
Article:
Tonon, A., Catasta, M., Prokofyev, R. et al. (2015) Contextualized ranking of entity types based on knowledge graphs. Journal of Web Semantics. ISSN: 1570-8268

https://doi.org/10.1016/j.websem.2015.12.005

Contextualized Ranking of Entity Types Based on Knowledge Graphs

Alberto Tonon[a,*], Michele Catasta[b,**], Roman Prokofyev[a,*], Gianluca Demartini[c], Karl Aberer[b,**], Philippe Cudré-Mauroux[a,*]

[a] *eXascale Infolab, University of Fribourg – Switzerland*
[b] *EPFL, Lausanne – Switzerland*
[c] *Information School, University of Sheffield – UK*

Abstract

A large fraction of online queries target 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 graph 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 these 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 knowledge graph structure interconnecting entities and types. An extensive experimental evaluation over several document collections at different levels of granularity (e.g., sentences, paragraphs) 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.

Keywords: Entity typing, Ranking, Context, Crowdsourcing, Knowledge Graphs

1. Introduction

A large fraction of online queries target entities [1]. 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 taken from a background knowledge graph. This search task is known as ad-hoc object retrieval [2, 3], that is, finding an entity described by a keyword query in a structured knowledge graph. 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 graph. It is possible, for example, to display pictures, a short textual description, and related entities.

One interesting entity facet which can be displayed in the SERP is its *type*. In public knowledge graphs such as Freebase, entities are associated with several types. For example, the entity 'Peter Jackson' in Freebase[1] 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

*name.surname@unifr.ch
**name.surname@epfl.ch
g.demartini@sheffield.ac.uk

[1] `http://www.freebase.com/edit/topic/en/peter_jackson`

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 the Google 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 to the best effectiveness. We also assess the scalability of our approach by designing and evaluating a Map/Reduce version of our ranking process over a large sample of the CommonCrawl dataset[2] exploiting existing `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.
- 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 [4] and Freebase by means of crowdsourced relevance judgements.
- A scalable version of our type ranking approach evaluated over a large annotated Web crawl.
- The proposed techniques are available as an open-source library[3] as well as an online web service[4].

The present work is based on our previous contribution on type ranking [5]. However, we extend our previous article in several different ways: We propose a new context-aware approach to rank entity types that extends the notion of context in which an entity appears to exploit the text surrounding it in addition to other co-occurring entities (Section 6.4), and a new method that mixes different features coming from both the knowledge base, including entity popularity, and the type hierarchy (Section6.5. We report additional detail on the methods we used to build our text collection, including a pilot study we did in order to evaluate the best task design to collect relevance judgements (Section 7). We add a discussion on the relation among the features we take into consideration and on the comparison between hierarchy based and context-aware methods for ranking entity types (Section 9).

The rest of the paper is structured as follows. We start below by describing related work from entity-search and ad-hoc object retrieval. Then, we introduce the most important concepts in the Web of Data (Section 3) to formally define our new type ranking task in Section 4. In Section 5 we present the architecture of our system, and in Section 6 propose a series of approaches to solve it based on collection statistics, type hierarchies, and entity graphs. Section 8 presents experimental results comparing the effectiveness of our various entity ranking approaches over different document

[2] `http://commoncrawl.org/`

[3] `https://github.com/MEMOR1ES/TRank`

[4] `http://trank.exascale.info`

collections and type hierarchies as well as a scalability validation of our Map/Reduce implementation over a large corpus. Finally, we conclude in Section 10.

2. Related Work

Entity-centric data management is an remerging area of research at the overlap 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 pre-existing knowledge graph.

Named Entity Recognition and Ranking. 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., [6, 7]). While this is useful for application 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 ranking has been studied for a few years. Historically, the first entity-oriented task being addressed was expert finding [8] where the focus is on one specific entity type, that is, people. The goal is to find people who are knowledgeable about the requested topic. After this, works have looked at how to search for multiple entity types. Early approaches on entity ranking focused on entities of different types which are present in Wikipedia [9, 10]. In the IR 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 [11]). Moreover, the IR community has organized workshops on entity search topics at the major research venue [12, 13].

More recently, we have proposed and hybrid approach to rank entity identifiers as answer to a Web search query [3]. We used both standard IR methods based on inverted indices as well as a structured search approach that exploits the graph structure (also including type information) connecting entities each other to improve search effectiveness. In [14] authors show how the number of entities used for the graph-based search step influences search effectiveness. Related to this is the aggregation of all data available about a specific entity [15], also including its types.

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 [16] 18 types are considered. In [17, 18], authors propose a NER system to recognize 100 entity types using a supervised approach. The starting point to define the 100 entity types is the BBN linguistic collection[6] which includes 12 top types and 64 subtypes.

Entity Types. The Semantic Web community has been creating large-scale knowledge graphs defining a multitude of entity types. Efforts such as YAGO [4] have assigned to LOD entities many types by combining Wikipedia categories and WordNet senses. More recently, projects such as DBpedia [19] and Freebase [20] have collected large collections of structured representations of entities along with their related types. Such knowledge graphs hence represent extremely valuable resources when working on entity type ranking as we do in this paper.

An early work about ranking entity types is [21] where authors propose a method to select the best type of the result for a Web search query. Similarly, in [22] authors propose methods to select the best type given a query by exploiting the type hierarchy from a background knowledge graph. As compared to this, we aim to rank types assigned to entities mentioned on the Web and (as pointed out in [22] as particularly challenging) to select the right granularity of types from the background type hierarchy.

In a recent demo [23], the task of selecting the most relevant types 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

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

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

is described. Similarly to that work, we build our approaches using large knowledge graphs such as YAGO and DBpedia. In *Tipalo* [24], the authors propose an algorithm to extract entity types based on the natural language description of the entity taken from Wikipedia. *PEARL* [25] is an approach that selects the most appropriate type for an entity by leveraging a background textual corpus of co-occurring entities and performing fuzzy pattern matching for new entities. We also rely on a background n-gram corpus to compute type probabilities (See Section 6.4), however, the task the authors tackle is different from ours: while their goal is to find the correct type of emerging entities (that is, entities not yet present in a knowledge graph), we rank the types of entities already contained in the knowledge graph, assuming they are correct. Yao et al. also worked on extracting entity types from text [26]; However, contrary to our case, their approach is not bound to a fixed ontology and extracts entity types coming from textual surface-form expressions (for example, "Jan Zamoyski, a magnate") without linking them to any knowledge graph.

Related Applications. Several applications of our techniques could be based on existing work. For instance, entity-type ranking could be applied on open-domain Question Answering [27], where candidate answers are first generated and later on filtered based on the expected answer types. For systems like Watson [28], 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 [29], 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. The Knowledge Graph

The task we address in this paper is addressed on top of the Linked Open Data (LOD) cloud[7]. This is a large collections of datasets describing entities in structured format. Popular datasets in LOD include DBPedia, Freebase, and Geonames.

Each entity in these dataset is uniquely identified by an http URI and is described by means of RDF triples, that is, factual statements including a subject (i.e., and entity URI), a predicate, and an object. Predicates are taked from a predefined schema (i.e., an ontology) and objects may be either other entity URIs (e.g., URI_1 *is_married_to* URI_2) or textual elements (e.g., URI_1 *has_name* 'John Doe').

In this work we focus on a specific set of predicates in LOD datasets that indicate entity type information. For a target entity, we collect all indidated types. In detail, we look at the object values of the set of triples containing as subject the target entity URI and as predicate one of those indicating type information. Such set of objects represent all correct types of the target entity. The task we address in this paper is to produce a ranked list of correct types with the goal of ranking first the most relevant types for the entity also based on the textual context where the entity appears.

