International Journal of Semantic Computing

Enriching Textual Search Results at Query Time using Entity Mining, Linked Data and Link Analysis

PAVLOS FAFALIOS, PANAGIOTIS PAPADAKOS and YANNIS TZITZIKAS
Institute of Computer Science, Foundation for Research and Technology - Hellas, and Computer Science Department, University of Crete, GREECE
{*fafalios,papadako,tzitzik*}*@ics.forth.gr*

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

The integration of the classical Web (of documents) with the emerging Web of Data is a challenging vision. In this paper we focus on an integration approach during *searching* which aims at enriching the responses of non-semantic search systems with semantic information, i.e. Linked Open Data (LOD), and exploiting the outcome for offering advanced exploratory search services which provide an overview of the search space and allow the users to explore the related LOD. We use *named entities* identified in the search results for automatically connecting search hits with LOD and we consider a scenario where this entity-based integration is performed at query time with no human effort and no a-priori indexing which is beneficial in terms of configurability and freshness. However, the number of identified entities can be high and the same is true for the semantic information about these entities that can be fetched from the available LOD. To this end, in this paper we propose a Link Analysis-based method which is used for ranking (and thus selecting to show) the more important semantic information related to the search results. We report the results of a survey regarding the marine domain with promising results, and comparative results that illustrate the effectiveness of the proposed (PageRank-based) ranking scheme. Finally, we report experimental results regarding efficiency showing that the proposed functionality can be offered even at query time.

Keywords: exploratory search, entity mining, linked data, link analysis, semantic search

1. Introduction

The Web has evolved from an information space of interconnected web pages to one where both unstructured documents and structured data in various forms co-exist. An important question is how typical web users, who mainly use keywords in searching, can access and exploit this increasing body of knowledge. In addition, most search methods are appropriate for *focalized search*, i.e. they make the assumption that users can accurately describe their information need using a small sequence of words and that they are interested only in the top hits. However, a high percentage of search tasks are *exploratory* and focalized search very commonly leads to inadequate interactions and poor results [1]. Our objective is to enable effective exploratory search services which can bridge the gap between the responses

of non semantic search systems (e.g., professional search systems, web search engines) and semantic information, i.e. Linked Open Data (LOD) [2]. An important observation is that *entity names* (e.g., persons, locations, organizations, etc.) occur in all kinds of artifacts: documents, database cells, RDF triples, etc. Therefore, a basic hypothesis that we investigate is whether and how we can exploit named entities for offering a kind of *entity-based integration* method; the *named entities* are used as the "glue" for automatically connecting documents (i.e. search results) with data and knowledge. For being configurable and for tackling the constant evolution of published LOD, we investigate a scenario where these services are provided as meta-services and the entity-based integration is performed at *query time*, with no human effort. This makes the LOD accessible to the end users and allows offering this functionality on top of existing search systems.

Consider the following scenario from the *marine* domain[a]: a biologist seeks information about *marine species* and submits to a professional search system a query requesting information about fish species of a particular genus. Past systems that provide a kind of semantic enrichment of search results (like [3, 4]) present to the user only the detected entities (e.g., several species identified in the search results) allowing the user to narrow the search space to a set of results that contain a particular entity. Figure 1 (left) shows an indicative screen shot of the `X-ENS` system [4]. However, the structured knowledge that is available for these entities is not exploited. For instance, an identified species (e.g., the *yellowfin tuna*) may have many properties (e.g., *family*, *genus*, *kingdom*, etc.) and related entities (e.g., *predators*, *binomial authority*, etc.), and can belong to multiple categories (e.g., *Fish*, *Eukaryote*, *Fish of Hawaii*, etc.). Furthermore, some species may share one or more common properties or related entities (e.g., two species belong to the same *genus* or *family*). All this information should be exploitable as it can provide useful information about the *context* of these entities. In addition, it allows the user to instantly inspect information that may exist in different places and that may be laborious and time consuming to locate, e.g., how the detected species *papuan seerfish* and *kanadi kingfish* are related, why the species *pacific bonito* was detected in the search results for the query *tuna*, etc. All this information can be integrated in the search process helping the user, apart from restricting the search space, to get a more sophisticated overview and to make better sense of the results.

However, the number of identified entities can be high and the amount of structured information that is available for these entities can be very high too, i.e. their associations and properties[b]. Therefore, there is a need for methods that *rank* all this semantic information in order to promote and present to the end-users the most important entities, associations and properties.

To tackle the above challenges, in this paper we propose a method founded on Link Analysis. Specifically, we introduce an appropriately biased PageRank-like al-

[a]Which is a real scenario related to the iMarine project (`http://www.i-marine.eu/`).
[b]We call *association* a relationship between two entities and *property* an attribute of an entity.

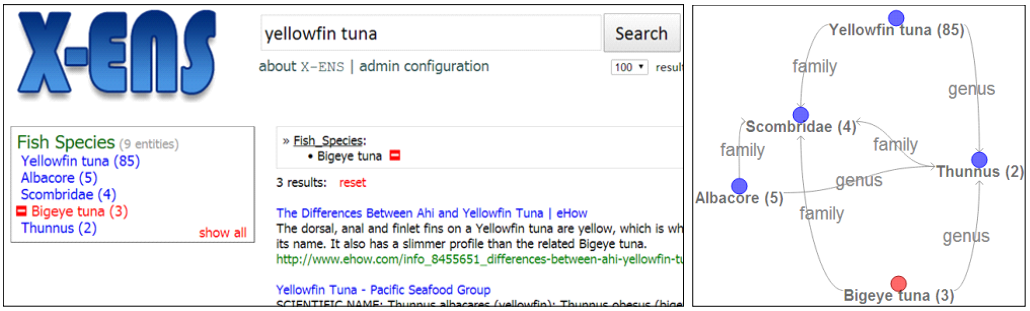


Fig. 1: An indicative screen shot of the `X-ENS` system (left) and the top-5 semantic graph that correspond to the top-5 detected entities (right).

gorithm for ranking entities and properties, which is also exploited for producing and showing to the user *top-K semantic graphs*. A top-K semantic graph can complement the query answer with useful information regarding the *connectivity* of the identified entities. The keypoint is that this approach can exploit associations and it is quite general and configurable. Moreover, it promotes the entities identified in the top ranked results as well as the semantic information that is linked with many important (i.e. highly ranked) entities. We report the results of a survey and of a comparative evaluation with other ranking methods that demonstrate the usefulness and the effectiveness of this approach. We also report experimental results that support the feasibility of this approach. For example and regarding the marine domain, by analyzing the snippets of the top-100 results that Bing web search engine returns for the query *yellowfin tuna* (with *fish species* as the entities of interest and exploiting DBpedia [5] at real-time), in the top semantic graphs we get information about the taxonomy of the *yellowfin tuna* (family, genus, etc.), other tuna species that belong to the same family or the same conservation status system (e.g., the *bigeye tuna*), how all these entities are connected, etc. We get all this information in only 3 seconds without performing any additional query. Figure 1 (right) depicts an example of a top-5 semantic graph.[c]

In a nutshell, the key contributions are the following:

- We introduce a search paradigm in which the search results are connected with data and knowledge at query time with no human effort (§2).
- We detail a biased PageRank algorithm for ranking entities and properties which can identify and promote the important semantic information (§4).
- We present a) the results of a survey regarding the marine domain which demonstrate the usefulness of the proposed approach (§5.1), b) the results of a comparative evaluation with other ranking methods that illustrate the effectiveness and quantify the difference of the proposed ranking scheme (§5.2 and §5.3), and c) experimental results that reveal the applicability

[c]A proof-of-concept prototype (configured for the marine domain) is available at `http://139.91.183.72/x-ens-2/`

and the efficiency of the proposed approach (§5.4). We also discuss how we achieve scalability (§5.5).

The rest of this paper is organized as follows: §2 describes the context of the proposed approach, §3 discusses related works, §4 describes in detail the link analysis-based approach, §5 presents the evaluation results, while §6 concludes and identifies directions for future research.

2. Context

At first we define (quite informally) the notion of *"entities of interest"* and the task *"semantic post-processing of textual search results"*.

Definition 1. *Entities of Interest (EoI) are categories of entities like persons, locations, organizations, etc. that are meaningful in the application context. For instance, the EoI of a marine-related search system may be fish species, water areas, countries, etc., while for a medical search system the EoI may include drugs, diseases and proteins. These entities are provided by the community that uses the underlying application.*

Definition 2. *With the term "semantic post-processing of textual search results" we refer to the task of (a) identifying EoI in the textual results of a search system, (b) retrieving semantic information about the identified entities by exploiting one or more Knowledge Bases (KBs), and (c) producing a semantic graph related to the search results where nodes correspond to entities and properties of entities, while edges correspond to associations among these entities and properties. In a nutshell, the input of this process is a set of ranked documents (or ranked document snippets) and the output is an RDF graph characterizing the documents according to a configuration, i.e. according to the EoI and the underlying KBs.*

This task can be described through the following process (depicted at Figure 2):

STEP 1: The user submits a keyword query to a search system (e.g., to a professional search system or to a web search engine).

