Toward Social, Structured and Semantic Search

Raphaël Bonaque[1,2], Bogdan Cautis[2,1] François Goasdoué[3,1], and Ioana Manolescu[1,2]

[1] INRIA [2] Université Paris-Sud [3] Université de Rennes 1

Abstract. Social content such as social network posts, tweets, news articles and more generally web page fragments is often *structured*. Such *social* content is also frequently enriched with annotations, most of which carry *semantics*, either by collaborative effort or from automatic tools. Searching for relevant information in this context is both a basic feature for the users and a challenging task. We present a data model and a preliminary approach for answering queries over such structured, social and semantic-rich content, taking into account all dimensions of the data in order to return the most meaningful results.

1 Introduction

In this work, we introduce S3, a novel data model for *structured, semantic-rich content exchanged in a social context*. The data model builds on widespread Web standards for structured documents (XML/JSON) and semantic Web data (RDF), while it also extends well-established models for data shared and queried in social applications. Based on this data model, we revisit top-k keyword search, well studied in the social context for unstructured (flat) data items, to also take into account document structure and semantics. The implementation and evaluation of a query answering algorithm for this framework is currently ongoing; more details are provided in our technical report [1]. This paper is organised as follows: we give a presentation of the data model and its various components (Sec.2), then a query model (Sec.3) and finally a quick overview of a query evaluation algorithm (Sec.4), and we conclude with the related works (Sec.5).

2 Data model

In this section, we describe our model integrating the social, structured and semantic-rich content we are interested in into a single graph. First, we define RDF graphs and use one as an instance of our problem, then we show how social networks are embedded in such an instance. Next, structured documents are considered along with the links between social, semantics and structure, and finally we introduce some tools to manipulate such an instance.

We assume given a set U of Uniform Resource Identifiers (URIs, in short), as defined by the standard [8], and a set of literals (constants) denoted L; U and L are disjoint.

Keywords We denote by $\mathcal{K} = U \cup stem(L)$ the set of all possible *keywords*, i.e., we consider that *stemming* has been applied on literals prior to their inclusion in $\mathcal{K}$. In this paper, literals are systematically shown between quotes, e.g., "literal".

2.1 RDF

We adopt a standard model for semantic Web data, as follows. An *RDF graph* (or *graph*, in short) is a set of *triples* of the form `s p o`, stating that its *subject* `s` has the *property*

Constructor	Triple	Relational notation under Open Word Assumption
Subclass constraint	$\mathtt{s} \prec_{\mathrm{sc}} \mathtt{o}$	$\mathtt{s} \subseteq \mathtt{o}$
Subproperty constraint	$\mathtt{s} \prec_{\mathrm{sp}} \mathtt{o}$	$\mathtt{s} \subseteq \mathtt{o}$
Domain typing constraint	$\mathtt{s} \hookleftarrow_{\mathrm{d}} \mathtt{o}$	$\Pi_{\mathrm{domain}}(\mathtt{s}) \subseteq \mathtt{o}$
Range typing constraint	$\mathtt{s} \hookrightarrow_{\mathrm{r}} \mathtt{o}$	$\Pi_{\mathrm{range}}(\mathtt{s}) \subseteq \mathtt{o}$

Fig. 1. RDFS statements.

p and the value of that property is the *object* o. A triple is *well-formed* whenever its subject, s, belongs to U, its property, p, belongs to U and its object, o, belongs to $\mathcal{K}$. In what follows, we only consider well-formed triples.

RDF Schema A valuable feature of RDF is RDF Schema (RDFS), which allows enhancing the resource descriptions provided by RDF graphs. An RDFS declares *semantic constraints* between the classes and the properties used in these graphs, through the use of RDF built-in properties, as summarized in Figure 1. In this figure, $\mathtt{s}, \mathtt{o} \in U$, while domain and range denote respectively the first and second attribute of every property.

Within the RDF standard [7], the built-in property `rdf:type`, denoted a from now on, is used to describe the *classes* to which a resource belongs, i.e., resource typing.