4. Task Definition

Given a knowledge graph 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, `<dp:Tom_Cruise>`[8] has an `<owl:sameAs>` connection to `<fb:Tom_Cruise>` which allows us to add the new type `<fb:fashionmodels>`.

The context c_e of an entity e is defined as 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' shall probably be different given one or the other context. The different context types we consider

[7] `http://linkeddata.org`

[8] In the rest of this article we refer to DBpedia resources with the namespace `dp`, to DBpedia ontology entries with `dpo`, and to Freebase resources with `fb`.

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 `<dp:Politician>` is a subclass of `<dp:Person>`. Knowing the relations among types and their depth in the hierarchy is often helpful to 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 8.

5. 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. 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[9]. Then, given an entity URI we retrieve (for example, thanks to a SPARQL query to a knowledge graph) all the types attached to the entity. In this way, we obtain types such as `<owl:Thing>`, `<dpo:EFF_Pioneer_Award_recipients>` and `<dpo:English_bloggers>` for the entity `<dp:Tim_Berners-Lee>`. Finally, our system produces a ranking of the resulting types based on evidences computed by using different methods exploiting the textual context where the entity has been mentioned. A summary of the different steps involved in are depicted in Figure 1.

We briefly describe the entity extraction and entity linking components below. The focus of the rest of this paper will then be on the definition and experimental comparison of different ranking functions for entity types.

Entity Extraction. The first component of the TRank^{++} pipeline takes as input a document collection and performs NER, that is, the identification of entity mentions in the text. Entities that can be accurately identified are persons, locations, and organizations. The current implementation of our system adopts a Conditional Random Field approach to identify entities [31].

Entity Linking. The following step is entity linking, which aims at assigning a URI to an entity mention identified in the previous step. The goal is to disambiguate an entity (e.g., 'Michael Jordan') by uniquely identifying it (e.g., the former NBA basketball player). In order to obtain a URI for the entity, we rank candidate URIs from DBPedia 3.8 using an inverted index by TF-IDF similarity on the label. Note that the focus of this paper is not on improving the state-of-the-art in named entity recognition or linking, thus we apply existing approaches for these steps.

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 [32] as well as based on the widespread success of Wikipedia combined with information extraction algorithms [19, 4]. However, the lack of alignment 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 [33] to build a coherent tree of 447,260 types, rooted on `<owl:Thing>` and with a max 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

[9] This is the same baseline approach used in [30] for Entity Linking.

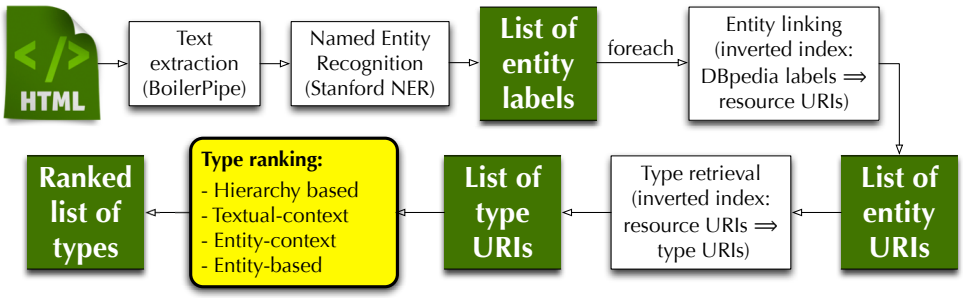

Figure 1: The TRank^{++} Architecture.

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 4 domain experts) to obtain a single type tree (rather than a forest of 5 trees).[10] 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[11] [34] to retrieve the types, which consists of about 11 billion RDF triples.

6. Approaches to Entity Type Ranking

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 graph following edges such as `<dpo: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 of e based on that.

[10]The such created type hierarchy 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`

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

6.1. Entity-Centric Ranking Approaches

We now turn to the 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 graph 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 graph. Hence, we count the number of neighboring entities that share the same type. This can be for example be performed by issuing the following SPARQL queries retrieving connected entities from/to e:

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

```
SELECT ?x
WHERE  { ?x <dpo: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 knowledge graph by following `<owl:sameAs>` connections and observing the types attached to such URIs (SAMEAS):

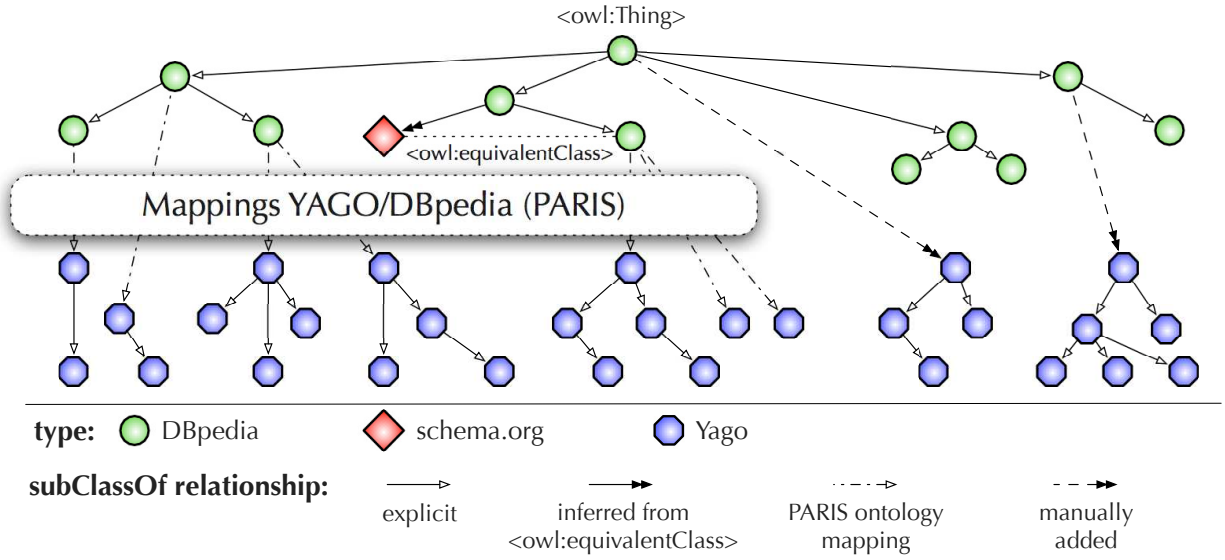

Figure 2: The integrated type hierarchy.

```
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 graph in order to find related entities[12]. 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.

6.2. 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).

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 type of e. That is, if $Ancestors(t_i)$ is the set of ancestors of t_i in the 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,

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

6.3. Entity-Based 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

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

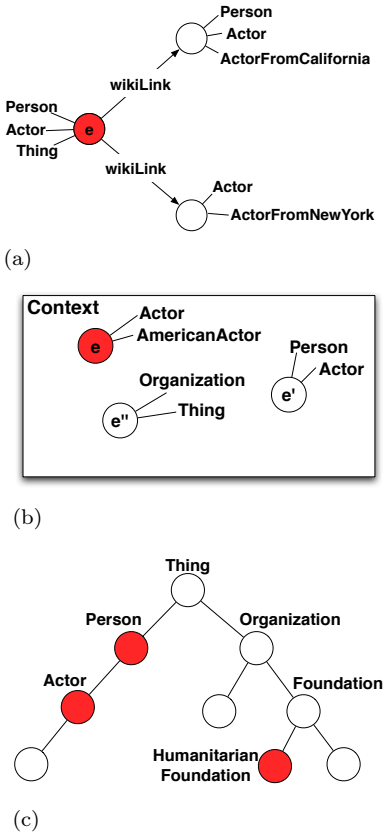

Figure 3: (a) Entity-centric (b) Context-aware (c) Hierarchy-based type ranking.

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.