STEP 2: The search system uses a component, we call it `SPP` from "Semantic Post Processor", which exploits a named entity recognition tool (in our prototype Gate Annie [6]) for identifying entities in the (top) search results (either in their textual snippets or in their full contents or in their metadata fields). For configuring the EoI (in a preprocessing step), we exploit the LOD as it is proposed in [7], i.e. we can define that the EoI are the names of entities returned by a given SPARQL query or those that belong to a particular RDF class (thereby each entity is accompanied by its URI).

STEP 3: `SPP` exploits the LOD for getting *more* information about the detected entities (their properties and related entities). For instance, by running SPARQL queries we can retrieve all the incoming and outgoing properties of each entity URI.

STEP 4: For the identification of the most *important* entities and properties, a PageRank-like ranking scheme is used (it is analyzed in detail in §4).

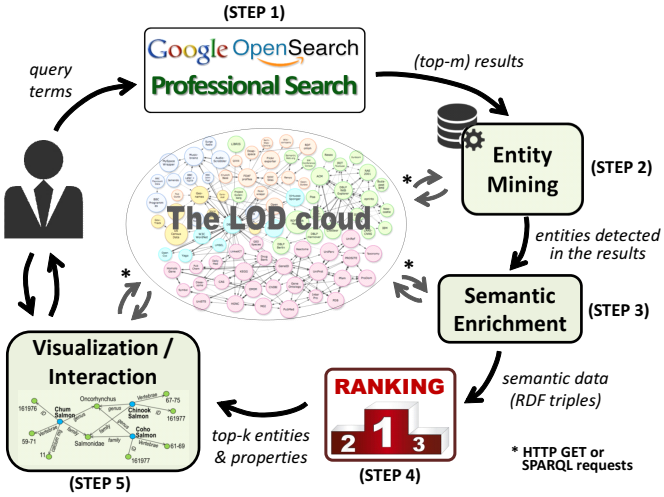


Fig. 2: Semantic post-processing of textual search results.

STEP 5: Apart from returning to the user the top-K entities and properties as derived by the PageRank-like algorithm, we can return a top-K semantic graph allowing the user to gradually increase or reduce the value of K. This is very important for showing how the entities are related. This functionality can be also offered on-demand as a complementary representation of the identified entities. Several user actions could be supported over this graph. For example, the user can inspect how two entities are associated, explore the properties of a particular entity, narrow the search space by selecting a detected entity or a characteristic (property) of the detected entities, etc. ⋄

We should stress that the above process is *fully configurable*. The user/admininstrator can easily configure the EoI and the KBs that are used for retrieving more information about the identified entities (accessible through SPARQL endpoints). Thereby, one can configure it for different domains. For example, for the *marine* domain, the useful categories include *fish species*, *countries*, *water areas*, etc., while for the *medical* domain *drugs*, *diseases*, *proteins*, etc. are interesting categories. As regards the underlying KBs, the LOD cloud contains numerous datasets covering many domains. For example, GeoNames [8] can be exploited for *geographic data*, DrugBank [9] for *drugs*, DBpedia [5] contains cross-domain data, etc. For reasons of homogeneity, all the examples in this paper concern the *marine* domain and we consider *fish species* from DBpedia as the EoI.

Note that the proposed approach does not mine/analyze the query string; it analyzes only the returned results. If the user submits a query that *semantically* is not related to the EoI, e.g. in a marine-related search system we submit the query *"everest mountain"* which is irrelevant to the application context, then the entity mining process may not identify entities (Step 2 of the process). Of course, in this case our approach does not produce a top-K graph. However, this is an important information which the user can take into account, e.g., for changing his/her query.

3. Related Work

Semantic post-processing of search results. [3, 4] presented a method to enrich the classical (keyword-based) web searching with entity mining that is performed at query time over the *snippets* of the search hits. The result of entity mining (entities grouped in categories) complements the query answer and can be further exploited in a faceted and session-based interaction scheme. In comparison to our work, this category of works does not exploit the structured knowledge that is available for the detected entities, i.e. their *properties* and the *associations* with other entities.

Keyword queries with entity-based markup. Works like [10, 11] propose frameworks for entity search in which users formulate queries that directly describe what types of entities they are looking for (using the prefix `#`, e.g. `#professor`). Again, the structured information about entities is not exploited.

Link Analysis for Entity Search. There are several works that exploit *link analysis-based* methods for *ranking* the results of an *entity search* process. [12] and [13] combine classical search techniques and *spread activation* techniques for ranking keyword search results, while [14] and [15] propose a *PageRank-like* method for ranking RDF resources and take into account the *data sources*. [16] adopts a modification of *Kleinberg's HITS algorithm* [17] for estimating the importance of RDF resources, [18] uses *PageRank* and *HITS* as features for ranking query-independent resources, while [19] applies link analysis methods based on a *rational surfer model* for ranking the importance of RDF documents. Finally, a most recent work [20] elaborates on *entity-relationship* queries and exploits the idea of *spread activation* for scoring the answers. However, the above works focus on retrieving and ranking resources from a semantic collection, and users get as output directly entity resources that match the query, not documents. Therefore, such works are quite distant from the way users search for information.

Exploiting semantic data in Web Search. *Google Knowledge Graph* (GKG) [21] evidences the increasing interest of exploiting semantic data in Web searching. GKG tries to understand the submitted query and presents a semantic description (in a right sidebar) of *only one* entity, the entity that the user is maybe looking for. However, for a bit more complex queries the user does not get any semantic information. For instance (and for the time being), for the query *"Barack Obama and Honolulu"*, GKG does not return any semantic information, although *Honolulu* is the birth place of *Barack Obama*, i.e. the two entities are highly related. Instead, our approach analyzes the search results and therefore can identify many entities, which is beneficial especially for recall-oriented information needs.

Synopsis. The approach that we propose does not change the (user-friendly) way users search for information, but acts as a *mediator* between any search system and semantic information (LOD); users still get documents as search results, but also get and interact with semantic information that is highly related to the results (providing a way of making the LOD accessible to the end-users). In addition, the derived *semantic graphs* show how the entities are connected and their context, and

deter the disengagement of the users from their initial task since users can instantly inspect semantic information that may be laborious and time consuming to detect.

4. A Link Analysis-based Approach

We focus on the problem of selecting and ranking the identified entities and their related structured information, i.e. on Steps 2, 3 and 4 of the process described in §2. The main idea is to construct *dynamically* an RDF graph about the identified entities and then to analyze it probabilistically. Below we describe the approach in detail by defining the required notions and notations.

4.1. *Search Results and Identified Entities*

For a given query submitted by the user, let A be the set of the top-L hits (e.g., $L = 200$) returned by the underlying search system. For a hit $a \in A$, let $ent(a)$ denote the set of entities that have been identified in a by applying entity mining (e.g., over its snippet, its full contents or over some of its metadata fields). Inversely, let $docs(e) = \{\ a \in A \mid e \in ent(a)\}$ denote the elements of A in which e has been identified. Let $E = \cup_{a \in A} ent(a)$, i.e. E is the set of all entities identified in A. We should stress here that, in our setting, we know the URI of each detected entity name. We obtained this information by exploiting the LOD during the specification (in a preprocessing step) of the EoI. Thereby, the set of detected entities E is actually a set of entity URIs.

Consider the following example, which for now on it will be our running example: The user submits a marine-related query to a search system, e.g., the query *"bonito"*. The search system applies entity mining (with *fish species* from DBpedia as the EoI) in the snippets of the top-10 hits and identifies the following three entities: "Striped bonito" (`http://dbpedia.org/resource/Striped_bonito`), "Sarda" (`http://dbpedia.org/resource/Sarda`), and "Blackfin tuna" (`http://dbpedia.org/resource/Blackfin_tuna`).

4.2. *The Graph SEGIE*

Here we describe how to enrich the set of detected entities E by exploiting the structured knowledge enclosed in one or more RDF graphs, in order to construct what we call SEGIE (Semantically Enriched Graph of Identified Entities), which is an RDF graph, denoted by $\mathcal{X}$.

Let us first formalize the structured knowledge available as LOD or queryable through a SPARQL endpoint. Consider an infinite set U of RDF URI references, an infinite set B of blank nodes [22] and an infinite set L of literals. A triple $(s, p, o) \in (U \cup B) \times U \times (U \cup B \cup L)$ is called an *RDF triple* (s is called the *subject*, p the *predicate* and o the *object*). An *RDF graph* G is a set of RDF triples. For an RDF Graph G_i we shall use U_i, B_i, L_i to denote the URIs, blank nodes and literals that appear in the triples of G_i respectively. The *nodes* of G_i are the values that appear as subjects or objects in the triples of G_i. Figure 3(a) depicts a simple RDF graph; node "`b`" (_:b) represents a blank node, node "`d`" (*Thunnus atlanticus@en*)

is a literal (specifically a string in English), while the other nodes represent URIs (for improving readability we have omitted the namespaces).