Saturation RDF graphs also feature *implicit* triples, notably when an RDFS is available. For instance, if $\mathrm{x\ a\ c_1}$ and $\mathrm{c_1} \prec_{\mathrm{sc}} \mathrm{c_2}$, then the implicit triple $\mathrm{x} \prec_{\mathrm{sc}} \mathrm{c_2}$ holds. A graph in which all implicit triples have been made explicit is termed the *saturation* (or closure) of the RDF graph. *In the following, we assume RDF graphs are saturated.*

Weighted RDF graph For the needs of our model, we introduce *weighted* triples of the form (s, p, o, w) such that (s, p, o) is an RDF triple, and $w \in [0, 1]$ is the *weight* of the (s, p, o) triple. *In the following, the weight of any RDF triple is assumed* 1 *unless specified otherwise.* We define the saturation of a weighted RDF graph as the saturation derived from *certain* facts (triples with weight 1) only, using the standard RDF [7].

S3 namespace and instance To model semantically rich data, we rely on a small special-purpose set of RDF classes and properties, identified in this paper by the *S3 namespace* prefix. Relationships between documents, users, comments, keywords, social connections etc. naturally lead to a graph structure. We encode them as weighted RDF triples in a special graph called an *S3 instance*, denoted I, as explained next.

2.2 Social network

We distinguish a non-empty set $\Omega \subset U$ of *social network users* and for each $u \in \Omega$ we add $\mathrm{u\ a\ S3{:}user}$ 1 to I where $\mathrm{S3{:}user}$ is a dedicated RDF class.

We introduce the property $\mathrm{S3{:}social}$ generalizing the various relationships which may connect users in the social network. For every relationship from a user u_1 to a user u_2 having the weight w, we have: $\mathrm{u_1\ S3{:}social\ u_2\ w} \in \mathrm{I}$.

In our model, the higher the weight, the closer we consider the two users are.

2.3 Documents

We consider that content is created under the form of structured *documents* e.g., XML, JSON, etc. A document is an unranked, ordered tree of *nodes*. Let N be a set of node names (e.g., legal XML or JSON node names), and ϵ be a special symbol denoting the empty node name (allowed in JSON for instance). Any node has an *URI*.

We denote by D the subset of U consisting of node URIs. Each node has a *name* from $N \cup \{\epsilon\}$, and a *content*: *a set of keywords* from $\mathcal{K}$. We assume every text node has been broken in words, stop words have been removed, and the remaining words are

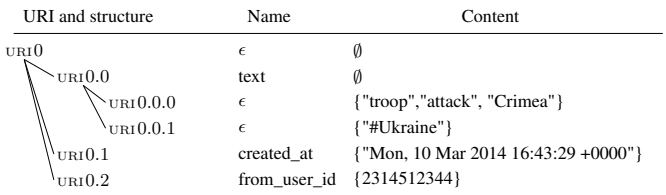

Fig. 2. Sample S3 document.

stemmed to obtain our version of the node's text content. For example, in Figure 2, the original text of URI0.0.0 could be *"Troops are attacking Crimea"*. For simplicity, we consider the *URI of a document* to be that of its root node.

We term any subtree rooted at a node in document d a *fragment* of d. The set of fragments (nodes) of a document d is denoted $Frag(d)$. We may use f to refer interchangeably to a fragment f and to its URI.

Document-derived triples We capture the relationships between documents, fragments and keywords through a set of RDF statements using S3-specific properties.

We introduce a dedicated RDF class S3:doc corresponding to the documents, and:

- For each $d \in D$, we have d a S3:doc $1 \in \mathrm{I}$;
- For each document $d \in D$ and each fragment n of d: n S3:partOf $\mathrm{d}\,1 \in \mathrm{I}$;
- For each node n and keyword k in the content of n: n S3:contains $\mathrm{k}\,1 \in \mathrm{I}$;
- For each node n with name m: n S3:nodeName $\mathrm{m}\,1 \in \mathrm{I}$.

Example 1. Based on the document in Figure 2, the following triples are in I:
URI0.1 S3:partOf URI0 1, URI0.0.0 S3:contains "Crimea" 1, and
URI0.0 S3:nodeName text 1.

Fragment position We assume available a function $pos(d, f)$ returning the *position* of fragment f within document d, as the sequence of nodes from node d to node f.

2.4 Relations between structure, semantics, and users

The multiple facets of our data model are interconnected, reflecting the relationships between users, content, and semantics. We model these relationships by using a set of S3 classes and properties, as we outline below.