6.4. Text-Based Context-Aware Approaches

We now propose two context-aware approaches that exploit the *textual* context in which the considered entities appear. Formally, we want to rank the types of an entity e, appearing in a window of text $(w_{-k}, w_{-k+1}, \ldots, w_0 = e, w_1, \ldots, w_k)$, where e is a mention of the entity occurring inside the window. In this method, we rank types according to the probability of seeing an entity of type t given the tokens w_{-k}, w_{-k+1}, $\ldots$, w_{-1}, w_1, $\ldots$, w_k.

We use the text of the Wikipedia Webpages in order to compute relevant statistics for type ranking. More precisely, we extend the user-annotated entities by using a state-of-the-art entity linking method: DBpedia Spotlight [35]. All the linked entities are then substituted by special tokens containing their entity types. For example, if we suppose that an entity e having types t_0 and t_1 appears in a certain window of text $\bar{w}$, we replicate twice $\bar{w}$ where the first time we replace the entity with t_0, and the second time with t_1. The resulting sequence of tokens is finally split into n-grams, and aggregated counts are calculated. Formally, we compute the probability of an entity type t given a window of text $\bar{w}$ by averaging the probability of finding an entity of type t in each individual n-gram we can extract from $\bar{w}$. The probability of a type t given an n-gram ng is computed as shown in Equation 2, where T is the set of all considered types.

$$\Pr(t \mid ng) = \frac{\text{Count}(t \mid ng)}{\sum_{j \in T} \text{Count}(t_j \mid ng)}. \qquad (2)$$

It is worth noting that, since every entity is an instance of the most general type, the probability of `<owl:Thing>` given any text is always 1. Unfortunately, as a consequence of the sparsity of natural language data, we do not always have evidence of the occurrence of all considered types, given a certain textual context. This is exacerbated when the knowledge graph taken into consideration contains many entity types. We address this issue using a popular technique from statistical machine translation known as Stupid Back-off smoothing [36], in

which we fall back to estimating the probability using $(n-1)$-grams if the original n-gram is not contained in our background corpus. For unigrams, the probability estimate is the probability of the given type.

Finally, the score that our first text-based context-aware approach (NGRAMS0) assigns to a DBpedia type given a window of text $\bar{w}$ is computed as shown in Equation 3. NGrams$(\bar{w}, n)$ is a function returning all the n-grams composing w. During our experimental evaluation, we fix the length of the textual context $\bar{w}$ to 5 tokens (that is, we take two tokens before an entity type, an two tokens after the entity type).

$$\text{Score}(t, \bar{w}) = \frac{\sum_{ng \in \text{NGrams}(\bar{w})} \text{NGramScore}(t \mid ng)}{|\text{NGrams}(\bar{w})|} \quad (3)$$

As one can notice, NGRAMS0 always prioritizes coarser types. In order to mitigate this phenomenon, we devise a method drawing inspiration from our recent work [37]. Our second text-based context-aware approach, NGRAMS1, models the probability of seeing an instance of a certain entity type given a window of text as a flow crossing the type hierarchy starting from the root (with probability 1) and descending the tree until either a leaf is reached or the flow of probability is broken (that is, we reach a node that does not transmit any probability to its children). Additionally, in order to avoid cases in which deeper or coarser types are always prioritized, when computing the score of an entity type we take into consideration the entropy its children's probabilities, and that of its sibling's probabilities, too. For example, suppose that the probability that flowed to `dpo:Actor` is scattered almost equally among all its 465 children. In this case `dpo:Actor` should be prioritized as it is more "informative" than each of its children. The main idea supporting this decision is the fact that when the probability mass of the current type in the hierarchy is scattered among many of its children, this can give us clues about how the current type should be ranked with respect to its children. In Section 9, we give more detail on the relation between entropy, number of children, and relevance.

Despite using the same concepts, we cannot directly apply the algorithm we proposed in our previous work since it was designed to return only one type. What we propose instead is a generalization of this approach using the probability of the current node and the entropy of its children as two features to assign a score to an entity type. This is appropriate in our context as we can exploit the labeled data of our testset (see Section 7 for more information). Specifically, we use a decision tree [38] to assign a score to the current type, given a certain window of text, by exploiting the following features:

1. `NGRAMS0`, the score of the NGRAMS0 method, playing the role of the type probability;
2. `ratioParent`, the ratio between the current type probability and the probability of its parent;
3. `nChildren`, the number of children to which some probability mass flows;
4. `hChildren`, the entropy of the probabilities of the children;
5. `nSiblings`, the number of siblings of the current node having some probability mass;
6. `hSiblings`, the entropy of the probabilities of the siblings.

Section 8 gives more technical detail and reports on the evaluation of NGRAMS0 and NGRAMS1; in addition, the relation between the features we selected and relevance is discussed in Section 9.

6.5. Mixed Approaches

Below, we report on techniques that exploit several features to rank entity types.

Mixing Evidence from Type Hierarchy and Knowledge Base. The first technique we propose, KB-HIER, uses decision trees [38] to combine features that come from the entity whose types are ranked, from the type hierarchy and from the knowledge base. The features we consider in that context are:

1. `popularity` of the entity in the knowledge base, measured by computing the number of triples with the given entity as the subject.
2. `nTypes`, that is, the number of types connected to the entity.
3. `nChildren`, `typeDepth`, `nSiblings`, the number of children, depth in the type hierarchy, and number of siblings of the entity type taken into consideration.

We focus on the interplay among those features as we believe that it is the most interesting aspect of this approach.

Learning to Rank Entity Types. Finally, since TRank^{++} ranking approaches cover fairly different types of evidences (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 [39]). Specifically, we use decision trees [40] and linear regression models to combine the ranking techniques described above into new ranking functions. The decision tree method we used is M5 [41], which is specifically designed for regression problems.

The effectiveness of these two approaches is discussed in Section 8, while in Section 9 we discuss the relations between the various features we use.

6.6. 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 indexes across the worker nodes to guarantee fast lookups and ranking of entity type. 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 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 6.2.

7. Crowdsourced Relevance Judgements

To create the ground truth judgements for the best types to be selected for an entity given a context we used paid crowdsourcing[13]. 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 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 in a context. Additionally, we allow the worker to annotate the entity as an error which could have happened either at the extraction or linking level, and to add an additional type if the proposed ones were not satisfactory. Overall, the relevance judgement creation costed \$190.

7.1. Pilot Study

In order to better understand how to obtain accurate relevance judgements from the crowd, we ran a pilot study where we compared three different task designs for the entity type relevance judgement. In order to compare our task designs, we assigned each different task to 10 workers for a total of 30 requests and a budget of \$6. The workers were requested to read a paragraph of text and rank the types of 6 entities having, on average, 11 entity types each.

Two of the task designs we analyzed are based on the interface depicted in Fig. 4 (left) but differ from the fact that in one the workers could select multiple relevant types, while in the other they could only select one relevant type (they had to explicitly mark all the other types as irrelevant). The rationale behind this choice was that having to express an opinion for each entity type forces the worker to read the type label, thus, reducing spam (i.e., random selection of one type). The high frequency of "yes" we recorded in the first task design (6.76 on average) led us to understand that the workers misinterpreted the crowdsourced task and signaled as relevant all the types that they know are correct for the entity displayed (which was not the goal as clearly explained in the task instructions), while the second task design was not popular among the workers due to the high number of clicks it required (only 7 workers out of 10 completed the task), leading to too potentially long delays to obtain judgements for all our datasets.[14] Finally, the interface of the third task design we considered is similar to the one shown in Fig. 4 (right). In this case, the users were still allowed to select only one relevant

[13] 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`

[14] During the pilot study we monitored the main web forums (e.g., `http://www/mturkforum.com`) where crowd workers exchange information about tasks and we noticed that this task design was criticized as using "mega-bubbles", a neologism used to describe the presence of numerous radio buttons that must be clicked to complete the task.