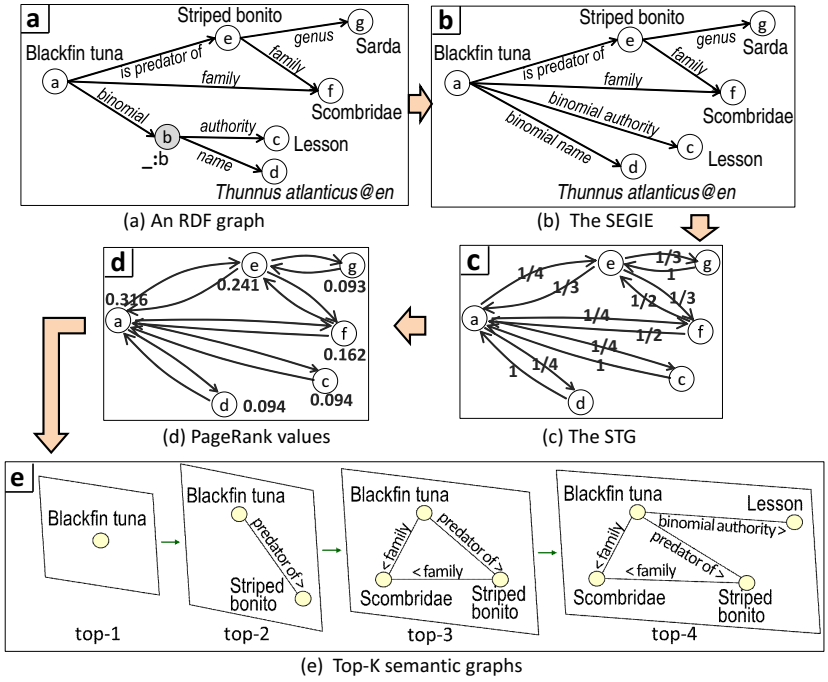


Fig. 3

Now we describe how from a set of entities E and an RDF Graph G_i, we will define the SEGIE $\mathcal{X}$. This corresponds to Step 3 of the process described in §2. Let $out(e) = \{o \mid (e, p, o) \in G_i\}$, i.e. all objects that are *pointed by* an entity $e \in E$, and $in(e) = \{s \mid (s, p, e) \in G_i\}$, i.e all subjects that *point to* an entity $e \in E$ (for simplicity, we consider subjects and objects as entities in our setting). We first define the *base set* S as follows $S = E \cup (\cup\{out(e)|e \in E\}) \cup (\cup\{in(e)|e \in E\})$. Note that one could also add to S the sets $out(e')$ and $in(e')$, for each $e' \in out(e) \cup in(e)$, and so on, i.e. subjects and objects in any *radius*. For the moment, we consider radius equal to 1. In the example of Figure 3(a), the nodes "a", "e" and "g" correspond to entities detected in the search results (i.e. entities in E), while the remaining nodes correspond to semantic information derived by exploiting an RDF graph G_i (i.e. entities and properties in $out(e)$ and $in(e)$, for each $e \in E$).

If the object o (or subject s) is a *blank node* b, and in order to avoid losing information that may be important, we include in the graph the set $out(b)$ (or $in(b)$ respectively) and not the blank node b. If $out(b)$ or $in(b)$ contains a blank node, we ignore it. For labeling the edge that connects an entity e with an entity e' in $out(b)$ (or in $in(b)$ respectively), we *concatenate* the names of the properties. For example, if e *birth place* b and b *city* e', then the name of the edge is "*birth place city*". Figure 3(b) depicts the RDF graph of Figure 3(a) without blank nodes.

In addition, we specially treat the `same-as` OWL property because it states explicitly that the corresponding object and subject refer to the same real-world object. Specifically, for a triple (s, p, o), if p is a `same-as` property then we merge the nodes that correspond to s and o.

We also blacklist some RDF properties that provide little insight into the description of an entity (like the properties that describe metadata, e.g., `dbpedia-owl:wikiPageID`), as well as some very generic resource types that are not indeed useful (e.g., `owl:Thing`, `dbpedia-owl:Agent`, etc.). These properties and resource types are chosen manually based on an inspection of a sample of the underlying KB and are provided in a text file. Alternatively, one could exclude these resources by forming accordingly the SPARQL query that is used for retrieving the properties, i.e. in Step 3 of the process described in §2.

The SEGIE $\mathcal{X} = (\mathcal{E}_{\mathcal{X}}, \mathcal{P}_{\mathcal{X}})$, where $\mathcal{E}_{\mathcal{X}} = S$ and $\mathcal{P}_{\mathcal{X}}$ is the directed links that connect S, of our running example is shown in Figure 3(b). Informally, a SEGIE is an RDF graph corresponding to a set of entity URIs which however does not contain blank nodes, `same-as` edges and some (blacklisted) properties and resources.

The next step is to apply on SEGIE a PageRank-like [23] algorithm for identifying the *more important* entities and properties. We prefer to follow a PageRank-inspired method because the underlying theoretical framework is solid (random walks and stochastic processes) and it can be customized (biased) according to the needs of different types of applications (as it will be shown later). The intuition behind PageRank (which was proposed and has been successful in web search) is that the *important* web pages are pointed by several other *important* web pages. Analogously, in our problem an entity is considered *important* (and thus it is worth presenting to the user) if several other *important* entities point to it. Below we detail the probabilistic analysis.

4.3. *The State Transition Graph (STG)*

We will define a STG $\mathcal{G} = (\mathcal{E}, \mathcal{P})$ over which a *random walk model* can be applied, i.e. its nodes $\mathcal{E}$ correspond to *states* and its edges $\mathcal{P}$ to (state) *transitions*. For each node in $\mathcal{X}$, we create a node in $\mathcal{G}$. For each directed edge $(e_1 \rightarrow e_2)$ in $\mathcal{X}$, we create *two* directed edges in $\mathcal{G}$; one of the same direction $(e_1 \rightarrow e_2)$ and one of the opposite direction $(e_2 \rightarrow e_1)$. We do that because we consider that if a property connects two entities, then these entities are *semantically biconnected*. For example, in Figure 3(a) the entities *Blackfin tuna* and *Scombridae* are connected by the relation *family*, meaning that *Blackfin tuna "belongs to the family" Scombridae* and equivalently that *Scombridae "is the family of" Blackfin tuna*, i.e. the difference lies in how we name the property. In addition, in $\mathcal{G}$ we collapse multiple directed edges that connect two nodes in $\mathcal{X}$ in one directed edge, but we also specify accordingly the *edge weights*. Note that the weights of the outgoing edges from a given entity must represent transition probabilities, i.e. they must sum to 1. For a given entity $e \in \mathcal{E}$, let $o(e) \subseteq \mathcal{P}$ be the set of outgoing (directed) edges of e, and $i(e) \subseteq \mathcal{P}$ the set of incoming (directed) edges. Let also $props(e, e') \subseteq o(e)$ be the set of (directed)

edges that connect e with e' (i.e. the properties that connect the two entities). Note that in our setting $|o(e)| = |i(e)|$ and $|props(e, e')| = |props(e', e)|$. The weight of the single outgoing edge that connects e with e' is $\frac{|props(e,e')|}{|o(e)|}$. Figure 3(c) illustrates the STG that corresponds to the SEGIE of Figure 3(b) and also shows the edge weights as described above.

4.4. *Analyzing the STG*

Here we describe how we analyze the above graph for identifying the important entities and properties. The PageRank-like value $r(e)$ of an entity e is defined as:

$$r(e) = q \cdot Jump(e) + (1 - q) \cdot \sum_{e' \in i(e)} \frac{|props(e', e)|}{|o(e')|} r(e') \quad (1)$$

where q is a decay factor (typically set to 0.1-0.2), while $Jump(e)$ expresses the probability of random jumps to e, and thus it can be defined as $Jump(e) = \frac{1}{|S|}$ if we assume uniform distribution (S is the *base set*). The initial PageRank value of each entity also equals $\frac{1}{|S|}$. The equivalent matrix equation form is: $\mathbf{r} = q \cdot \mathbf{J} + (1-q) \cdot \mathbf{T} \cdot \mathbf{r}$, where $\mathbf{J}[e_i] = Jump(e_i)$ and $\mathbf{T}$ is the transition matrix:

$$\mathbf{T}(e, e') = \begin{cases} 0 & \text{if } (e' \rightarrow e) \notin \mathcal{P} \\ \frac{|props(e', e)|}{|o(e')|} & \text{if } (e' \rightarrow e) \in \mathcal{P} \end{cases} \quad (2)$$

Note that the value of an entity e is the sum of two components: one part of the value is equal for all entities and expresses the probability of a random jump to e, and the other part comes from entities that point to e. The values can be computed iteratively and iterations should be run to convergence. According to [23], the number of iterations required for convergence is empirically O(log$n$), where n is the number of edges. In Appendix A, we provide the PageRank-like algorithm for identifying the most important entities and properties.

Figure 3(d) shows the PageRank values regarding the STG of Figure 3(c) (with decay factor 0.15 and performing 10 iterations). The ranking is the following:

1. `Blackfin tuna (node "a")`
2. `Striped Bonito (node "e")`
3. `Scombridae (node "f")`
4. `Lesson (node "c") and Thunnus atlanticus (node "d")`
5. `Sarda (node "g")`

We notice that the entities *Blackfin tuna* and *Striped bonito* have the highest PageRank values because they have many connections and are also interconnected.

