize the content of Web pages or online articles. For example, one might build a summary for a given news article by extracting the most important entities in the article and listing their most relevant types (e.g., 'this article is about two actors and the president of Kenya').

In this paper, we focus on the novel task of ranking available entity types based on their relevance given a context. We propose several methods exploiting the entity type hierarchy (i.e., types and their subtypes like 'person' and 'politician'), collection statistics such as the popularity of the types or their co-occurrences, and the graph structure connecting semantically related entities (potentially through the type hierarchy).

We experimentally evaluate our different approaches using crowdsourced judgments on real data and extracting different contexts (e.g., word only, sentence, paragraph) for the entities. Our experimental results show that approaches based on the type hierarchy perform more effectively in selecting the entity types to be displayed to the user. The combination of the proposed ranking functions by means of learning to rank models yields the best effectiveness. We also assess the scalability of our approach by designing and evaluating a Map/Reduce version of our system, *TRank*, over a large sample of the CommonCrawl dataset[4] containing `schema.org` annotations.

In summary, the main contributions of this paper are:

- The definition of the new task of entity type ranking, whose goal is to select the most relevant types for an entity given some context.

[3] `http://www.freebase.com/edit/topic/en/peter_jackson`
[4] `http://commoncrawl.org/`

– Several type-hierarchy and graph-based approaches that exploit both schema and instance relations to select the most relevant entity types based on a query entity and the user browsing context.
– An extensive experimental evaluation of the proposed entity type ranking techniques over a Web collection and over different entity type hierarchies including YAGO [23] and DBpedia [1] by means of crowdsourcing relevance judgements.
– A scalable version of our type ranking approach evaluated over a large annotated Web crawl.

The rest of the paper is structured as follows. We start below by describing related work surveying entity-search and ad-hoc object retrieval techniques. We formally define our new type ranking task in Section 3 and propose a series of approaches to solve it based on collection statistics, type hierarchies, and entity graphs in Section 4. Section 5 presents experimental results comparing the effectiveness of our various entity ranking approaches over different document collections and type hierarchies as well as a scalability validation of our Map/Reduce implementation over a large corpus. Finally, we conclude in Section 6.

2 Related Work

Entity-centric data management is an emerging area of research at the intersection of several fields including Databases, Information Retrieval, and the Semantic Web. In this paper we target the specific problem of assigning types to entities that have been extracted from a Web page and correctly identified in a preexisting knowledge base.

Classic approaches to Named Entity Recognition (NER) typically provide as output some type information about the identified entities; In most cases, such types consist of a very limited set of entities including Person, Location, and Organization (see e.g., [4, 5]). While this is useful for applications that need to focus on one of those generic types, for other applications such as entity-based faceted search it would be much more valuable to provide specific types that are also relevant to the user's browsing context.

In the field of Information Retrieval, entity retrieval has been studied for a few years. In this context, TREC[5] organized an Entity Track where different entity-centric search tasks have been studied: Four entity types were considered in that context, i.e., people, products, organizations, and locations. Type information can also be used for entity search tasks, e.g., by matching the types of the entities in the query to the types of the retrieved entities (see for instance [7]). In the NLP field, entity extraction methods are continuously being developed. Here also, the types that are considered are typically rather limited. For example, in the method proposed in [9] 18 types are considered. In [19, 18], authors propose a NER system to recognize 100 entity types using a supervised approach. The

[5] `http://trec.nist.gov`

starting point to define the 100 entity types is the BBN linguistic collection[6] which includes 12 top types and 64 subtypes.

The Semantic Web community has been creating large-scale knowledge bases defining a multitude of entity types. Efforts such as YAGO [23] have assigned to LOD entities many types by combining Wikipedia categories and WordNet senses. More recently, projects such as DBpedia [1] and Freebase [2] have collected large collections of structured representations of entities along with their related types. Such knowledge bases hence represent extremely valuable resources when working on entity type ranking as we do in this paper.

In a recent demo [25], the task of selecting the most relevant types to be used to summarize an entity has been proposed. However, the focus of this work was on generating an entity description of a given size, while our focus is to select the most relevant types given the context in which the entity is described. Similarly to that work, we build our approaches using large knowledge bases such as YAGO and DBpedia. Another related approach is *Tipalo* [10], where the authors propose an algorithm to extract entity types based on the natural language description of the entity taken from Wikipedia.

Several applications of our techniques could be based on existing work. For instance, entity-type ranking could be applied on open-domain Question Answering [13], where candidate answers are first generated and later on filtered based on the expected answer types. For systems like Watson [26], identifying specific and relevant entity types could potentially significantly improve effectiveness. Another application depending on high-quality entity types is entity resolution over datasets of different entity types. In [27], the authors evaluate their approach on top of four entity types (that is, persons, addresses, schools, and jobs). The availability of more specific entity types would probably be beneficial for this type of task as well.