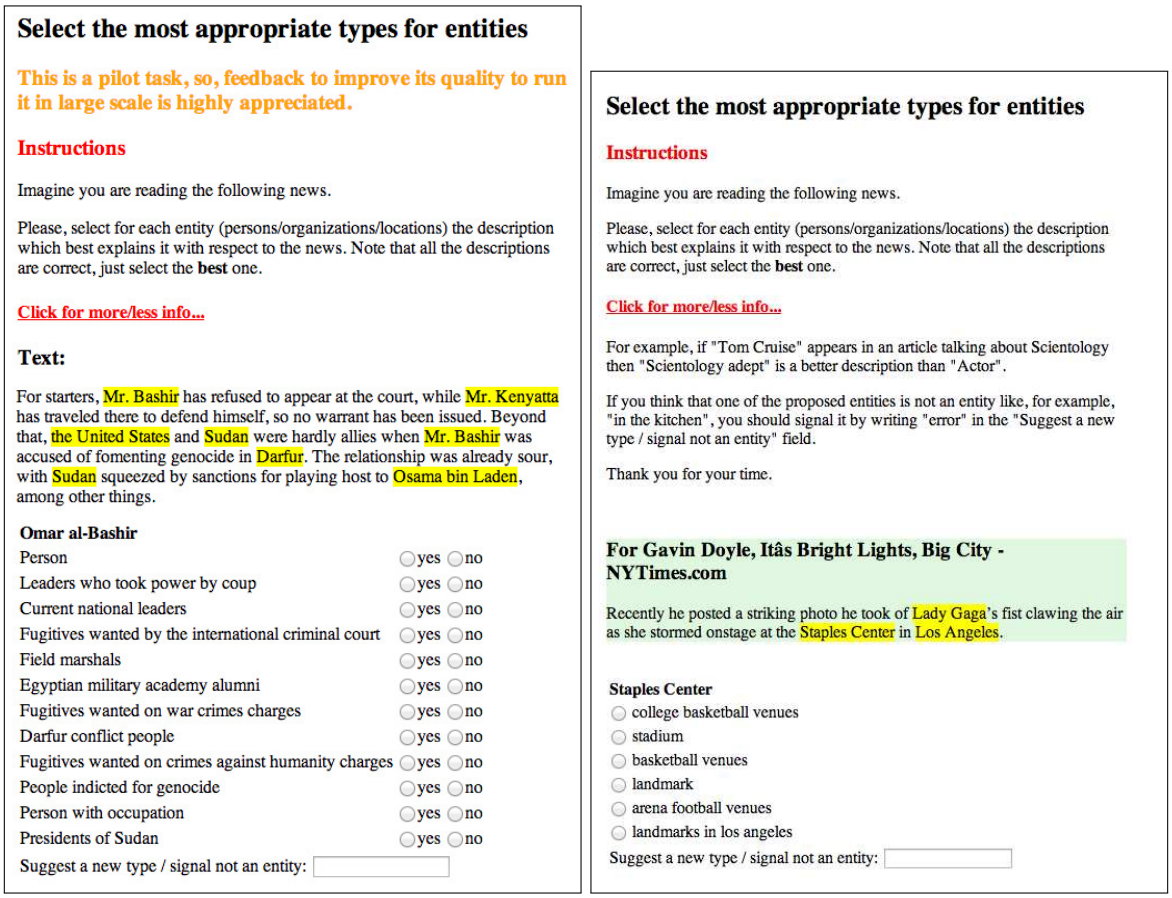

Figure 4: Task design (left) and final version of the interface used by the crowd to generate relevance judgements for entity types (right).

type for each entity displayed, but only one click was needed to complete the task.

Given the results of the pilot study, we selected our last task design to generate the relevance judgements in our datasets since it also simulates our target use case of showing one single entity type to a user browsing the Web. Table 1 lists the inter-rater agreement among workers in terms of Fleiss' k computed on the final relevance judgements for each test collection we made. We observe that agreement increases with an increasing degree of context. We argue that the context helps the worker in selecting the right type for the entity; without much context, the worker is prone to subjective interpretation of the entity, that is, he/she associates to the entity the type that is the most relevant *based on his/her background knowledge*.

Table 1: Agreement rate among crowd assessors during the evaluation of the four test collections.

Collection	Fleiss' κ
Entity-only	0.3662
Sentence	0.4202
Paragraph	0.4002
3-Paragraphs	0.4603

8. Experiments

8.1. Experimental Setting

We have 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 website during the period 21 Feb – 7 Mar 2013. On average, each article contains 12 entities. After the entity linking step, each entity is associated to an average of 10.2 types from in our Linked

Data collection. We crowdsourced the selection of the most relevant types by asking workers which types are the most relevant given a specific textual context. To generate our ground truth out of the crowdsourcing results we consider relevant each type which has been selected by at least one worker to obtain binary judgements, and we consider the number of workers who selected the type as its relevance score in a graded relevance setting.

Evaluation Measures. As main evaluation measure 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

$$AP(T_e) = \frac{\sum_{t_i \in T_e} rel(t_i) \cdot P@i}{|Rel(T_e)|} \quad (4)$$

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 considers 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 Normalize Discounted Cumulative Gain (NDCG) [42] 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, *discounted cumulative gain* measures the overall gain obtained by reaching rank k putting more weight at the top of the ranking:

$$DCG[k] = \sum_{j=1}^{k} G[j]/(log_2(1+j)) \quad (5)$$

To obtain NDCG, we normalize it dividing DCG by its optimal value obtained with the optimal gain vector which puts the most relevant results first. This gives a measure in $[0, 1]$ where 1 is obtained with the best possible ranking.

During the evaluation of our methods we only consider entity-types belonging to the integrated TRank^{++} hierarchy.

8.2. Dataset Analysis

Out of the NYT articles we have crawled, we created four different datasets to evaluate and compare 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 collections is composed by 770 distinct entities: out of the original 990 extracted entities we consider only those with at least two types to be ranked and we removed the errors in NER and entity linking which were identified by the crowd during the relevance judgements. Each entity has, on average, 10.2 types to be ranked.

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 in the given context (e.g., a sentence). Thus, the assessor has to read the context and select the type that best describes the entity given the presented text. 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 paragraph 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 paragraph in the news article. As the paragraph collection, 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 mid paragraph.

We measured the similarity among relevance judgements run displaying different entity contexts to the human judges. Specifically, to compare the judgements over two contexts A and B we measure

$$JudgOverlap(A, B) = \frac{|sharedRelevant(A, B)|}{min(|relevant(A)|, |relevant(B)|)} \quad (6)$$

Table 2 shows the similarity scores among judgements. We can observe that showing exclusively the

Table 2: Overlap of type relevance judgements run using different entity contexts.

Context A	Context B	JudgOverlap
Sentence	Entity only	0.7644
Paragraph	Entity only	0.7689
3-Paragraphs	Entity only	0.7815
Sentence	Paragraph	0.8501
Sentence	3-Paragraphs	0.8616
Paragraph	3-Paragraphs	0.8328

entity to the human assessor yields to somehow different judgements as compared to displaying some contextual information about the entity. No major differences can be observed among different sizes of contexts.

8.3. TRank^{++} Effectiveness Results

Evaluation over different Contexts. Figure 5 shows the evolution of MAP and NDCG values by varying the number of types associated to an entity. We can see that when an entity has many different types it is more difficult to rank types for it. Even for the simple approach FREQ when few types are assigned to an entity we can obtain effective results. On the right side of Figure 5 we can see the robustness of DEC-TREE over an increasing number of types associated to the entity.