4.5. *Promoting the Entities of the Top-ranked Hits*

So far we have ignored the rank of the results in which an entity occurred. However, it is reasonable to consider that the top-ranked results will probably contain more useful entities than the low-ranked results. To capture this, here we introduce a *biased* version of the scoring scheme. Instead of assuming a uniform distribution for the random jumps, we will now bias it. Specifically:

$$Jump(e) = \frac{HitScore(e)}{\sum_{e' \in E} HitScore(e')} \tag{3}$$

where (as in [3]):

$$HitScore(e) = \sum_{a \in docs(e)} ((|A| + 1) - rank(a)) \tag{4}$$

where $rank(a)$ stands for the position of an $a \in A$ in the answer (the first hit has rank equal to 1, the second 2, etc.). This means that the probability of a random jump to e is higher if e has been identified in the top ranked documents. The probability of a random jump to an entity that has not been identified in the search results is zero. Notice that the above scoring scheme could be adjusted to use similarity scores instead of ranking scores. Since though most web search systems provide only the ranking of the results, we use this more general scoring scheme.

To grasp the effect of the biased approach, in our running example consider that: *Striped bonito (node "e") was detected in the 1st, 2nd and 3rd result, Sarda (node "g") was detected in the 1st and 3rd result, while Blackfin tuna (node "a") was detected in the 8th result only.* By running the biased version of PageRank as described above (with decay factor 0.15 and performing 10 iterations), we now get the following ranking:

```
1. Striped bonito (node "e"), with score 0.331
2. Blackfin tuna (node "a"), with score 0.260
3. Sarda (node "g"), with score 0.150
4. Scombridae (node "f"), with score 0.149
5. Lesson (node "c") and Thunnus atlanticus (node "d"), with score 0.055
```

Now the top-scored entity is *Striped bonito*, *Sarda* has gained two ranks, while *Blackfin tuna*, *Scombridae*, *Lesson*, and *Thunnus atlanticus* have lost one rank. We notice that the entities identified in the top search results have been promoted.

One could exploit the biased version for supporting also various other kinds of *personalization*, e.g., promotion of entities of one or more particular categories (RDF classes) or those coming from particular KBs, etc.

4.6. *Top-K Semantic Graphs*

Apart from producing and returning the top-K entities, say $\mathcal{E}_\mathcal{K}$ ($\mathcal{E}_\mathcal{K} \subseteq \mathcal{E}$), as derived by the biased PageRank algorithm, we can return (at query time or on-demand) the top-K graph $\mathcal{G}_\mathcal{K} = (\mathcal{E}_\mathcal{K}, \mathcal{P}_\mathcal{K})$ for any K from 1 to $|\mathcal{E}|$, allowing the user to increase or reduce the value of K. The set of edges $\mathcal{P}_\mathcal{K}$ of this graph consists of those elements of $\mathcal{P}$ that connect elements of $\mathcal{E}_\mathcal{K}$, i.e. it is the *restriction* of $\mathcal{P}$ on $\mathcal{E}_\mathcal{K}$, i.e. we can write $\mathcal{P}_\mathcal{K} = P_{|\mathcal{E}_\mathcal{K}}$. Figure 3(e) depicts several top-K semantic graphs of our running example. The value of these graphs is that they show how the top-K entities are connected.[d]

[d]The visualization and the layout of the semantic graphs, as well as the design of the interaction model, are beyond the scope of this paper but an important direction for future research.

The Anatomy of a Top-K Semantic Graph. The possible different *types of vertices* in a top-K semantic graph is an important information which must be taken into account in an exploratory search process, since it characterizes *groups of information* with some common properties. Since a top-K semantic graph is actually an RDF graph, its vertices are either RDF resources or literal values (in our setting, by construction a top-K graph does not contain blank nodes). However, some resources correspond to *categories of entities*, including resource classes (e.g., `dbpedia-owl:Fish`, `dbpedia-owl:Country`) and SKOS [24] concepts (e.g., `category:Fish_of_Hawaii`). Moreover, a resource may actually be a *web address*, i.e. a URL that represents a web page, photo, etc. We can distinguish the vertices of a top-K semantic graph as follows:

– $\mathbf{V}_{\texttt{ans}}$**:** Vertices corresponding to entities detected in the query answer. Each entity is associated with one or more URIs.
– $\mathbf{V}_{\texttt{rel}}$**:** Vertices corresponding to entities related to the detected entities. These resources have not been detected in the search results, but they were derived by exploiting the LOD. Each entity is associated with one or more URIs.
– $\mathbf{V}_{\texttt{ctg}}$**:** Vertices corresponding to categories of entities. These resources have been derived by exploiting the LOD and each one is associated with a URI.
– $\mathbf{V}_{\texttt{lit}}$**:** Vertices corresponding to literals, i.e. numeric values, strings, dates, or boolean values, derived by exploiting the LOD (e.g., birth date).
– $\mathbf{V}_{\texttt{web}}$**:** Vertices corresponding to web addresses, i.e. URLs that represent web pages, photos, etc. These URLs have been derived by exploiting the LOD.

We call *clusters* these "groups" of vertices and their union constitutes the set of vertices $\mathcal{E}_{\mathcal{K}}$, i.e. $\mathsf{V}_{\texttt{ans}} \cup \mathsf{V}_{\texttt{rel}} \cup \mathsf{V}_{\texttt{ctg}} \cup \mathsf{V}_{\texttt{lit}} \cup \mathsf{V}_{\texttt{web}} = \mathcal{E}_{\mathcal{K}}$. Figure 4 depicts an example of a graph containing vertices of these clusters. Considering that only the entity *Yellowfin tuna* was detected in the search results, vertex A is of type $\mathsf{V}_{\texttt{ans}}$, vertex B is of type $\mathsf{V}_{\texttt{rel}}$, vertex C is of type $\mathsf{V}_{\texttt{ctg}}$, vertex D is of type $\mathsf{V}_{\texttt{lit}}$, and vertex E is of type $\mathsf{V}_{\texttt{web}}$. In the default case in which we retrieve LOD only for the entities identified in the search results (i.e. for $radius = 1$), the graph contains edges which are subset of the following cartesian products: $\mathsf{V}_{\texttt{ans}} \times \mathsf{V}_{\texttt{ans}}$, $\mathsf{V}_{\texttt{ans}} \times \mathsf{V}_{\texttt{rel}}$, $\mathsf{V}_{\texttt{ans}} \times \mathsf{V}_{\texttt{ctg}}$, $\mathsf{V}_{\texttt{ans}} \times \mathsf{V}_{\texttt{lit}}$, and $\mathsf{V}_{\texttt{ans}} \times \mathsf{V}_{\texttt{web}}$.

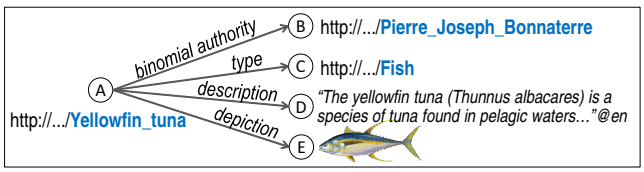


Fig. 4: The main vertex types of a top-K semantic graph.

Of course, one could refine some clusters, e.g., dichotomize web addresses to *images* and *web pages*. One could even specify clusters according to the application context, e.g., a cluster that represents entities detected in a particular "interesting" set of results, etc.

The clustering of vertices in a top-K semantic graph provides useful information which can be exploited by the interaction model and the underlying visualization/layout algorithm. In §5.2, we report the results of a user study regarding the importance of these clusters in an exploratory search process. We also report experimental results regarding the distribution of vertices in these clusters for several top-K graphs produced by a prototype system.

5. Evaluation

In this section, we first present the results of a survey regarding the usefulness of the proposed approach (§5.1). Subsequently, we comparatively evaluate the proposed ranking scheme with other ranking methods (§5.2). Furthermore, we compare the proposed approach with the plain entity mining approach (§5.3). Finally, we report experimental results regarding the efficiency of the proposed approach (§5.4). We also discuss how we can achieve scalability and reliability (§5.5).

5.1. *Usefulness*

In order to get a first feedback for the usefulness of the proposed approach, we performed a survey regarding the *marine* domain. The objective is to study whether the depiction of associations among the derived semantic information (through a top-K semantic graph related to the search results) can help the users in an exploratory search process.

5.1.1. *Setting*

The survey is based on a questionnaire (Google Form) in which we ask participants related to the marine domain (like marine biologists) to answer a few questions related to five particular queries (each query corresponds to a different *query type*). At first, for each query we derive the top-5 semantic information (entities and properties) by applying the proposed approach (described in §4), specifically by performing entity mining in the top-100 snippets as returned by Bing search engine, with *Fish Species* as the EoI and using DBpedia as the underlying KB. Then, we depict this semantic information in two different ways: as a *top-5 list* (as proposed in [3]) and as *top-5 graph*. We select to show the top-5 list and graph (and not for example the top-10 or the top-20) because we do not want the quality of the visualization of the graph to affect the participants' opinion (since this is beyond the scope of this paper). Then, the participant must answer the following question:
Q1. *"In an exploratory search process regarding the query <`here the query`>, how would you prefer to see the top-5 entities and properties related to that query?"*.