Connections between documents and semantics Document semantics can be characterized by I quadruples relying on a class we call hereafter S3:relatedTo. Intuitively, S3:relatedTo *accounts for the multiple ways in which a node of a document can be related to a keyword by a user*. S3:relatedTo is thus used to make statements about the semantics of a document (or node), as illustrated by the following example, where user u_4 states that the fragment URI0.0.0 is somehow related to Russia:

$$\mathrm{a}\ \mathrm{a}\ \text{S3:relatedTo}\ 1 \qquad \mathrm{a}\ \text{S3:hasSubject}\ \text{URI}0.0.0\ 1$$
$$\mathrm{a}\ \text{S3:hasKeyword}\ \text{"Russia"}\ 1 \qquad \mathrm{a}\ \text{S3:hasAuthor}\ \mathrm{u}_4\ 1$$

The above four triples are part of I; a is an ad-hoc resource whose role is to encapsulate the S3:relatedTo statement.

The property S3:hasSubject can take values either from D or from other instances of S3:relatedTo. The latter allows to express higher-level annotations, i.e., make statements about the semantics of other statements.

We define by T the set of resources of type S3:relatedTo *and we refer to them as tags.*

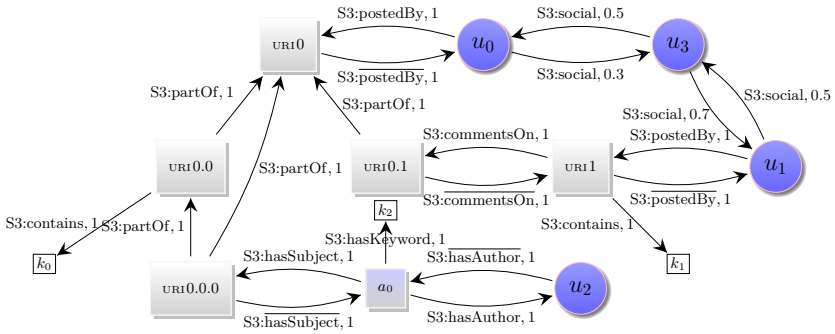

Fig. 3. Sample graph I.

We stress that we do not aim at a single, "standard", interpretation of S3:relatedTo, since there are already many concrete instantiations thereof. For instance, a natural language processing (NLP) tool may recognize a text fragment as related to a person or a topic, a user may state that a document is related to a certain topic, etc. Any such concrete flavors of "being related to" can be used, and we consider that they are all stated to be *subclasses (in RDFS sense)* of our special class S3:relatedTo.

Connections between users and documents We identify two different relationships between users and documents, as detailed below.

1. Users can *post* documents. We model this using a dedicated property S3:postedBy; for each user u having posted a document d, $\mathrm{d}\ \mathrm{S3{:}postedBy}\ \mathrm{u}\ 1$ is in I.
2. Users can also create documents that *comment* on other documents. We capture this using the dedicated property S3:commentsOn. For each user u having created a document c as a comment on the document d, the following hold:
$$\mathrm{c}\ \mathrm{S3{:}commentsOn}\ \mathrm{d}\ 1 \in \mathrm{I} \text{ and } \mathrm{c}\ \mathrm{S3{:}postedBy}\ \mathrm{u}\ 1 \in \mathrm{I}.$$

Observe that when user u posts a document c, which comments on a document d, and possibly cites part of d, each fragment of d exactly replicated in the comment is now part of c, and thus has a new URI.

Example 2. Assuming that $\mathrm{URI}1$, posted by u_1, is a comment on $\mathrm{URI}0$, we have: $\mathrm{URI}1\ \mathrm{S3{:}postedBy}\ \mathrm{u}_1\ 1 \in \mathrm{I}$ and $\mathrm{URI}1\ \mathrm{S3{:}commentsOn}\ \mathrm{URI}0\ 1 \in \mathrm{I}$.