Figure 6 shows the average effectiveness values for entities in different sections of our NYT collection. We can see that the most effective type ranking can be obtained on articles from the Dealbook section which deals with merge, acquisitions, and venture capital topics. Most challenging categories for finding the correct entity type include Arts and Style. On the right side of Figure 6 we can see the number of types associated to entities appearing in a specific section of the NYT. We can see that the entities with most types (16.9) appear in the Opinion section. The less types (8.7) are associated to entities appearing in the Dealbook section which may also explain the fact that their types can be ranked best.

Table 3 reports the results of the evaluation of all the methods described in Section6. We notice that, when we compare the results obtained among the different collections (i.e., entity-only, sentences, paragraph, and 3 paragraphs), the 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 WIKILINK-OUT, that is the approach that follows the `<dpo:wikiLink>` edges starting from the entity e we are ranking types for and checks the frequency of its types among the connected entities. Among the context-aware approaches the NGRAMS1 method performs best while, as expected, NGRAMS0 performs poorly, obtaining scores which are similar to those of our baseline. Despite being conceptually simple, the hierarchy-based approaches clearly outperform most of the other methods, showing scores similar to the most sophisticated context-aware approach. Even the simple DEPTH approach performs effectively.

To evaluate NGRAMS1 and KB-HIER, we used 5-fold cross validation over 4'334, 6'011, 5'616, and 5'620 data points in the four different collections. We increased the number of splits when evaluating the combination of all the approaches since we had more data points: we ran 10-fold cross validation over 7'884, 11'875, 11'279, and 11'240 data points. For this last approach, out of the ranking approaches we have proposed, we selected 12 features which cover the different methodologies (i.e., entity-centric, context-aware, and hierarchy-based) to train regression models for entity type ranking. We can observe that the best performing method is the one based on decision trees (DEC-TREE) which outperforms all other approaches. KB-HIER, which was initially design to investigate how hierarchy-based features interplay with features extracted from the knowledge base, performs actually better than our linear regression approach (LIN-REG) that combines scores coming from the other approaches.

Crowd-powered Entity Type Assignment. For some entities the knowledge base may not contain good enough types. For example, some entities have only `<owl:Thing>` and `<rdfs:Resource>` attached to them. In such cases, we ask the crowd to suggest a type for the entity they are judging. While it is not the focus of this paper to extend existing LOD ontologies with additional schema, we claim that this can be easily done by means of crowdsourcing. Some example of crowd-originated entity types are listed in Table 4.

Table 3: Type ranking effectiveness in terms of NDCG and MAP for different textual contexts. Statistically significant improvements (t-test $p < 0.05$) of the mixed approaches over the best ranking approach are marked with *.

Approach	Entity-only		Sentence		Paragraph		3-Paragraphs	
	NDCG	MAP	NDCG	MAP	NDCG	MAP	NDCG	MAP
FREQ	0.5997	0.4483	0.5546	0.3898	0.5503	0.3932	0.5163	0.3682
WIKILINK-IN	0.6390	0.5110	0.5658	0.4194	0.5759	0.4361	0.5520	0.4054
WIKILINK-OUT	0.6483	0.5189	0.5795	0.4509	0.5864	0.4579	0.5668	0.4294
SAME-AS	0.6451	0.5088	0.5731	0.4192	0.5823	0.4243	0.5583	0.4046
LABEL	0.6185	0.4777	0.5693	0.4173	0.5555	0.4090	0.5344	0.3875
SAMETYPE	-	-	0.5964	0.4465	0.5938	0.4385	0.5583	0.4083
PATH	-	-	0.5959	0.4654	0.5966	0.4642	0.5609	0.4290
NGRAMS0	-	-	0.5559	0.3865	0.5449	0.3855	0.5144	0.3625
NGRAMS1	-	-	0.6401	0.5255	**0.6608**	0.5526	**0.6413**	**0.5431**
DEPTH	0.7016	0.5994	0.6210	0.5082	0.6279	0.5171	0.6117	0.4984
ANCESTORS	0.7058	0.6041	**0.6484**	**0.5434**	0.6510	**0.5544**	0.6335	0.5322
ANC_DEPTH	**0.7138**	**0.6186**	0.6352	0.5269	0.6420	0.5370	0.6211	0.5149
KB-HIER	0.7179	0.6167	0.6954*	0.5966*	0.7050*	0.6233*	0.6759*	0.5885*
DEC-TREE	**0.7248**	**0.6224**	**0.7282***	**0.6535***	**0.7297***	**0.6618***	**0.7003***	**0.6279***
LIN-REG	0.6930	0.5847	0.6337	0.5304	0.6488	0.5465	0.6202	0.5100

8.4. TRank^{++} Scalability

We run the MapReduce TRank^{++} pipeline over a sample of CommonCrawl[15] which contains `schema.org` annotations. At the moment of writing this paper, CommonCrawl is formed by 177 valid crawling segments, accounting for 71TB of compressed Web content. We sampled uniformly 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 formed by 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 indexes ($\sim$ 600MB) used by the TRank^{++} pipeline allowed us to replicate the indexes 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–mandatory requirements for any architecture which could be considered Web-scale ready. We argue that the good performance of our MapReduce pipeline is majorly due to the use of small, pre-computed inverted indexes instead of expensive SPARQL queries.

Processing the corpus on such testbed takes 25 minutes on average, that is to say each server runs the whole *TRank^{++}* pipeline on 72 documents per second. Table 5 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 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 [43] (see Table 6).[16]

Table 7 shows the most frequent entity types selected by TRank^{++} for entities contained in Web pages annotated with the top `schema.org` classes. We can observe how TRank^{++} types refer to specific entities mentioned in topic-specific pages as for example, `<yago:InternetCompaniesOfTheUnitedStates>` entities are contained in `http://schema.org/Product` Web pages.

Table 8 shows the entity types that most frequently co-occur in our sample of CommonCrawl. Most frequent entity types mentioned together are about actors.

Figure 7 shows the diversity of entity types selected by TRank^{++} for Web pages annotated with

[15] `http://commoncrawl.org/`

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

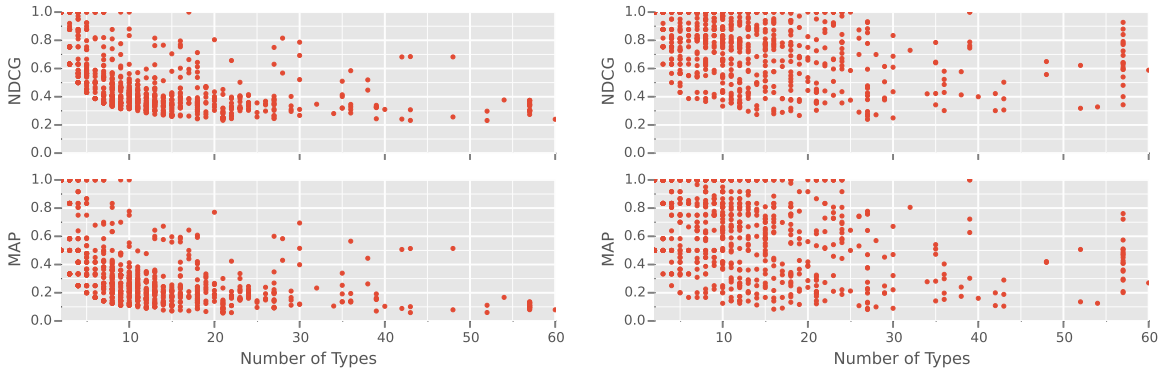

Figure 5: MAP and NDCG of FREQ (top) and DEC-TREE (bottom) for entities with different numbers of types on the 3-paragraphs collection.