3 Task Definition

Given a knowledge base containing semi-structured descriptions of entities and their types, we define the task of *entity type ranking* for a given entity e appearing in a document d as the task of ranking all the types $T_e = \{t_1, \ldots, t_n\}$ associated to e based on their relevance to its textual context c_e from d. In RDFS/OWL, the set T_e is typically given by the objects that are related to the URI of e via the `<rdfs:type>` predicate. Moreover, we take into consideration entities connected to e via a `<owl:sameAs>` to URIs of other selected ontologies and we add to T_e all the types directly attached to them. For example, `<dbpedia:Tom_Cruise>` has an `<owl:sameAs>` connection to `<freebase:Tom_Cruise>`, which allows us to add the new type `<freebase:fashion_models>`.

The context c_e of an entity e is defined as the textual content surrounding the entity taken from the document d in which e is mentioned. This context can have a direct influence on the rankings. For example, the entity 'Barack Obama' can be mentioned in a Gulf War context or in a golf tournament context. The most relevant type for 'Barack Obama' is probably different given one or the

[6] `http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2005T33`

other context. The different context types we consider in this paper are: i) three paragraphs around the entity reference (one paragraph preceding, one following, and the paragraph containing the entity); ii) one paragraph only, containing the entity mention; iii) the sentence containing the entity reference; and iv) the entity mention itself with no further textual context.

To rank the types by their relevance given a context, we exploit hierarchies of entity types. In RDFS/OWL, a type hierarchy is typically defined based on the predicate `<rdfs:subClassOf>`. For example, in DBpedia we observe that `<dbpedia-owl:Politician>` is a subclass of `<dbpedia-owl:Person>`. Knowing the relations among types and their depth in the hierarchy is often helpful when automatically ranking entity types. For example, given a type hierarchy related to a specific entity, we might prefer a more specific type rather than a too general one.

We evaluate the quality of a given ranking $(t_i, \ldots, t_j)$ by using ground truth relevance judgements assessing which types are most relevant to an entity e given a context c_e. We discuss rank-based evaluation metrics in Section 5.

4 Approaches to Entity Type Ranking

4.1 *TRank* System Architecture

Our solution, *TRank*, automatically selects the most appropriate entity types for an entity given its context and type information. *TRank* implements several components to extract entities and automatically determine relevant types. First, given a Web page (e.g., a news article), we identify entities mentioned in the textual content of the document using state-of-the-art NER focusing on persons, locations, and organizations.[7] Next, we use an inverted index constructed over DBpedia literals attached to its URIs and use the extracted entity as a query to the index to select the best-matching URI for that entity.[8] Then, given an entity URI, we retrieve (for example, thanks to a SPARQL query to a knowledge base) all the types attached to the entity. In this way, we obtain types such as `<owl:Thing>`, `<yago:JapanPrizeLaureates>` and `<yago:ComputerPioneers>` for the entity `<dbpedia:Tim_Berners-Lee>`. Finally, our system produces a ranking of the resulting types based on the textual context where the entity has been mentioned. A summary of the different steps involved is depicted by Figure 1.

Integrating Different Type Hierarchies. For the purpose of our task, we require a large, integrated collection of entity types to enable fine-grained typing of entities. There are several large ontologies available, both manually constructed [16] as well as based on the widespread success of Wikipedia combined with information extraction algorithms [1, 23]. However, the lack

[7] The current implementation of our system adopts a Conditional Random Field approach to identify entities [8].

[8] This is the same baseline approach used in [6] and in [24] for Entity Linking.

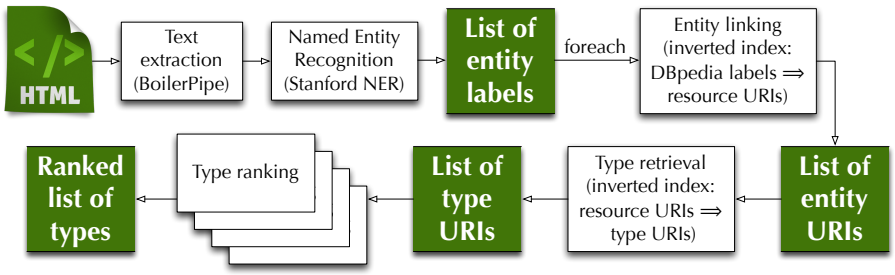

Fig. 1. The *TRank* Architecture.

of alignments among such ontologies hinders the ability of comparing types belonging to different collections.
In *TRank*, we exploit pre-existing mappings provided by DBpedia and PARIS [22] to build a coherent tree of 447,260 types, rooted on `<owl:Thing>` and with a depth of 19. The tree is formed by all the `<rdfs:subClassOf>` relationships among DBpedia, YAGO and schema.org types. To eliminate cycles and to enhance coverage, we exploit `<owl:equivalentClass>` to create `<rdfs:subClassOf>` edges pointing to the parent class (in case one of the two Classes does not have a direct parent). Considering that the probabilistic approach employed by PARIS does not provide a complete mapping between DBpedia and YAGO types, we have manually added 4 `<rdfs:subClassOf>` relationships (reviewed by domain experts) to obtain a single type tree rather than a forest of 5 trees.[9] Figure 2 shows a visual representation of the integrated type hierarchy used by *TRank*.