Inverse properties Observe that the relations introduced previously are interesting to explore *in both directions*. For instance, it is interesting to know what comments have been posted on a given document d, and by whom, but it is also interesting to know "What are the comments posted by user u?", or "What does this comment refer to?". To support bidirectional manipulation of such relationship, we introduce in our model a set of *inverse properties* for S3:postedBy, S3:commentsOn, S3:hasSubject and S3:hasAuthor, denoted respectively:

$$\mathrm{S3{:}}\overline{\mathrm{postedBy}}, \mathrm{S3{:}}\overline{\mathrm{commentsOn}}, \mathrm{S3{:}}\overline{\mathrm{hasSubject}}, \mathrm{S3{:}}\overline{\mathrm{hasAuthor}}$$

with the straightforward semantics: $\mathrm{s}\ \bar{\mathrm{p}}\ \mathrm{o}\ 1 \in \mathrm{I}$ iff $\mathrm{o}\ \mathrm{p}\ \mathrm{s}\ 1 \in \mathrm{I}$, for $p \in \{\mathrm{S3{:}postedBy}, \mathrm{S3{:}commentsOn}, \mathrm{S3{:}hasSubject}, \mathrm{S3{:}hasAuthor}\}$.

2.5 Manipulating the S3 instance I

As we will show, in our framework, user queries are expressed using search keywords. Based on the semantics encoded in the RDF schema, we define for every keyword $k \in \mathcal{K}$ a simple semantic extension as follows:

Definition 1 (Keyword extension). *Given a S3 instance* I *and a keyword* $k \in \mathcal{K}$*, the extension of* k*, denoted* $Ext(k)$*, is defined as follows:*

- $k \in Ext(k)$*;*
- *for any weighted triple* $\mathrm{b\ a\ k\ 1}$*,* $\mathrm{b} \prec_{\mathrm{sc}} \mathrm{k\ 1}$ *or* $\mathrm{b} \prec_{\mathrm{sp}} \mathrm{k\ 1}$ *in* I*, we have* $b \in Ext(k)$*.*

For example, given the keyword *fish* and assuming that $\mathrm{tuna} \prec_{\mathrm{sc}} \mathrm{fish\ 1}$ holds in I, we have *tuna* $\in Ext($*fish*$)$. Note that the extension is not a generalization, in the sense that it does not introduce any loss of precision: whenever we include a keyword (URI) k_2 in the extension of another keyword k_1, it follows from the RDFS in I that k_2 is an *instance* of or a *specialization* (particular case) of k_1.

Social components A notion of *distance* is natural in a social context such as the one we consider. We consider that a *subset* $\mathrm{I_{net}}$ of I's edges encapsulate quantitative information on the strength of the links between users, documents and tags. The network edges, $\mathrm{I_{net}}$, are exactly those labeled with properties from the $S3$ namespace (other than $\mathrm{S3{:}partOf}$), and whose subject and object are either users, documents, or tags.

For instance in Figure 3, $\mathrm{u_1\ S3{:}social\ u_3\ 0.5}$ and $\textsc{uri}0\ \mathrm{S3{:}\overline{postedBy}\ u_1\ 1}$ are network edges but $\textsc{uri}0.0\ \mathrm{S3{:}contains\ k_0\ 1}$ and $\textsc{uri}0.1\ \mathrm{S3{:}partOf}\ \textsc{uri}0\ 1$ are not: the first triple's object is neither a user, a document, or a tag but a keyword while the second has the property $\mathrm{S3{:}partOf}$.

Observe that the above notion of network edges: (i) includes any edge between two social network users (given that for any s-labeled edge between two users $u_1, u_2 \in \Omega$, s is a sub-class of $\mathrm{S3{:}relatedTo}$); (ii) includes all the $S3$-prefixed properties we introduced, such as commenting upon, authoring, annotating etc.; (iii) does not include relations between document fragments. While a document node can be seen as "close" to its children or ancestors, two distant nodes from the same document may be harder to relate. We formalize the possible connections between document nodes by introducing:

Definition 2 (Document vertical neighbourhood). *Two documents are* vertical neighbours *if one of them is a fragment of the other. We define by* $neigh$ *the set of vertical neighbours of an URI.*

In Figure 3, $\textsc{uri}0$ and $\textsc{uri}0.0.0$ are vertical neighbours; $\textsc{uri}0.0.0$ and $\textsc{uri}0.1$ are not.