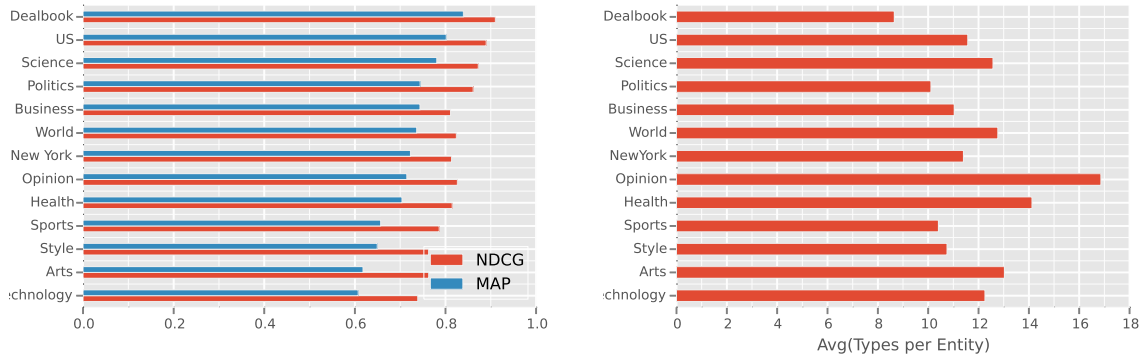

Figure 6: Distribution of NDCG and MAP scores over article categories (top) and average number of types per entity over article categories (bottom).

different `schema.org` classes. We can clearly see power low distribution where the top schema.org classes contain very many different entity types while most of the other have low diversity of entity types.

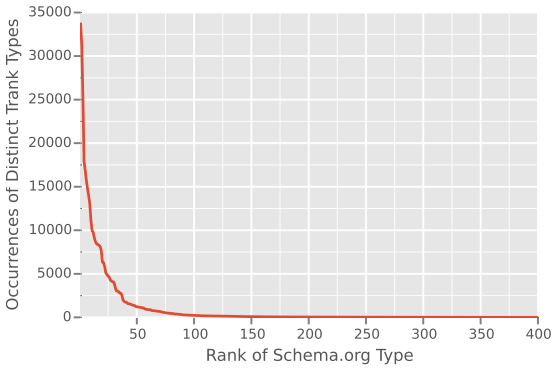

Figure 7: Occurrences of distinct TRank^{++} types in CommonCrawl (log scale).

9. Discussion

In this section, we comment on the performance of the various approaches we empirically evaluated. We focus particularly on the new text-based approaches and on KB-HIER.

We begin by pointing the reader's attention to the bottom part of Figure 8, as it illustrates where the relevant types are located in our type hierarchy. As can be observed, most of the relevant types can be found between the second and the fifth level of our tree, with a remarkable peak at Type Depth = 4. We also notice that the greatest number of

Table 4: Examples of crowd-originated entity types.

Entity Label	Existing Types	Crowd Suggested Type
David Glassberg	Alumnus, Resource, Northwestern Uni. Alumni, US 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 5: Performance breakdown of the MapReduce pipeline.

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

leaves can be found at that depth level. This provides a hint as of why DEPTH performs worse than other methods: as many of the relevant types have a depth of four, returning deeper results is typically not optimal.

The upper part of the figure shows how some of the features taken into consideration by NGRAMS1 behave. We can see that both `nSiblings` and `hSiblings` are able to detect the peak of relevant results at Type Depth = 4 . These results suggest that `hSiblings` and `NGRAMS0` are good estimators of the relevance of entity types.

As for the `hChildren` feature, the experimental results depicted in Figure 9 confirm that relevance is related to entropy. As can be seen, nodes whose children's entropy is lower have a lower relevance score. Despite the interesting properties of the metrics just described, the most predictive features for NGRAMS1 are, in order, `NGRAMS0` (41%), `ratioParent` (15%), `hSiblings` (14%), `nSiblings` (11%), `hChildren` (10%).[17] Figure 10 shows how `NGRAMS0`, `ratioParent` and relevance relate to each other and explains why the NGRAMS0 approach

[17]The reported numbers are computed on the Sentences Collection; a similar trend was observed in the other datasets.

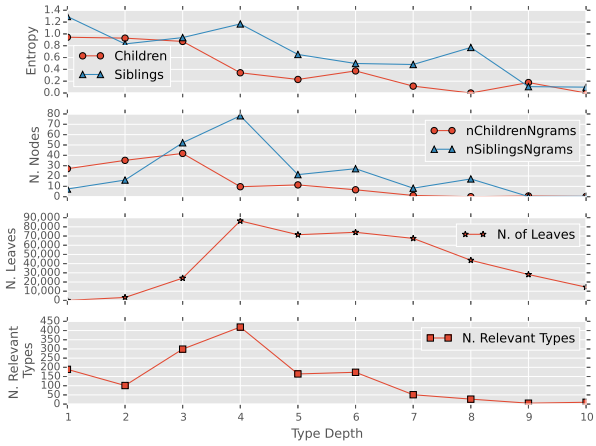

Figure 8: Distribution of NGRAMS1 features, number of leaves in the type hierarchy, and number of relevant results with varying a type depth.

performs poorly: it can easily detect the really few types which have the highest level of relevance but fails in detecting irrelevant types, which are by far more numerous. It is also interesting to notice that, on average, the n-gram probability mass is scattered among very few nodes at Type Depth = 5, . . . , 7, where most of the leaves are. This can be due partly by the fact that the extension of the text window we used is not big enough to catch the difference among deeper types, thus not assigning any probability to types of those level, and partly to the fact that in our testset there are fewer occurrences of types having depth greater than 5. Finally, from Figure 8 we postulate that the learner possibly favored `hSiblings` over `hChildren` because of its abil-

Table 6: CommonCrawl sample statistics.

Domain	% in Corpus	Schema.org Type	% in Corpus
youtube.com	39.65	VideoObject	40.79
blogspot.com	9.26	Product	32.66
over-blog.com	0.67	Offer	28.92
rhapsody.com	0.54	Person	20.95
fotolog.com	0.52	BlogPosting	18.97

Table 7: Co-occurrences of Schema.org annotations with entity types.

Schema.org Type	top-3 most frequent TRank^{++} types
VideoObject	dp:GivenName dp:Settlement dp:Company
Product	yago:InternetCompaniesOfTheUnitedStates yago:PriceComparisonServices dp:Settlement
Offer	yago:InternetCompaniesOfTheUnitedStates yago:PriceComparisonServices dp:Company
Person	dp:GivenName dp:Company yago:FemalePornographicFilmActors
BlogPosting	dp:GivenName dp:Settlement yago:StatesOfTheUnitedStates

ity to detect the peak of relevant results occurring at Type Depth = 4.

As for KB-HIER, surprisingly the most predictive feature selected by the learner is `nTypes` (0.38%), followed by `popularity` (20%), `nSiblings` (17%), `nChildren` (14%), and `typeDepth` (11%).[18] We studied the relation between `nTypes`, `nSiblings`, and relevance. The results of our analysis are shown in Figures 11 and 12. From the former figure we observe that, in general, with small numbers of types to rank, those with fewer siblings are more likely to be relevant, while as the number of types to be ranked increases, types with fewer siblings get more relevant. The latter figure suggests that popular entities have more types, thus suggesting a connection between `nTypes` and `popularity`.