Entity Type Retrieval and Ranking. Finally, given the entity URI we retrieve all its types (from a background RDF corpus or from a previously created inverted index) and rank them given a context. In this paper, we use the Sindice-2011 RDF dataset[10] [3] to retrieve the types, which consists of about 11 billion RDF triples.

The proposed approaches for entity type ranking can be grouped in entity-centric, context-aware, and hierarchy-based. Figure 3 shows on which data such approaches are based. The entity-centric approaches look at the relation of the entity e with other entities in a background knowledge base following edges such as `<dbpedia-prop:wikiLink>` and `<owl:sameAs>`. Context-aware approaches exploit the co-occurrence of the entity e with other entities in the same textual context. Hierarchy-based approaches look at the structure of the type hierarchy and rank types based on it.

[9] The type hierarchy created in this way is available in the form of a small inverted index that provides for each type the path to the root and its depth in the hierarchy at `http://exascale.info/TRank`

[10] `http://data.sindice.com/trec2011/`

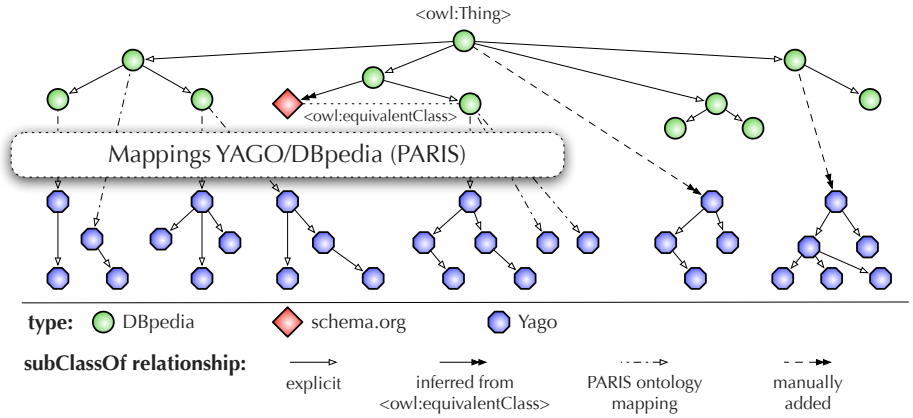

Fig. 2. The integrated type hierarchy.

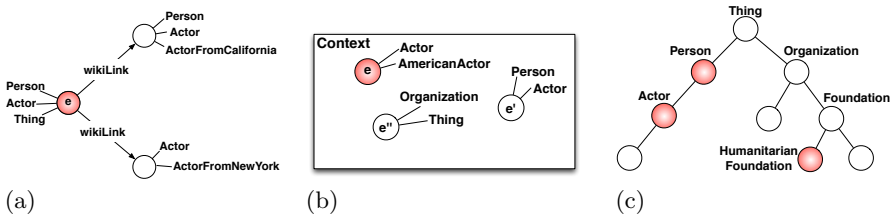

Fig. 3. (a) Entity-centric (b) Context-aware (c) Hierarchy-based type ranking.

4.2 Entity-Centric Ranking Approaches

We now turn to the detailed description of several techniques to rank entity types. The first group of approaches we describe only considers background information about a given entity and its types without taking into account the context in which the entity appears.

Our first basic approach (FREQ) to rank entity types is based solely on the frequency of those types in the background knowledge base ranking first the most frequent type of an entity. For example, the type Person has a higher frequency (and thus is more popular) than EnglishBlogger.

Our second approach (WIKILINK) exploits the relations existing between the given entity and further entities in the background knowledge base. Hence, we count the number of neighboring entities that share the same type. This can be performed by issuing the following SPARQL queries retrieving connected entities from/to e to rank $t_i \in T_e$:

```
SELECT ?x WHERE { <e> <dbpedia-prop:wikilink> ?x . ?x <rdfs:type> <t_i> }
SELECT ?x WHERE { ?x <dbpedia-prop:wikilink> <e> . ?x <rdfs:type> <t_i> }
```

For example, in Figure 3a to rank types for the entity e we exploit the fact that linked entities have also the type 'Actor' to rank it first.

In a similar way, we exploit the entity graph from the knowledge base by following `<owl:sameAs>` connections and observing the types attached to such URIs (SAMEAS):

```
SELECT ?x WHERE {<e> <owl:sameAs> ?x . ?x <rdfs:type> <t_i> }
```

Our next approach (LABEL) adopts text similarity methods. We consider the label of e and measure its TF-IDF similarity with other labels appearing in the background knowledge base in order to find related entities.[11] At this point, we inspect the types of the most related entities to rank the types of e. More specifically, we select the top-10 entities having the most similar labels to e and rank types based on the frequency of $t_i \in T_e$ for those entities.