The participant can select one of the following options: *Only the LIST is enough*, *Only the GRAPH is enough*, *I would like to see BOTH*, *I do not want to see neither the list nor the graph*. The participant must answer the above question for each one of the five queries. In the next step (in a new page), we ask the participant to answer the following question (again for each one of the five queries):

Q2. *"In an exploratory search process regarding the query <here the query>, do you believe that the appearance of a graph of semantic information related to the search results can help the user during his/her search process?"*

The participant can select one of the following options: *Yes*, *Maybe Yes - it depends on the interaction model and the quality of the visualization of the graph*, *Maybe No*, *No*. Clearly, the type of result expected depends on the *type* of the query. For example, a query such as *tuna species* is looking for instances of a class of entities, while a query like *yellowfin tuna* is looking for information for one particular entity, in this case a certain tuna species. Pound et al. [25] proposed a classification of queries from a semantic search point of view by expected result:

- *Entity query:* its intention is to find information about a particular entity.
- *Type query:* its intention is to find entities of a particular type or class.
- *Attribute query:* its intention is to find values of a particular attribute of an entity or type.
- *Relation query:* its intention is to find how two or more entities or types are related.
- *Other keyword query:* its intention is described by some keywords that do not fit into any of the above categories.

The existence of these query types is very important in our problem since each type requires a different type of result and thus must be evaluated differently by the human judge. Thereby, the participants must answer the aforementioned two questions for five different queries, each one belonging to a different type. Specifically, we selected the following queries: *yellowfin tuna* (entity query), *jack fishes* (type query), *chum salmon genus* (attribute query), *zander and walleye* (relation query), *fishing in Hawaii* (other keyword query). Note that the selection of the queries to use in the questionnaire does not affect the purpose of this survey since this is not a task-based evaluation. Unless we explicitly define the interaction model we cannot perform a task-based evaluation (this is the next step of this research).

5.1.2. *Results*

We distributed the questionnaire to marine biologists and to persons working on marine-related projects who have a basic knowledge on marine species. 30 subjects participated in the user study (22 to 60 years old), from 6 countries and 12 organizations. Table 1 depicts the results.

As regards Q1, we notice that the majority of the participants (67% to 84%) would like to see a graph representation (ONLY GRAPH or BOTH) of the top-5 entities. As expected, the biggest percentage is for the query *zander and walleye* which is a relation query, and the smallest is for the query *fishing in Hawaii* which does not belong to a particular type. In addition, we notice that for the first three types of queries (*entity*, *type* and *attribute*), more participants prefer to see both a list and a graph which means that the graph representation can be offered *on-demand* as a complementary representation which enables the users to inspect the connectivity of the identified entities.

Table 1: Survey results.

Query	Q1					Q2				
	only list	only graph	both	no list, no graph	χ^2 test $p =$	yes	maybe yes	maybe no	no	χ^2 test $p =$
yellowfin tuna **(entity query)**	23%	30%	43%	3%	0.01857	23%	63%	10%	3%	$9.537e^{06}$
jack fishes **(type query)**	13%	37%	47%	3%	0.002262	27%	67%	7%	0%	$4.31e^{07}$
chum salmon genus **(attribute query)**	17%	37%	43%	3%	0.00694	33%	57%	7%	3%	$5.052e^{05}$
zander and walleye **(relation query)**	13%	43%	40%	3%	0.002905	33%	53%	13%	0%	0.000205
fishing in Hawaii **(other query)**	23%	37%	30%	10%	0.1979	30%	43%	23%	3%	0.01857

Regarding Q2, we notice that the majority of the participants (73% to 94%) believe that the appearance of a graph of semantic information related to the search results can help users during a search process (YES or MAYBE YES). In addition, most of them consider that the success of this approach depends on the interaction model and the quality of the visualization of the graph (MAYBE YES). Notice also that if we consider both the percentages of users that selected ONLY GRAPH or BOTH in Q1 and the percentages of users that selected YES or MAYBE YES in Q2, the latter is always bigger. The above are a strong rationale for elaborating in the future on the interaction model and on the visualization of the top-K graphs. We also notice that for the last query (of type *other query*) which is a quite general query, a high percentage of participants (26%) selected MAYBE NO or NO.

Statistical significance. In order to check for the randomness of our results, we conducted a *Pearson's χ^2 statistical test (Goodness of Fit)* [26] which is appropriate for categorical data, testing the null hypothesis that the different answers of Q1 and Q2 respectively have been equally represented in our results. We used $\alpha = 0.05$ which is widely used in the bibliography. As regards Q1, the results depicted in the respective column of Table 1 show that we can reject the null hypothesis for the first four queries with a Type-I statistical error of 5%, while we cannot reject it for the last one, i.e. for the query *fishing in Hawaii* of type "other query". However, we should note that this query does not seem to fit a particular information need for the marine biologists (it is a quite general query), and probably this is the reason for receiving many negative selections in both Q1 and Q2. As regards Q2, the results in the respective column of Table 1 show that we can reject the null hypothesis for all the queries. Here, again the last query has the highest p value. From the above results, we can support that the reported results are statistically significant for the first 4 types of queries, i.e. it is unlikely to have occurred by chance alone.

5.2. *Effectiveness: Comparative Results on Ranking*

5.2.1. *Setting*

We performed a user study regarding the marine domain. The objective is to evaluate the effectiveness of the proposed PageRank-based ranking scheme (described in §4). Specifically, we comparatively evaluated the proposed Biased PageRank algorithm (`BiPR`) with i) the plain PageRank algorithm (`PR`), and ii) a Spreading

Activation [27] algorithm (SA). As regards the PageRank-based algorithms (BiPR and PR), we set decay factor 0.15 and we performed 50 iterations. Regarding SA, we adopted a similar approach with [12] and [20]. However, we set the initial activation of a node that represents an identified entity to be the score of this entity as derived by Formula 3, while the remaining nodes have zero activation. In addition, we set the decay factor α to equal 0.85 which in our setting appears to produce the best results (the decay factor corresponds to the percentage of activation that is lost every time an edge is processed). We also set the *firing threshold* to equal a very small real number (0.00001) because we want all nodes representing entities identified in the results to fire even if their ranking score is very small.

We deployed a web application which implements the proposed functionality. Specifically, the system accepts keyword-based queries and performs entity mining in the top-100 snippets as returned by Bing, with *Fish Species* (from DBpedia) as the EoI and using DBpedia for retrieving the properties of the identified entities. We allowed the users to submit their own queries so that they can better judge the results. We also stored the results for each submitted query as well as various statistics and features.

For each submitted query, the system presents three top-10 lists (the one next to the other with random display order) of ranked semantic information (entities and properties) related to the results, each one produced by one of the aforementioned ranking schemes (BiPR, PR and SA). The user can evaluate the ranking of each list by selecting one of the following options: 1 (poor), 2 (not bad), 3 (good), 4 (very good), 5 (excellent). In addition, the user can inspect many top-K lists, for several values of K ($5 \leq K \leq 50$), so that he/she can better judge each ranking. We also included guidelines and a brief description of the proposed functionality allowing the participants to better understand the context of the evaluation.

Furthermore, we asked participants to answer a question regarding the *importance* of the different types of displayed semantic information. The objective is to get a first feedback regarding the value of the entities and properties (that derive by exploiting the LOD) in an exploratory search process. Specifically, we asked them to answer the following question:

"Taking into account the submitted query <here the query>, please vote the importance of each category of displayed entities:

- Fish Species detected in the search results.

- Fish Species related to the detected species, but not detected in the results.

- Properties (literal and numeric values) related to the detected species.

- Categories related to the detected species.

- Web addresses (e.g., web pages) or photos related to the detected species."

These types of semantic information correspond to the five clusters described in §4.6 (V_{ans}, V_{rel}, V_{lit}, V_{ctg}, V_{web}). The user can select a value between 0 (useless) and 5 (absolutely important).

5.2.2. *Results*

17 subjects performed the evaluation (part of those who completed the survey) and totally 51 queries were submitted. For each submitted query, on average 11.5 entities were detected in the search results, 4,687 triples were derived from DBpedia, and 2,031 entities and properties had to be ranked.[e]

Figure 5 depicts the results. We notice that `BiPR` outperforms `PR` and `SA`, receiving many "4" (very good) scores. Specifically, the average score of `BiPR` was 3.53/5.0 (good to very good) and the median 4 (very good), the average score of `PR` was 2.47/5.0 (not bad to good) and the median 2 (not bad), while the average score of `SA` was 2.90/5.0 (almost good) and the median 3 (good). We can conclude that promoting the entities identified in the top search results affects positively the ranking of the semantic information and thus the elements of the top-K graphs. Moreover, a biased PageRank-inspired method appears to produce better rankings than a Spreading Activation algorithm. Thus, we can also support that the semantic information that is accessible by many (highly ranked) entities is useful for the users.