We conclude this section with a final remark about the effectiveness of context-aware and context-unaware methods. We noticed that, in general, preferring deeper types has often a positive effect on the final ranking of the types of a given entity. This is highlighted by the high scores achieved by all hierarchy based methods and by NGRAMS. However, there are cases for which such a choice does not pay off and for which context-based methods perform better. For instance, in document P3-0146, "Mali" co-occurs with "Paris", "Greece", and "Europe". The top-3 results selected by ANCESTORS (context-unaware) are "LeastDevelopedCountries", "LandlockedCountries", and "FrenchspeakingCountries" but they are all marked as non-relevant by the Crowd since they were deemed too specific. In contrast, the top-3 types selected by PATH (context-aware), namely, "PopulatedPlace", "Place", and "Country", are all relevant. In this case, ANCESTORS obtained a low score since it favored the most specific types, while PATH obtained a higher score since it exploited the types of

[18]The reported numbers are computed on the Entity-only collection; a similar trend was observed in the other datasets.

Table 8: Co-occurrences of entity types in the CommonCrawl sample.

Type	Type	%
yago:Actor109765278	yago:Actor109765278	0.193
dp:GivenName	dp:GivenName	0.077
dp:Settlement	dp:Settlement	0.072
yago:Actor109765278	yago:AmericanStageActors	0.064
dp:Person	yago:Actor109765278	0.061
dp:GivenName	dp:Settlement	0.040
yago:EnglishTelevisionActors	yago:Actor109765278	0.039
dp:GivenName	yago:FirstName106337307	0.038
yago:StatesOfTheUnitedStates	yago:StatesOfTheUnitedStates	0.035
yago:AmericanStageActors	yago:AmericanStageActors	0.030

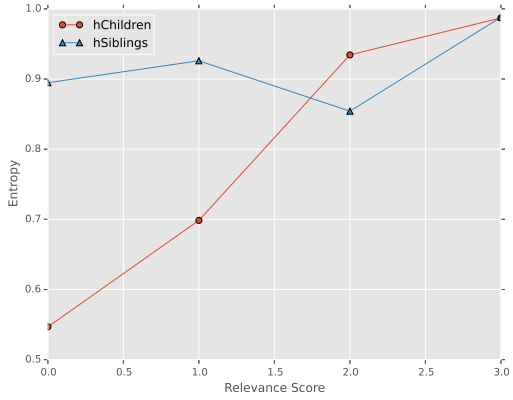

Figure 9: Relation between `hChildren`, `hSiblings`, and relevance. As can be observed, there is an almost linear relation between `hChildren` and relevance.

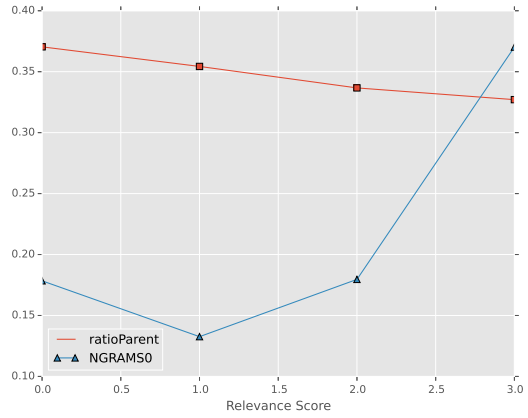

Figure 10: Relations between the two most predictive features of NGRAMS1 and relevance.

the other entities to favor coarser types. We believe that this justifies the adoption of a mixed approach to rank entity types and partly explains the high effectiveness achieved by the DEC-TREE approach.

10. Conclusions

In this paper, we focused on the task of ranking a set of types associated to an entity in a background knowledge graph to select the most relevant types given the context in which the entity appears in text. Displaying such types can improve user engagement in online platforms as well as improve a number of advanced services such as ad-hoc object retrieval, text summarization or knowledge capture.

We proposed different classes of ranking approaches and evaluated their effectiveness using crowdsourced relevance judgments. We have also evaluated the efficiency of our approaches by using inverted indices for fast access to entity and type hierarchy information and a Map/Reduce pipeline for entity type ranking over a Web crawl. Our experimental evaluation shows that features extracted from the type hierarchy and from the textual context surrounding the entities perform well for ranking entity types in practice, and that methods based on such features outperforms other approaches. A regression model learned over training data that combines the different classes of approaches significantly improves over the individual ranking functions reaching a NDCG value of 0.73. As future work, we aim at evaluating the user impact of the proposed techniques by running a large scale-experiment through the deployment of a browser plugin for contextual entity type display. An additional dimension we plan to investigate in the future

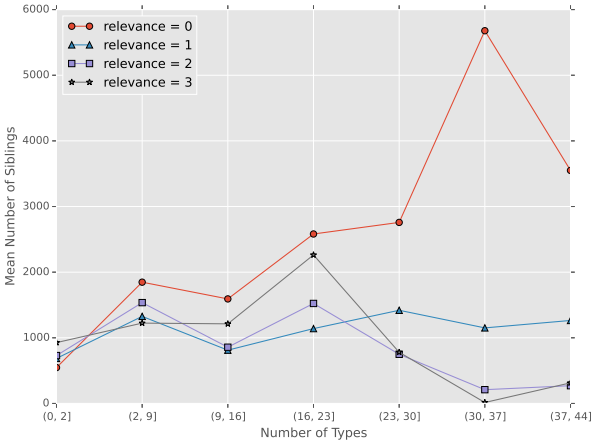

Figure 11: Interplay among `nTypes`, `nSiblings` (two of the most predictive features of KB-HIER), and relevance.

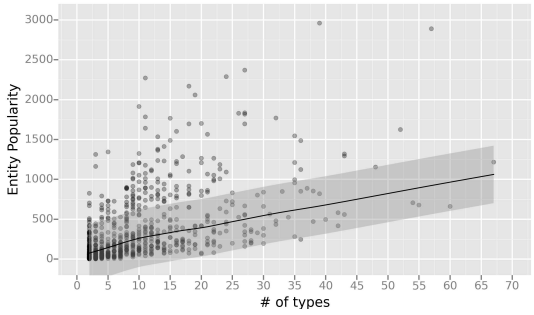

Figure 12: Relation between the popularity of entities, computed on Sindice2011 and their number of types.

is the creation and display of supporting evidence for the selection of the entity types. For example, a sentence or related entities may be presented to motivate the fact that the selected type is not the most popular type related to the entity (e.g., Tom Cruise is a vegetarian according to Freebase).

References

[1] R. Kumar, A. Tomkins, A characterization of online search behavior, IEEE Data Eng. Bull.

[2] J. Pound, P. Mika, H. Zaragoza, Ad-hoc object retrieval in the web of data, in: WWW, ACM, New York, NY, USA, 2010, pp. 771–780.

[3] A. Tonon, G. Demartini, P. Cudré-Mauroux, Combining inverted indices and structured search for ad-hoc object retrieval, in: Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval, SIGIR '12, ACM, New York, NY, USA, 2012, pp. 125–134.

[4] F. M. Suchanek, G. Kasneci, G. Weikum, Yago: a core of semantic knowledge, in: Proceedings of the 16th international conference on World Wide Web, WWW '07, ACM, New York, NY, USA, 2007, pp. 697–706.

[5] A. Tonon, M. Catasta, G. Demartini, P. Cudré-Mauroux, K. Aberer, TRank: Ranking entity types using the web of data, in: Lecture Notes in Computer Science, Vol. 8218 LNCS, 2013, pp. 640–656.

[6] M. Ciaramita, Y. Altun, Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger, in: Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, EMNLP '06, Association for Computational Linguistics, Stroudsburg, PA, USA, 2006, pp. 594–602.

[7] H. Cunningham, K. Humphreys, R. Gaizauskas, Y. Wilks, GATE: a general architecture for text engineering, in: Proceedings of the fifth conference on Applied natural language processing: Descriptions of system demonstrations and videos, ANLC '97, Association for Computational Linguistics, Stroudsburg, PA, USA, 1997, pp. 29–30.

[8] K. Balog, Y. Fang, M. de Rijke, P. Serdyukov, L. Si, Expertise retrieval, Foundations and Trends in Information Retrieval 6 (2-3) (2012) 127–256.