4.3 Context-Aware Ranking Approaches

We describe approaches leveraging the entity context below. A first approach (SAMETYPE) taking into account the context c_e in which e appears is based on counting how many times each type $t_i \in T_e$ appears in the co-occurring entities $e' \in c_e$ also mentioned in the context. In this case, we consider a match whenever the same type URI is used by e and e', or when the type of e' has the same label as the type from e. For example, in Figure 3b we rank first the type 'Actor' for the entity e because it co-occurs with other entities of type Actor in the same context.

A slightly more complex approach (PATH) leverages both the type hierarchy and the context in which e appears. Given all entities appearing in the context $e' \in c_e$, the approach measures how similar the types are based on the type hierarchy. We measure the degree of similarity by taking the intersection between the paths from the root of the type hierarchy (i.e., `<owl:Thing>`) to $t_i \in T_e$ and to $t_j \in T_{e'}$. For instance, when ranking types for the entity 'Tom Hanks' in a context where also 'Tom Cruise' appears, we measure the similarity between the types by considering the common paths between the root of the type hierarchy and both types, e.g., "Thing-Agent-Person-Artist-Actor-AmericanTelevisionActors" and "Thing-Agent-Person-Artist-Actor-ActorsFromNewJersey" would be considered as highly similar. On the other hand, the 'Tom Hanks' type path "Thing-PhysicalEntity-CausalAgent-Person-Intellectual-Scholar-Alumnus-CaliforniaStateUniversity,SacramentoAlumni" is not very similar with the previous 'Tom Cruise' path. Hence, the approach ranks the 'AmericanTelevisionActors' type higher given the context in which it appears.

4.4 Hierarchy-Based Ranking Approaches

The more complex techniques described below make use of the type hierarchy and measure the depth of an entity type t_i attached to e in order to assess its relevance. We define the DEPTH ranking score of a type t_i as the depth of t_i in the type hierarchy. This approach favors types that are more specific (i.e., deeper in the type hierarchy).

[11] This can be efficiently performed by means of an inverted index over entity labels.

In some cases, the depth of an entity type in the hierarchy may not be enough. To detect the most relevant entity types, it might also be useful to determine the branch in the type hierarchy where the most compelling entity types are defined. In that context, we define a method (ANCESTORS) that takes into consideration how many ancestors of $t_i \in T_e$ are also a type of e. That is, if $Ancestors(t_i)$ is the set of ancestors of t_i in the integrated type hierarchy, then we define the score of t_i as the size of the set $\{t_j | t_j \in Ancestors(t_i) \wedge t_j \in T_e\}$. For example, in Figure 3c we rank first the type 'Actor' because 'Person' is its ancestor and it is also a type of e. On the other hand, the type 'Humanitarian Foundation' has a bigger depth but no ancestor which is also a type of e.

A variant of this approach (ANC_DEPTH) considers not just the number of such ancestors of t_i but also their depth. Thus,

$$ANC\_DEPTH(t_i) = \sum_{t_j \in Ancestor(t_i) \wedge t_j \in T_e} depth(t_j). \quad (1)$$

4.5 Learning to Rank Entity Types

Since *TRank* ranking approaches cover fairly different types of evidence (based on the entity-graph, the context, or the type hierarchy) to assess the relevance of a type, we also propose to combine our different techniques by determining the best potential combinations using a training set, as it is commonly carried out by commercial search engines to decide how to rank Web pages (see for example [15]). Specifically, we use decision trees [11] and linear regression models to combine the ranking techniques described above into new ranking functions. The decision tree method we used is M5 [21], which is specifically designed for regression problems. The effectiveness of this approach is discussed in Section 5.

4.6 Scalable Entity Type Ranking with MapReduce

Ranking types using the above methods for all the entities identified in a large-scale corpus using a single machine and SPARQL end-points is impractical, given the latency introduced by the end-point and the intrinsic performance limitations of a single node. Instead, we propose a self-sufficient and scalable Map/Reduce architecture for *TRank*, which does not require to query any SPARQL end-point and which pre-computes and distributes inverted indices across the worker nodes to guarantee fast lookups and ranking of entity types. More specifically, we build an inverted index over the DBpedia 3.8 entity labels for the entity linking step and an inverted index over the integrated *TRank* type hierarchy which provides, for each type URI, its depth in the integrated type hierarchy and the path to the root of the hierarchy. This enables a fast computation of the hierarchy-based type ranking methods proposed in Section 4.4.

5 Experiments

5.1 Experimental Setting

We created a ground truth of entity types mentioned in 128 news articles selected from the top news of each category from the New York Times (NYT) website

during the Feb 21 – Mar 7 2013 period. On average, each article contains 12 entities. After the entity linking step, each entity gets associated with an average of 10.2 types from our Linked Data collection. We crowdsourced the selection of the most relevant types by asking the workers to select the most relevant type given a specific textual context.