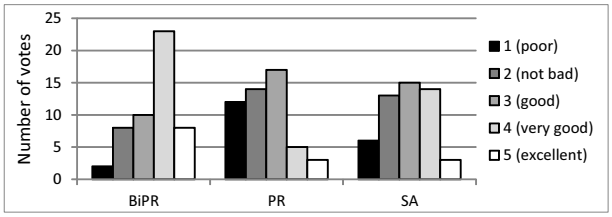


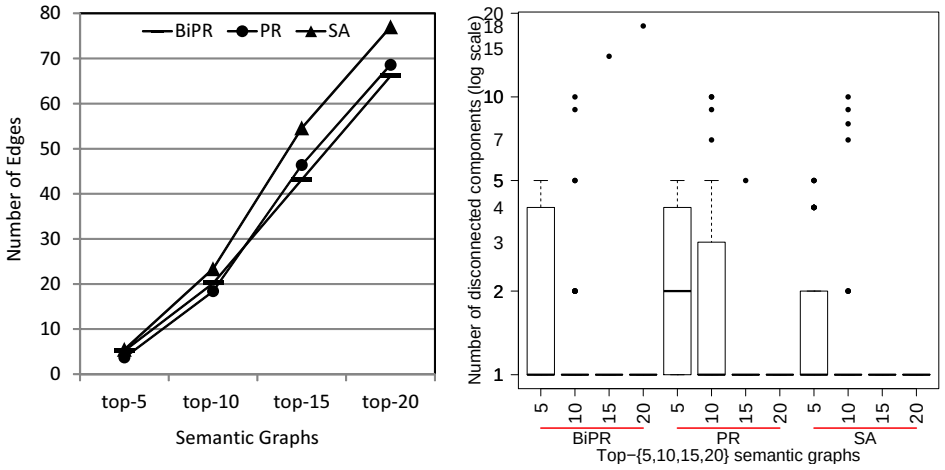
Fig. 5: The scores given to the three ranking schemes.

Regarding the importance of the different types of entities, users considered the entities detected in the search results ($\mathsf{V}_{\mathtt{ans}}$) the most important semantic information with score 4.3/5, which is a predictable result since this information derives by analyzing the results of a given user query. As regards the remaining clusters, their importance vary from 3.2/5 to 3.7/5 ($\mathsf{V}_{\mathtt{rel}}$= 3.3, $\mathsf{V}_{\mathtt{ctg}}$= 3.4, $\mathsf{V}_{\mathtt{lit}}$= 3.7, $\mathsf{V}_{\mathtt{web}}$= 3.2). We notice that the higher score is for the literal properties ($\mathsf{V}_{\mathtt{lit}}$) which correspond to characteristics of the detected entities. This is justified by the fact that the majority of the submitted queries where *entity queries* whose intention is to find information about a particular entity. In general, we notice that the score of all clusters is high meaning that users are interested not only in entities detected in the search results, but also in information related to the detected entities.

5.2.3. *Graph Features*

For each query submitted during the user study we measured various features of the semantic graphs derived by applying each one of the three ranking algorithms

[e]The full results, the identified entities and the RDF triples retrieved from DBpedia for each submitted query, and the top-200 rankings as produced by each one of the three ranking algorithms, are available to download through `http://139.91.183.72/x-ens-2/fullEvalResults.zip`.



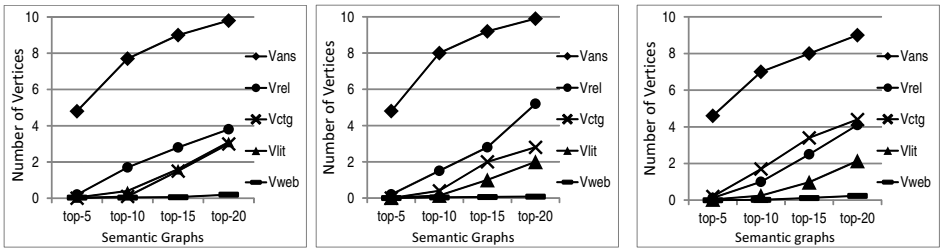
(a) (b)

Fig. 6: (a) Average number of edges and (b) number of disconnected components in top-5, top-10, top-15 and top-20 semantic graphs.

(`BiPR`, `PR` and `SA`). The objective is to inspect if the derived semantic graphs are well-connected as well as what types of vertices they contain, since such information is important for deciding how to visualize the graphs. Another objective is to inspect the difference of both the elements and the relative order of the common elements of several top-K graphs as derived by each one of the three ranking algorithms. If the difference is very low then the results of the evaluation may not be valid since the displayed lists look alike. Finally, and since the participants performed the evaluation in a real prototype system, we checked the average time for running each algorithm as well as the total time of the whole searching procedure.

Graph connectivity. Figure 6a depicts the average number of edges of several top-K graphs, for each one of the ranking algorithms. We notice that all algorithms produce graphs with many edges and, as expected, the more vertices the graph contains, the more "complex" the graph is because of the large number of edges. An interesting remark is that the semantic graphs produced by the SA algorithm have slightly more edges than the graphs produced by the PageRank-based algorithms. This can be justified by the fact that a PageRank algorithm applies random jumps, meaning that it can reach and score vertices with few or no connections.

Figure 6b depicts a boxplot with the number of disconnected components of several top-K graphs (where one disconnected component means a connected graph). As shown, when the value of K $>$ 5, in most cases the graph is connected. The only exceptions are the top-10 case of the PR algorithm, where 50% of the queries result to a disconnected graph, and some other outlier queries for all algorithms (e.g., there is one query with 18 disconnected components). Regarding the top-5 answer, the PR algorithm seems to create the most disconnected components in the returned graph, with only 25% of the queries returning a connected graph, and more than



(a) BiPR (b) PR (c) SA

Fig. 7: Distribution of vertices in top-K semantic graphs.

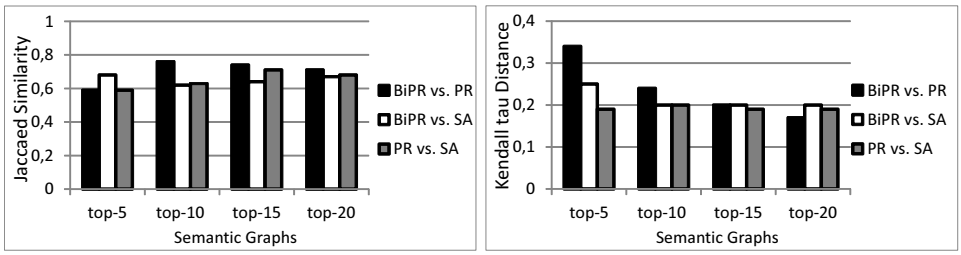
50% of the queries returning graphs with more than 2 disconnected components. It is interesting also that the average number of disconnected components is smaller in SA. This is justified by the ''nature'' of the spreading activation algorithm; the node weights decay as activation propagates through the graph. We should stress here that the *median* is 1 in all cases except for the top-5 graphs produced by PR where the median is 2 (i.e. in most cases the returned graph is connected).

Distribution of vertices. Figure 7 depicts the average distribution of the top-K entities and properties in the five clusters described in §4.6. We notice that in all cases the majority of vertices correspond to entities detected in the search results (type $\mathsf{V}_{\mathtt{ans}}$). We also see that in the top-5 graphs, almost all vertices are of type $\mathsf{V}_{\mathtt{ans}}$. This is a predictable result since: i) in the case of BiPR the vertices that correspond to the detected entities are promoted since they have a non-zero probability of random jump, ii) in the case of PR the vertices that correspond to the detected entities have much more connections compared to the other vertices (since we retrieve their incoming and outgoing properties from the underlying KB), and iii) in the case of SA the vertices that correspond to the detected entities have a non-zero activation value.

We should stress here that both the connectivity of the top-K graphs and the distribution of vertices highly depend on the contents and the quality of the KBs that we exploit. The more information a KB contains, the more data we can retrieve for the detected entities, while if the resources of a KB are well interconnected, the top-K graphs will be also well-connected. In addition, in the same KB a category of entities may contain a lot of information about its instances, while another category may not be "rich" enough. Furthermore, some categories may contain many related entities (i.e. big number of $\mathsf{V}_{\mathtt{rel}}$), while others many literals (i.e. big number of $\mathsf{V}_{\mathtt{lit}}$). Therefore, the quality and the contents of the underlying KBs highly affect the quality and the contents of the top-K graphs.

Jaccard Similarity and Kendall tau Distance. We compared the three ranking algorithms in terms of a) their difference in the top elements, and b) their difference in the order of their common elements.

As regards the difference in the top elements, we compared the algorithms using the *Jaccard similarity coefficient*. If A is the set of the top-k entities and properties



(a) Jaccard Similarity (b) Kendall tau Distance

Fig. 8: Similarity of the top-K semantic graphs.

returned by the one ranking algorithm and B the set of the top-k entities and properties returned by another algorithm, then their Jaccard similarity is $J(A,B) = \frac{|A \cap B|}{|A \cup B|}$. Figure 8a depicts the results for all pairs of algorithms. We notice that in all cases the Jaccard similarity is between 0.6 and 0.75 meaning that the top-K entities and properties differ significantly.

As regards the difference in the order of the elements, we compared the algorithms using the *Kendall tau* distance measure. Let E be a set of elements, and let A and B be two linear orders of E. The *Kendall's tau* distance between A and B, denoted by $K(A,B)$, is equal to the number of bubble sort swaps that are necessary to convert A to B. To define it precisely, let $I_{A,B}$ be a function such that $I_{A,B}(e1,e2) = 0$ if $e1$ and $e2$ are ranked in the same order in both A and B, and $I_{A,B}(e1,e2) = 1$ otherwise. If P denotes the set of all distinct unordered pairs of elements of E, then $K(A,B)$ is defined as $K(A,B) = \sum_{\{e1,e2\} \in P} I_{A,B}(i,j)$ [28]. Since the maximum value of *Kendall's tau* is $|E| * (|E|-1)/2$, occurring when the linear order A is the reverse of B, the distance can be normalized by dividing by $|E| * (|E|-1)/2$, yielding a measure whose values range $[0,1]$. In our case we compared the linear orders produced by two ranking algorithms. In this comparison, we considered as set E, the set of elements that exist in the corresponding top-K sets of both linear orders. Figure 8b depicts the results for all pairs of algorithms. We notice that the Kendall tau distance ranges from about 0.17 to 0.35, meaning that the difference in the linear orders is not negligible even when comparing only the order of the common elements.