[9] H. Zaragoza, H. Rode, P. Mika, J. Atserias, M. Ciaramita, G. Attardi, Ranking very many typed entities on wikipedia, in: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, CIKM '07, ACM, New York, NY, USA, 2007, pp. 1015–1018.

[10] G. Demartini, C. Firan, T. Iofciu, R. Krestel, W. Nejdl, Why finding entities in Wikipedia is difficult, sometimes, Information Retrieval 13 (5) (2010) 534–567. doi:10.1007/s10791-010-9135-7.

[11] 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: Proc. of TREC, 2010.

[12] K. Balog, D. Carmel, A. P. de Vries, D. M. Herzig, P. Mika, H. Roitman, R. Schenkel, P. Serdyukov, D. T. Tran, The first joint international workshop on entity-oriented and semantic search (jiwes), SIGIR Forum 46 (2) (2012) 87–94.

[13] K. Balog, A. P. de Vries, P. Serdyukov, J.-R. Wen, The first international workshop on entity-oriented search (eos), SIGIR Forum 45 (2) (2011) 43–50.

[14] M. Bron, K. Balog, M. de Rijke, Example based entity search in the web of data, in: ECIR, 2013, pp. 392–403.

[15] G. Tummarello, R. Cyganiak, M. Catasta, S. Danielczyk, R. Delbru, S. Decker, Sig.ma: Live views on the Web of Data, Web Semantics: Science, Services and Agents on the World Wide Web 8 (4) (2010) 355–364.

[16] J. R. Finkel, C. D. Manning, Joint parsing and named entity recognition, in: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, NAACL '09, Association for Computational Linguistics, Stroudsburg, PA, USA, 2009, pp. 326–334.

[17] D. Nadeau, P. D. Turney, S. Matwin, Unsupervised named-entity recognition: generating gazetteers and resolving ambiguity, in: Proceedings of the 19th international conference on Advances in Artificial Intelligence: Canadian Society for Computational Studies of Intelligence, AI'06, Springer-Verlag, Berlin, Heidelberg, 2006, pp. 266–277.

[18] D. Nadeau, Semi-supervised named entity recognition:

learning to recognize 100 entity types with little supervision, Ph.D. thesis, Ottawa, Ont., Canada, Canada, aAINR49385 (2007).

[19] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, S. Hellmann, DBpedia - A crystallization point for the Web of Data, Web Semant. 7 (3) (2009) 154–165.

[20] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, J. Taylor, Freebase: a collaboratively created graph database for structuring human knowledge, in: Proceedings of the 2008 ACM SIGMOD international conference on Management of data, SIGMOD '08, ACM, New York, NY, USA, 2008, pp. 1247–1250.

[21] D. Vallet, H. Zaragoza, Inferring the most important types of a query: a semantic approach, in: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '08, ACM, New York, NY, USA, 2008, pp. 857–858.

[22] K. Balog, R. Neumayer, Hierarchical target type identification for entity-oriented queries, in: CIKM, 2012, pp. 2391–2394.

[23] T. Tylenda, M. Sozio, G. Weikum, Einstein: physicist or vegetarian? summarizing semantic type graphs for knowledge discovery, in: Proceedings of the 20th international conference companion on World wide web, WWW '11, ACM, New York, NY, USA, 2011, pp. 273–276.

[24] A. Gangemi, A. G. Nuzzolese, V. Presutti, F. Draicchio, A. Musetti, P. Ciancarini, Automatic typing of dbpedia entities, in: International Semantic Web Conference (1), 2012, pp. 65–81.

[25] N. Nakashole, T. Tylenda, G. Weikum, Fine-grained semantic typing of emerging entities, in: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Sofia, Bulgaria, 2013, pp. 1488–1497.

[26] L. Yao, S. Riedel, A. McCallum, Universal schema for entity type prediction, in: Proceedings of the 2013 Workshop on Automated Knowledge Base Construction, AKBC '13, ACM, New York, NY, USA, 2013, pp. 79–84.

[27] A. Kalyanpur, J. W. Murdock, J. Fan, C. Welty, Leveraging community-built knowledge for type coercion in question answering, in: Proceedings of the 10th international conference on The semantic web - Volume Part II, ISWC'11, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 144–156.

[28] C. Welty, J. W. Murdock, A. Kalyanpur, J. Fan, A comparison of hard filters and soft evidence for answer typing in watson, in: International Semantic Web Conference (2), 2012, pp. 243–256.

[29] S. E. Whang, H. Garcia-Molina, Joint Entity Resolution on Multiple Datasets, The VLDB Journal.

[30] G. Demartini, D. E. Difallah, P. Cudré-Mauroux, ZenCrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking, in: Proceedings of the 21st international conference on World Wide Web, WWW '12, ACM, New York, NY, USA, 2012, pp. 469–478.

[31] J. R. Finkel, T. Grenager, C. Manning, Incorporating non-local information into information extraction systems by gibbs sampling, in: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, ACL '05, Association for Computational Linguistics, Stroudsburg, PA, USA, 2005, pp. 363–370.

[32] C. Matuszek, J. Cabral, M. Witbrock, J. DeOliveira, An introduction to the syntax and content of cyc, in: Proceedings of the 2006 AAAI spring symposium on formalizing and compiling background knowledge and its applications to knowledge representation and question answering, Vol. 3864, Citeseer, 2006.

[33] F. M. Suchanek, S. Abiteboul, P. Senellart, Paris: Probabilistic alignment of relations, instances, and schema, Proceedings of the VLDB Endowment 5 (3) (2011) 157–168.

[34] S. Campinas, D. Ceccarelli, T. E. Perry, R. Delbru, K. Balog, G. Tummarello, The Sindice-2011 dataset for entity-oriented search in the web of data, in: 1st International Workshop on Entity-Oriented Search (EOS), 2011, pp. 26–32.

[35] P. N. Mendes, M. Jakob, A. García-Silva, C. Bizer, Dbpedia spotlight: Shedding light on the web of documents, in: Proceedings of the 7th International Conference on Semantic Systems, I-Semantics '11, ACM, New York, NY, USA, 2011, pp. 1–8.

[36] T. Brants, A. C. Popat, P. Xu, F. J. Och, J. Dean, G. Inc, Large language models in machine translation, in: Conference on Empirical Methods in Natural Language Processing (EMNP), 2007, pp. 858–867.

[37] A. Tonon, M. Catasta, G. Demartini, P. Cudré-Mauroux, Fixing Domain and Range of Properties in Linked Data by Context Disambiguation (2015).

[38] P. Geurts, D. Ernst, L. Wehenkel, Extremely randomized trees, Machine Learning 63 (1) (2006) 3–42. doi:10.1007/s10994-006-6226-1.

[39] T.-Y. Liu, Learning to rank for information retrieval, Found. Trends Inf. Retr. 3 (3) (2009) 225–331.

[40] G. Holmes, M. Hall, E. Frank, Generating rule sets from model trees, in: Proceedings of the 12th Australian Joint Conference on Artificial Intelligence: Advanced Topics in Artificial Intelligence, AI '99, Springer-Verlag, London, UK, UK, 1999, pp. 1–12.

[41] J. R. Quinlan, Learning with continuous classes, in: Proceedings of the 5th Australian joint Conference on Artificial Intelligence, Vol. 92, Singapore, 1992, pp. 343–348.

[42] K. Järvelin, J. Kekäläinen, Cumulated gain-based evaluation of ir techniques, ACM Trans. Inf. Syst. 20 (4) (2002) 422–446.

[43] H. Mühleisen, C. Bizer, Web data commons - extracting structured data from two large web corpora, in: LDOW, 2012.