Crowdsourced Relevance Judgements. We used paid crowdsourcing to create the ground truth.[12] We decided to ask anonymous Web users rather than creating the ground truth ourself as they are a real sample of Web user who could benefit from the envisioned application. Each task, which was assigned to 3 different workers from the US, consists of asking the most relevant type for 5 different entities, and was paid $0.10 for entities without context and $0.15 for entities with a context. Additionally, we allowed the workers to tag entities that were wrongly extracted, and to add an additional type if the proposed ones were not satisfactory. Overall, the relevance judgement creation cost $190.

In order to better understand how to obtain the right information from the crowd, we ran a pilot study where we compared different task designs for the entity type judgement task. We assessed the approach of asking the crowd to select all types which are relevant for an entity given its context as compared to asking which is the best type. Given the results of the pilot study, we selected the design that asks the worker to pick the *best* type only as this also best models the use case of showing one single entity type to a user browsing the Web. To generate our ground truth out of the crowdsourcing results, we consider as relevant each type which has been selected by at least one worker, in order to obtain binary judgements. We take the number of workers who selected a type as its relevance score in a graded relevance setting.

Evaluation Measures. As the main evaluation measures for comparing different entity type ranking methods, we use Mean Average Precision (MAP). Average Precision (AP) for the types T_e of an entity e is defined as

$$AP(T_e) = \frac{\sum_{t_i \in T_e} rel(t_i) \cdot P@i}{|Rel(T_e)|} \tag{2}$$

where $rel(t_i)$ is 1 if t_i is a relevant type for the entity e and 0 otherwise, $Rel(T_e)$ is the set of relevant types for e, and $P@i$ indicates Precision at cutoff i. MAP is defined as the mean of AP over all entities in the collection.

MAP is a standard evaluation measure for ranking tasks which consider binary relevance: A type t_i is either correct or wrong for an entity e. Since the original relevance judgements are not binary (i.e., more than one worker can vote for a type and thus have a higher relevance value than a type with just one vote), we also measure the Normalize Discounted Cumulative Gain (NDCG) [12], which is a standard evaluation measure for ranking tasks with non-binary relevance judgements. NDCG is defined based on a gain vector G, that is, a vector containing the relevance judgements at each rank. Then, the *discounted*

[12] We run our tasks over the Amazon MTurk platform. The collected data and task designs are available for others to reuse at `http://exascale.info/TRank`

cumulative gain measures the overall gain obtained by reaching rank k putting more weight at the top of the ranking: $DCG[k] = \sum_{j=1}^{k} G[j]/(log_2(1+j))$. To compute the final NDCG, we normalize it by dividing DCG by its optimal value obtained with the optimal gain vector which puts the most relevant results first.

5.2 Dataset Analysis

Out of the NYT articles we have crawled, we created four different datasets to evaluate and compare our approaches for the entity type ranking task. First, we use a collection consisting exclusively of entities and their types as extracted from the news articles. This collection is composed of 770 distinct entities: out of the original 990 extracted entities we consider only those with at least two types and we removed the errors in NER and entity linking which were identified by the crowd during the relevance judgements.

Sentence Collection. We built a Sentence collection consisting of all the sentences containing at least two entities. In this and the following collections we asked the human assessor to judge the relevance of a type given a context (e.g., a sentence). This collection contains 419 context elements composed of an average number of 32 words and 2.45 entities each.

Paragraph Collection. We constructed a collection consisting of all the paragraphs longer than one sentence and containing at least two entities having more than two types. This collection contains 339 context elements composed of an average number of 66 words and 2.72 entities each.

3-Paragraphs Collection. The last collection we have constructed contains the largest context for an entity: the paragraph where it appears together with the preceding and following paragraphs in the news article. This collection contains 339 context elements which are composed on average of 165 words each. The entire context contains on average 11.8 entities which support the relevance of the entities appearing in the central paragraph.

5.3 Experimental Results

Table 1 shows the overall effectiveness obtained by the proposed approaches. When we compare the results obtained among the different collections (i.e., entity-only, sentence, paragraph, and 3 paragraphs) we observe that effectiveness values obtained without context are generally higher, supporting the conclusion that the type ranking task for an entity without context is somehow easier than when we need to consider the story in which it is mentioned. Among the entity centric approaches, in most of the cases the best approach is WIKILINK-OUT, that is, the approach that follows the `<dbpedia-prop:wikiLink>` edges starting from the entity e and that checks the frequency of its types among its connected entities. Among the context-aware approaches, the PATH method performs best. Interestingly, the hierarchy-based approaches clearly outperform the methods

Table 1. Type ranking effectiveness for different textual contexts. Statistically significative improvements (t-test $p < 0.05$) of the regression methods over the best ranking approach are marked with * and of the best hierarchy-based method over the best method from the other groups with †.