Average Time. For each submitted query, we recorded the time required to run each one of the three algorithms. The time for running the PageRank-based algorithms (BiPR and PR) was about 80 ms, while the time for running SA was about 5 ms. We notice that the SA algorithm is more efficient, however the time for running the PageRank-based algorithms is also very low. We should also note that the total time (including the time to fetch the results, the time to perform entity mining in the top-100 snippets, and the time to retrieve the properties of the detected entities) was on average less than 3 seconds. This demonstrates the efficiency of the proposed approach for a marine-related scenario. In §5.4 we report the results of an *extensive* evaluation regarding the efficiency of each task of the proposed approach.

5.2.4. *Discussion*

We should stress that there is not any standard evaluation procedure and collection for our purpose. Approaches like the SemSearch Challenge [29] and the SEALS Project [30] (i.e. retrieve resources from the underlying semantic collection regarding a query) cannot be applied in our problem because the proposed approach does not retrieve resources that much the criteria described by a query, but it *semantically post-analyzes* the results of a search system. Thus, there are many parameters that affect the results like the quality of the hits, the quality of the underlying KBs, the specified EoI, the underlying entity mining algorithm.

5.3. *Plain Entity Mining approach vs. Link Analysis approach*

Below we compare the proposed approach with the method of [3], i.e. with an approach that does not exploit the structured knowledge that is available for the detected entities. Let A be the set of the top-K entities as derived by the link analysis approach. Let B denote the set of the top-K entities as produced by [3]. We want to answer the following questions:

(a) How different A and B are (if the difference is low then what we propose may have minor impact).

(b) Is the relative order of the entities which belong to both A and B equal or different? This allows testing whether the connectivity obtained through LOD affects the order; if only the biased jumps determine the ranking then A and B are expected to be quite similar.

(c) Understand the kind of elements of B which are not in A. Are they elements occurring in search results or not (if not then they were fetched from LOD).

We ran experiments using the 51 queries submitted in the aforementioned user study with *Fish Species* as the EoI and DBpedia as the underlying KB. We performed entity mining in the top-200 snippets returned by Bing and, as regards the link analysis approach, we ran PageRank with decay factor 0.15 and performing 50 iterations.

Differences in the top-K entities. We compared the top-5, top-10, top-15 and top-20 entities using the *Jaccard similarity coefficient* as described in the previous section. The results showed that on average the top-5 and the top-10 entities have Jaccard similarity about 0.65, the top-15 have Jaccard similarity about 0.60, while the top-20 have Jaccard similarity about 0.57. This means that they differ significantly.

Differences in the order of the common elements. Regarding the difference in the *order* of the entities, we used the *Kendall's tau* distance measure as described in the previous section. In our case we compared the linear order produced by the plain entity mining approach with the linear order produced by the link analysis approach. In this comparison, we considered as set E, the set of entities derived by entity mining *only*, i.e. we ignored the entities that came from the LOD. The average *Kendall's tau* distance ranges from 0.23 to 0.3. If we recall that in the link analysis

approach the probability of a random jump to an entity has been defined just like the formula that scores the entity in the plain entity mining approach [3], it follows that the associations and the properties fetched from the LOD are responsible for a change of 23%-30% (which is not negligible) in the ranked order. Specifically, the link analysis approach favors the entities that have many associations with other highly ranked entities.

In the same experiments we also compared the linear order of the top-5, top-10, top-15 and top-20 entities (not all the returned entities as previously), ignoring the entities that do not exist in the corresponding top-K entities of both approaches, with a view to clarify the positions in which there are differences in the linear order. On average, the top-5 entities have *Kendall tau* distance 0.08, meaning that often (but not always) there is a pair of entities in the top-5 list with different order in the two approaches. The top-10 entities have *Kendall tau* distance 0.10 (meaning that there are about 4-5 pairs of entities in the top-10 list with different order), while the top-15 and top-20 entities have *Kendall tau* distance about 0.20.

5.4. *Efficiency*

Performing real-time entity mining (using Gate Annie) in the top-100 snippets returned by a web search system costs about 1 second [3, 31] (the cost is about 10 ms per snippet). Here we measure the average time for i) creating the SEGIE (with DBpedia as the underlying KB), ii) creating the STG, iii) running PageRank, and iv) creating the top-500 graph, for various numbers of randomly selected entities. We run the experiments for entities belonging to 10 randomly selected RDF classes (i.e. categories of entities): *Tennis Player*, *Boxer*, *Country*, *Philosopher*, *Drug*, *Disease*, *Chemical Substance*, *Bacteria*, *Fish*, and *Golf Player*. In a real setting, the randomly selected entities correspond to entities discovered in the search results or entities related to the detected entities (if we consider radius > 1). For achieving accuracy, we repeated the experiments 20 times and we computed the average values.[f]

Regarding i), i.e. the creation of the SEGIE, we decided to access DBpedia at *real-time* (and not to download its datasets and load them in one or more servers) because we want to preserve the dynamic nature of our approach (since the LOD constantly changes and increases). For each entity, DBpedia offers its data (properties and related entities) online in various forms: JSON, XML, triples and N3/Turtle. We access the data in the N3/Turtle form. As regards the objects in the triples that represent literals, we retrieve only those in English language. Although DBpedia offers a SPARQL endpoint, we decided to access its data by directly parsing the N3/Turtle pages because this fits our problem (since for each entity we want only its incoming and outgoing properties) and because the parsing of pages turned out

[f]The experiments were carried out using an ordinary laptop with processor Intel Core i5 @ 2.4Ghz CPU, 4GB RAM and running Windows 7 (64 bit). The implementation is in Java 1.7 and for the creation and the management of the graphs we use the Java Universal Network/Graph Framework (JUNG) (`http://jung.sourceforge.net/`).

Table 2: Graphs statistics and creation time, and PageRank execution time, for several number of (randomly selected) entities.

# entities	SEGIE #vertices	SEGIE #edges	STG #edges	Top-500 Graph #edges	SEGIE creation time	STG creation time	Time for Running PR	Top-500 creation time
50	2,573	3,790	7,580	889	1.4 sec	28 ms	194 ms	42 ms
100	4,133	6,193	12,386	1,493	2.9 sec	95 ms	329 ms	68 ms
500	20,743	34,816	69,632	3,471	13 sec	298 ms	1.7 sec	343 ms
1,000	49,954	84,893	169,786	3,411	27 sec	480 ms	3.9 sec	552 ms
10,000	528,815	995,981	1,991,962	3,421	258 sec	8 sec	58 sec	22 sec

to be much more efficient than running SPARQL queries (allowing us to run concurrent threads). Table 2 reports the average values and also includes the main characteristics of the graphs in order to better understand how these characteristics affect the times. Moreover, in Appendix B we provide a table which summarizes the time dependencies of each task of the proposed approach.

As expected, the most time consuming task is the creation of the SEGIE since for each entity we access DBpedia at *real-time* and retrieve its related LOD. Notice that the time is linear in relation to the number of detected entities. For up to 100 detected entities (which is the common case for snippet-mining), the time is on average less that 3 seconds. For bigger number of detected entities the time is higher, e.g., for 1,000 entities about 30 seconds are required. We should stress that this is often acceptable in professional search, e.g., persons working in *patent* offices spend many hours for a particular patent search request (the same is true in *bibliographic* and *medical* search).

The rest of the tasks require around one order of magnitude less time. We can conclude that the proposed functionality can be offered at real-time for about 100 identified entities even if we query an online KB like DBpedia. In addition, as we have already said, this functionality can be also offered *on-demand* so the user can decide if he/she wants to pay the cost.

5.5. *Scalability and Reliability*

As previously noted, the described process can be time consuming if the number of detected entities is big. Although such timings (in the scale of minutes) are often acceptable in professional search, we can mitigate scalability issues as follows:

Bounding the response time. One reasonable (default) policy that we adopt for being scalable is to retrieve LOD (Step 3 of the process described in §2) only for the top-m (e.g., $m = 100$) detected entities as returned by the plain entity mining approach [3]. These top-m entities exist in most of the top-ranked results, therefore they are probably the more important. Thereby, we bound the maximum response time.

Reliability. From our experimentation with LOD, the existing online KBs (like DBpedia) are not reliable since they mainly serve demonstration purposes. The fact that everyone can query them affects their efficiency and availability. We expect that this problem will be handled in the near future since such technologies

mature and get used in applications. In the meantime, to increase the reliability and the efficiency of the proposed approach, we could adopt a caching mechanism or we could index a part of the underlying KB. Of course, in a real application the underlying KBs may not be publicly available or a *dedicated warehouse* can be constructed that will only serve the intended application (like the marineTLO-based warehouse [32]). The KBs or the warehouse can also be distributed in many servers, taking advantage of load balancing techniques [33].