	Entity-only		Sentence		Paragraph		3-Paragraphs	
Approach	**NDCG**	**MAP**	**NDCG**	**MAP**	**NDCG**	**MAP**	**NDCG**	**MAP**
FREQ	0.6284	0.4659	0.5409	0.3758	0.5315	0.3739	0.5250	0.3577
WIKILINK-OUT	0.6874	0.5406	0.6050	0.4521	0.6063	0.4550	0.6059	0.4444
WIKILINK-IN	0.6832	0.5342	0.5907	0.4213	0.5879	0.4254	0.5853	0.4143
SAMEAS	0.6848	0.5328	0.6049	0.4310	0.5990	0.4221	0.6172	0.4417
LABEL	0.6672	0.5067	0.6075	0.4265	0.5883	0.4104	0.5821	0.4034
SAMETYPE	-	-	0.6024	0.4452	0.5917	0.4327	0.5813	0.4256
PATH	-	-	0.6507	0.4956	0.6538	0.4974	0.6315	0.4742
DEPTH	0.7432	0.6128	0.6754	0.5385	0.6797	0.5475	0.6741	0.5354
ANCESTORS	0.7424	0.6154	**0.6967**†	**0.5637**†	**0.6949**†	**0.5662**†	**0.6879**†	**0.5562**†
ANC_DEPTH	**0.7469**†	**0.6236**†	0.6832	0.5488	0.6885	0.5546	0.6796	0.5423
DEC-TREE	**0.7614**	**0.6361**	**0.7373***	**0.6079***	**0.7979***	**0.7019***	**0.7943***	**0.6914***
LIN-REG	0.7373	0.6079	0.6906	0.5579	0.6987	0.5702	0.6899	0.5529

looking at the context or at the entity itself. The relatively simple DEPTH approach performs very effectively. The approaches looking at the ancestor of a type in the integrated hierarchy are the most effective approaches for ranking entity types among the ones we propose. Nevertheless, there are cases in which context-aware approaches rank types better than hierarchy-based ones. For example, in some document "Mali" co-occurs with "Paris", "Greece", and "Europe". The top-3 results selected by ANCESTORS for "Mali" are "LeastDevelopedCountries", "LandlockedCountries", and "French-speakingCountries", which are all non-relevant since they are too specific with respect to the context. In contrast, the top-3 types selected by PATH: "PopulatedPlace", "Place", and "Country", are all relevant according to the crowd.

To evaluate the combination of approaches using machine-learning methods, we run 10-fold cross validation over 7884, 11875, 11279, and 11240 data points in the four different collections. Out of the ranking approaches we have proposed, we selected 12 features which cover all the different methodologies (i.e., entity-centric, context-aware, and hierarchy-based) to train regression models for entity type ranking. We observe that the best performing method is the one based on decision trees, which significantly outperforms all other approaches.

Figure 4 shows the evolution of MAP and NDCG values by looking at entities with a different number of associated types. Generally speaking, we see that entity having many different types are more difficult to handle. On the other hand, even for the simple approach FREQ, when few types are assigned to an entity we obtain effective results. On the right side of Figure 4, we can observe the robustness of DEC-TREE over entities with an increasing number of types.

Crowd-powered Entity Type Assignment. In some cases the knowledge base may not contain types that are good enough. For example, some entities have only `<owl:Thing>` and `<rdfs:Resource>` attached to them. In such cases, we asked the crowd to suggest a new type for the entity they are judging. While extend existing LOD ontologies with additional schema element is not the focus of this paper, we observe that this can be easily achieved by means of crowdsourcing.

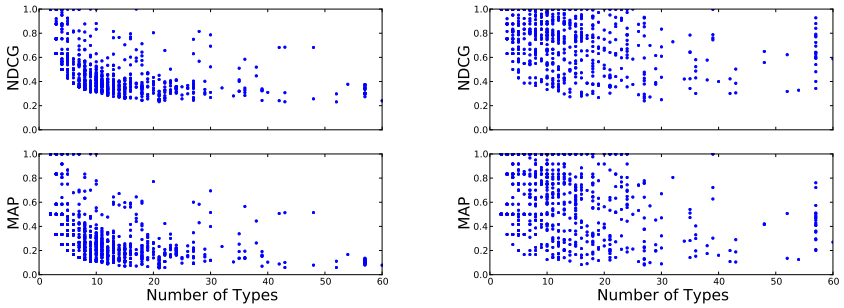

Fig. 4. MAP and NDCG of FREQ (left) and DEC-TREE (right) for entities with different numbers of types on the 3-paragraphs collection.

The suggestion of new types from the crowd may also suggest an error at the entity linking step (i.e., a wrong URI has been assigned to the entity mention). Some examples of crowd-originated entity types are shown in Table 2.

TRank *Scalability.* We ran the MapReduce *TRank* pipeline over a sample of CommonCrawl containing `schema.org` annotations. Upon writing this paper, CommonCrawl is formed by 177 valid crawling segments, accounting for 71TB of compressed Web content. We uniformly sampled 1TB of data over the 177 segments, and kept only the HTML content with `schema.org` annotations. This resulted in a corpus of $1,310,459$ HTML pages, for a total of 23GB (compressed). Our MapReduce testbed is composed of 8 slave servers, each with 12 cores at 2.33GHz, 32GB of RAM and 3 SATA disks. The relatively small size of the 3 Lucene inverted indices ($\sim$ 600MB) used by the *TRank* pipeline allowed us to replicate the indices on each single server (transparently via HDFS). In this way, no server represented a read hot-spot or, even worse, a single point of failure.We argue that the good performance of our MapReduce pipeline is in majorly due to the use of small, pre-computed inverted indices instead of expensive SPARQL queries.
Processing the corpus on such a testbed takes 25 minutes on average, that is, each server runs the whole *TRank* pipeline at 72 documents per second. Table 3 shows a performance breakdown for each component of the pipeline. The value reported for "Type Ranking" refers to the implementation of ANCESTORS, but it is comparable for all the other techniques presented in the paper (except

Table 2. Examples of crowd-originated entity types.

Entity Label	Existing Types	Crowd Suggested Type
David Glassberg	Alumnus, Resource, Northwestern University Alumni, American television journalists	New York City policeman
Fox	Thing, Eukaryote	Television Network
Bowie	Minor league team, Minor league sports team	Musical Artist
Atlantic	Resource, Populated Place	Ocean
European Commission	Type of profession, Landmark	Governmental Organizations
Childress	Thing, Resource	Locality

Table 3. Efficiency breakdown of the *TRank* MapReduce pipeline.

Text Extraction	NER	Entity Linking	Type Retrieval	Type Ranking
18.9%	35.6%	29.5%	9.8%	6.2%

Table 4. CommonCrawl sample statistics.

Domain	% in corpus
youtube.com	39.65
blogspot.com	9.26
over-blog.com	0.67
rhapsody.com	0.54
fotolog.com	0.52

Schema.org type	% in corpus
http://schema.org/VideoObject	40.79
http://schema.org/Product	32.66
http://schema.org/Offer	28.92
http://schema.org/Person	20.95
http://schema.org/BlogPosting	18.97

the ones based on the Learning to Rank approach, which we did not test in MapReduce).

The observed `schema.org` class distributions almost overlaps with the one previously found by [17] (see Table 4).[13] Table 5 shows the most frequent entity types selected by *TRank* for entities contained in our sample of CommonCrawl. We can observe how *TRank* types refer to specific entities mentioned in topic-specific pages as, for example, `<yago:InternetCompaniesOfTheUnitedStates>` entities that are contained in `<http://schema.org/Product>` Web pages.

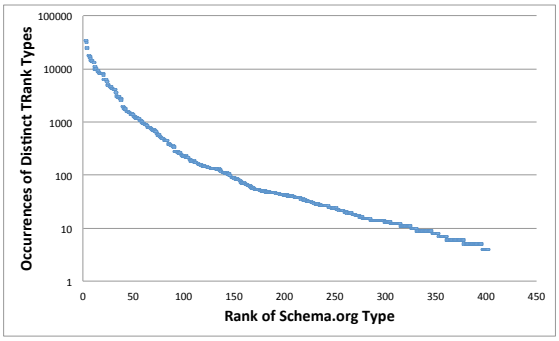

Fig. 5. Occurrences of distinct *TRank* types in CommonCrawl (log scale).

Figure 5 shows the variety of entity types selected by *TRank* for Web pages annotated with different `schema.org` classes. We clearly recognize a power-low distribution where the top `schema.org` classes contain very many different entity types while most of the others have a low diversity of entity types.

6 Conclusions

In this paper, we focused on the new task of ranking types for online entities given some textual context and links to background knowledge bases. Numerous

[13] More statistics can be found at `http://exascale.info/TRank`

Table 5. Co-occurrences of top Schema.org annotations with entity types.

Schema.org type	top-3 most frequent *TRank* types
http://schema.org/VideoObject	<dbpedia-owl:GivenName> <dbpedia-owl:Settlement> <dbpedia-owl:Company>
http://schema.org/Product	<yago:InternetCompaniesOfTheUnitedStates> <yago:PriceComparisonServices> <dbpedia-owl:Settlement>
http://schema.org/Offer	<yago:InternetCompaniesOfTheUnitedStates> <yago:PriceComparisonServices> <dbpedia-owl:Company>
http://schema.org/Person	<dbpedia-owl:GivenName> <dbpedia-owl:Company> <yago:FemalePornographicFilmActors>

applications can be developed once the most relevant entity types are correctly determined, including SERP enrichment, faceted browsing, and document summarization. We proposed different classes of ranking approaches and evaluated their effectiveness using crowdsourced relevance judgments. We also evaluated the efficiency of the proposed approach by taking advantage of inverted indices for fast access to entity and type hierarchy information and of a MapReduce pipeline for efficient entity type ranking over a Web crawl.