6. Conclusion

In this paper, we have focused on integrating search results with LOD for offering advanced exploratory search services and making the LOD available to the end users. Towards this direction, we have used *entity names* as the "glue" for automatically connecting search results with LOD. However, the amount of structured information that is available for the identified entities (their properties and related entities) can be very high. For selecting the entities and properties that better characterize the search results and their context, we have proposed a Link Analysis-based method which promotes the entities identified in the top ranked results as well as the semantic information that is linked with many highly ranked entities. The produced top-K semantic graphs allow the users to instantly inspect information that may exist in different places and that may be laborious and time consuming to locate, avoiding thereby the disengagement of the users from their initial task. In addition, they provide useful information about the context of the identified entities and allow the users to get a more sophisticated overview and to make better sense of the results.

We performed a survey regarding the marine domain which showed that the majority (more than 70%) of the participants a) would like to see a graph representation of the top-5 entities regardless the type of the submitted query, and b) believe that the appearance of a graph of semantic information related to the search results can help them during an exploratory search process. The conducted statistical significance test rejects the randomness of the reported results with a 5% Type-I error. We also reported comparative results which illustrated that the proposed (biased PageRank-like) ranking scheme produces more preferred rankings compared to other algorithms (about 22% better ranking compared to a Spreading Activation algorithm and 43% compared to the plain PageRank algorithm). As regards efficiency, we have analyzed experimentally the costs of all steps and we have seen that for up to 100 detected entities (which is the case for snippet-mining), we can offer the proposed functionality at real-time (in less than 4 seconds) even if we access an online KB like DBpedia. Finally, we showed how we can bound the maximum response time (for being scalable) and we also discussed approaches on how to achieve reliability.

We believe that the result of this research can provide a general, flexible and adaptive method for enriching search results with structured knowledge in the con-

text of a session-based exploratory search interaction. In future, amongst others, we plan to elaborate on the interaction model and on the visualization of the semantic graphs.

Acknowledgements

We thankfully acknowledge the support of *iMarine* (FP7 Research Infrastructures, 2011-2014) and *MUMIA* COST action (IC1002, 2010-2014).

References

[1] G. Marchionini, "Exploratory Search: from Finding to Understanding," *Communications of the ACM*, 2006.
[2] C. Bizer, T. Heath, and T. Berners-Lee, "Linked Data - the Story so far," *Semantic Web and Information Systems*, vol. 5, no. 3, 2009.
[3] P. Fafalios, I. Kitsos, Y. Marketakis, C. Baldassarre, M. Salampasis, and Y. Tzitzikas, "Web searching with entity mining at query time," in *Proceedings of the 5th Information Retrieval Facility Conference*, (Vienna, Austria), July 2012.
[4] P. Fafalios and Y. Tzitzikas, "X-ens: Semantic enrichment of web search results at real-time," in *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, (Dublin, Ireland), 2013.
[5] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "Dbpedia: A Nucleus for a Web of Open Data," in *The semantic web*, Springer, 2007.
[6] "GATE Annie." `http:/gate.ac.uk/ie/annie.html`.
[7] P. Fafalios, M. Baritakis, and Y. Tzitzikas, "Configuring named entity extraction through real-time exploitation of linked data," in *4th International Conference on Web Intelligence, Mining and Semantics (WIMS'14)*, (Thessaloniki, Greece), ACM, June 2014.
[8] "GeoNames." `http://www.geonames.org/`.
[9] "DrugBank." `http://www.drugbank.ca/`.
[10] T. Cheng and K. Chang, "Beyond Pages: Supporting Efficient, Scalable Entity Search with Dual-Inversion Index," in *EDBT'10*, 2010.
[11] T. Cheng, X. Yan, and K. C.-C. Chang, "EntityRank: Searching Entities Directly and Holistically," in *VLDB'07*, 2007.
[12] C. Rocha, D. Schwabe, and M. Aragao, "A Hybrid Approach for Searching in the Semantic Web," in *WWW'04*, 2004.
[13] M. Ciglan, K. Nørvåg, and L. Hluchỳ, "The SemSets Model for Ad-hoc Semantic List Search," in *WWW'12*, 2012.
[14] A. Harth, S. Kinsella, and S. Decker, "Using Naming Authority to Rank Data and Ontologies for Web Search," in *ISWC'09*, 2009.
[15] R. Delbru, N. Toupikov, M. Catasta, G. Tummarello, and S. Decker, "Hierarchical Link Analysis for Ranking Web Data," in *ESWC'10*, 2010.
[16] B. Bamba and S. Mukherjea, "Utilizing Resource Importance for Ranking Semantic Web Query Results," *Semantic Web and Databases*, 2005.
[17] J. Kleinberg, "Authoritative Sources in a Hyperlinked Environment," *Journal of the ACM (JACM)*, 1999.
[18] L. Dali, B. Fortuna, T. Duc, and D. Mladenic, "Query-Independent Learning to Rank for RDF Entity Search," in *ESWC'12*, 2012.
[19] L. Ding, R. Pan, T. Finin, A. Joshi, Y. Peng, and P. Kolari, "Finding and Ranking Knowledge on the Semantic Web," in *ISWC'05*, 2005.

[20] A. Agrawal, S. Sudarshan, A. Sahoo, A. Sandalwala, and P. Jaiswal, "Entity Ranking and Relationship Queries Using an Extended Graph Model," in *COMAD'12*, 2012.
[21] "Google Knowledge Graph." `http://www.google.com/insidesearch/features/search/knowledge.html`.
[22] D. Beckett and B. McBride, "Rdf/xml syntax specification (revised)," *W3C recommendation*, vol. 10, 2004.
[23] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web," 1999.
[24] "Simple Knowledge Organization System." `http://www.w3.org/2004/02/skos/`.
[25] J. Pound, P. Mika, and H. Zaragoza, "Ad-hoc Object Retrieval in the Web of Data," in *WWW'10*, pp. 771–780, ACM, 2010.
[26] R. L. Plackett, "Karl pearson and the chi-squared test," *International Statistical Review/Revue Internationale de Statistique*, pp. 59–72, 1983.
[27] F. Crestani, "Application of Spreading Activation Techniques in Information Retrieval," *Artificial Intelligence Review*, vol. 11, no. 6, 1997.
[28] M. G. Kendall, "Rank correlation methods," Griffin, London, 1970.
[29] "SemSearch Challenge." `https://km.aifb.kit.edu/ws/semsearch11/`.
[30] "SEALS Project." `http://www.seals-project.eu/`.
[31] P. Fafalios, M. Salampasis, and Y. Tzitzikas, "Exploratory Patent Search with Faceted Search and Configurable Entity Mining," in *1st International Workshop on Integrating IR technologies for Professional Search (ECIR 2013 Workshop)*, (Moscow, Russia), March 2013.
[32] Y. Tzitzikas, C. Alloca, C. Bekiari, Y. Marketakis, P. Fafalios, M. Doerr, N. Minadakis, T. Patkos, and L. Candela, "Integrating heterogeneous and distributed information about marine species through a top level ontology," in *Proceedings of the 7th Metadata and Semantic Research Conference (MTSR'13)*, (Thessaloniki, Greece), November 2013.
[33] V. Cardellini, M. Colajanni, and P. S. Yu, "Dynamic Load Balancing on Web-Server Systems," *Internet Computing, IEEE*, vol. 3, no. 3, 1999.

Appendix A. PageRank Algorithm

Algorithm 1 A PageRank algorithm for detecting the important entities and properties.

Require: $\mathbf{T}$ (transition matrix), $\mathbf{J}$ (random jumps matrix), q (decay factor), N (number of PageRank iterations).
Ensure: $\mathbf{r}$ (PageRank scores).

$\mathbf{r} = \mathbf{J}$ //initialize the PageRank score of all entities
for $i = 1$ **to** N **do**
  $\mathbf{r} = q \cdot \mathbf{J} + (1 - q) \cdot \mathbf{T} \cdot \mathbf{r}$
end for
return $\mathbf{r}$

Appendix B. Time Dependencies

Table 3 synopsizes the time dependencies of each task of the proposed approach. The time for retrieving the top results depends only on the underlying search system. The time for performing entity mining in the top results depends on i) the number of the top results that we want to mine, ii) the part of the answer in which

Table 3: Time Dependencies Table

Task	Time depends on:
Retrieving (top) results	underlying search system
Performing entity mining	i) num of results to analyze, ii) part of the answer upon which we perform entity mining (e.g., snippets or full contents), iii) efficiency of the entity mining algorithm
Creating the SEGIE	i) num of detected entities, ii) underlying KBs, iii) categories of the detected entities
Creating the STG	num of triples in SEGIE (i.e. number of edges)
Running PageRank	i) num of iterations, ii) number of edges in STG
Creating the top-K graph	num of vertices and edges in SEGIE

we perform entity mining (e.g., mining the full contents requires much more time than mining the snippets), and iii) the efficiency of the underlying entity mining algorithm, which in turn is affected by many parameters (e.g., number of supported categories, NLP/ML algorithm, etc.). The time for creating the SEGIE depends on the number of detected entities, the efficiency of the underlying KBs and the categories of the detected entities (since the entities of some categories may contain numerous incoming or outgoing properties). The time for creating the STG depends on the number of triples in SEGIE, while the time for running PageRank depends on the number of iterations and on the number of edges in the STG. Finally, the time for creating the top-K semantic graph depends on the size of SEGIE.