Our experimental results show that the approaches considering the relations between entity types in the overall type hierarchy outperform the other classes of approaches. A regression model learned over training data combining the different classes of ranking approaches significantly outperforms the individual ranking approaches, reaching a Mean Average Precision value of 0.70. As future work, we aim at improving *TRank* effectiveness by differentiating types and roles to design new ranking approaches based both on natural types and on the interaction among entities in the context. In addition, we plan to test the behavior of our system with different domains and ontologies, and to evaluate the user impact of entity typing by running a large-scale experiment using a browser plugin to display contextual entity types while the user is surfing.

7 Acknowledgments

This work was supported by the Swiss National Science Foundation under grant number PP00P2_128459, and by the Haslerstiftung in the context of the Smart World 11005 (Mem0r1es) project.

References

1. C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. DBpedia - A crystallization point for the Web of Data. *Journal of Web Semantics*, pages 154–165, 2009.
2. K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, pages 1247–1250, 2008.
3. S. Campinas, D. Ceccarelli, T. E. Perry, R. Delbru, K. Balog, and G. Tummarello. The Sindice-2011 dataset for entity-oriented search in the web of data. In *1st International Workshop on Entity-Oriented Search (EOS)*, pages 26–32, 2011.

4. M. Ciaramita and Y. Altun. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *EMNLP*, pages 594–602, 2006.
5. H. Cunningham, K. Humphreys, R. Gaizauskas, and Y. Wilks. GATE: a general architecture for text engineering. In *ANLC*, pages 29–30, 1997.
6. G. Demartini, D. E. Difallah, and P. Cudré-Mauroux. ZenCrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *WWW*, pages 469–478, 2012.
7. Y. Fang, L. Si, Z. Yu, et al. Purdue at TREC 2010 Entity Track: A Probabilistic Framework for Matching Types Between Candidate and Target Entities. In *TREC*, 2010.
8. J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL*, pages 363–370, 2005.
9. J. R. Finkel and C. D. Manning. Joint parsing and named entity recognition. In *NAACL*, pages 326–334, 2009.
10. A. Gangemi, A. G. Nuzzolese, V. Presutti, F. Draicchio, A. Musetti, and P. Ciancarini. Automatic typing of dbpedia entities. In *ISWC*, pages 65–81, 2012.
11. G. Holmes, M. Hall, and E. Frank. Generating rule sets from model trees. In *AI*, pages 1–12, 1999.
12. K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *TOIS*, pages 422–446, 2002.
13. A. Kalyanpur, J. W. Murdock, J. Fan, and C. Welty. Leveraging community-built knowledge for type coercion in question answering. In *ISWC*, pages 144–156, 2011.
14. R. Kumar and A. Tomkins. A characterization of online search behavior. *IEEE Data Eng. Bull.*, 2009.
15. T.-Y. Liu. Learning to rank for information retrieval. *FTIR*, pages 225–331, 2009.
16. C. Matuszek, J. Cabral, M. Witbrock, and J. DeOliveira. An introduction to the syntax and content of cyc. In *AAAI Spring Symposium*, 2006.
17. H. Mühleisen and C. Bizer. Web data commons - extracting structured data from two large web corpora. In *LDOW*, 2012.
18. D. Nadeau. *Semi-supervised named entity recognition: learning to recognize 100 entity types with little supervision.* PhD thesis, 2007.
19. D. Nadeau, P. D. Turney, and S. Matwin. Unsupervised named-entity recognition: generating gazetteers and resolving ambiguity. In *AI*, pages 266–277, 2006.
20. J. Pound, P. Mika, and H. Zaragoza. Ad-hoc object retrieval in the web of data. In *WWW*, pages 771–780, 2010.
21. J. R. Quinlan. Learning with continuous classes. In *AI*, pages 343–348, 1992.
22. F. M. Suchanek, S. Abiteboul, and P. Senellart. Paris: Probabilistic alignment of relations, instances, and schema. *PVLDB*, pages 157–168, 2011.
23. F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *WWW*, pages 697–706, 2007.
24. A. Tonon, G. Demartini, and P. Cudré-Mauroux. Combining inverted indices and structured search for ad-hoc object retrieval. In *SIGIR*, pages 125–134, 2012.
25. T. Tylenda, M. Sozio, and G. Weikum. Einstein: physicist or vegetarian? summarizing semantic type graphs for knowledge discovery. In *WWW*, pages 273–276, 2011.
26. C. Welty, J. W. Murdock, A. Kalyanpur, and J. Fan. A comparison of hard filters and soft evidence for answer typing in watson. In *ISWC*, pages 243–256, 2012.
27. S. E. Whang and H. Garcia-Molina. Joint Entity Resolution on Multiple Datasets. *The VLDB Journal*, 2013.