We are interested in vertical neighborhoods, because we consider that a social interaction (e.g., a comment or tag) upon a document fragment *carries over* to both the descendants and the ancestors of this fragment. As a consequence, we define:

Definition 3 (Social path). *A* social path *(or simply a* path*) in* I *is a chain of network edges such that the end of each edge and the beginning of the next one are* either the same or vertical neighbours*. We may also designate a path simply by the list of nodes it traverses, when there is no ambiguity as to the edges taken.*

In Figure 3, $u_2 \xrightarrow{\mathrm{u_2\ S3{:}hasAuthor\ a_0\ 1}} a_0 \xrightarrow{\mathrm{a_0\ S3{:}hasSubject\ URI0.0.0\ 1}} \textsc{uri}0.0.0 \dashrightarrow \textsc{uri}0 \xrightarrow{\mathrm{URI0\ S3{:}\overline{postedBy}\ u_0\ 1}} u_0$ is an example of such a path, stepping from $\textsc{uri}0.0.0$ to its vertical neighbour $\textsc{uri}0$ (the dotted arrow denotes this).

We use the notation $x \rightsquigarrow y$ to denote *the set of all social paths leading from node* x *or one of its vertical neighbours to* y *or one of its vertical neighbours in* I. Finally, $|p|$ denotes the number of edges along the path p.

Given a path p, the **normalized path** from p is the copy of p where the edge weights have been modified so that the sum of the weights of the edges outgoing from any source of social edge is 1. In more details, let n be the end point of a network edge in a path and e be the next edge in this path, the normalized weight of e for this path is:

$$w_{normalized\ e} = w_e / \sum_{e' \in out(neigh(n))} w_{e'}$$

where w_e denotes the weight of the edge e and $out(X)$ the network edges outgoing from a set of edges X. *In the following, whenever we work with a path, we first obtain its normalized version and manipulate it instead.*

3 Querying the S3 instance I

Users can search S3 instances by means of ranked search queries; the answer consists of the k top score fragments, according to a joint structured, social and semantic score.

Definition 4 (Query). *A query is a pair of the form (u, ϕ) where $u \in \Omega$ is a user and $\phi \subset \mathcal{K}$ is a set of keywords. We call u the seeker.*

The ranked results to be computed in response to a query require the presence of a numeric **score function**. More precisely, $score(q, d)$ assigns a numerical score from $\mathbb{R}$ to any document d or any fragment f thereof in the graph, for a given query q.

Definition 5 (Query answer). *The top-k answer to a query q is the k highest scoring documents or fragments for q, such that the presence of a document or fragment in a given rank prevents the inclusion of its vertical neighbours at lower ranks in the result.*

Our approach for effective search over the structured, social and semantic graph relies on two core ingredients: (i) a model of relationships between keywords and various graph components (users, documents etc.); (ii) a score function quantifying the interest of a fragment w.r.t. a query. We outline these two components next.

Relationships to keywords, and their sources We need to capture both the explicit and the implicit relationships connecting a document or a resource of type S3:relatedTo to a search keyword. The relationship can be explicit, for instance, when the document contains the keyword, or a member of its extension $Ext(k)$ (Definition 1). Implicit relationships are due to a document (or the S3:relatedTo resource) being connected through a succession of comments and tags, to a search keyword k. For instance, a document d_2 which has been stated as related to k may be a comment on another document d_1; in this case, d_1 is related to k through d_2.

To model these relationships, we introduce a function *relation source*, denoted $rel(d, k)$. For any document d and keyword k, $rel(d, k)$ is a set of tuples of the form $(type, frag, src)$ such that:

- $type \in \{\text{S3:contains}, \text{S3:relatedTo}, \text{S3:commentsOn}\}$ is the kind of relation,
- $f \in Frag(d)$ is the exact fragment of d due to which d is involved in this relation,
- $src \in \Omega \cup D$ (users or documents) records the origin of this relation between d and k, as we explain next.

Concretely, the relation source tuples are built as follows. Let d be in $D \cup T$, let $f \in Frag(d)$, let $k \in \mathcal{K}$, and let $src \in \Omega \cup D$ be a user or a document[4].

1. Documents connected to the keywords of their fragments If the fragment f contains a keyword k, then $(\text{S3:contains}, f, d) \in rel(d, k)$. This holds even if f does not contain k itself, but some $k' \in Ext(k)$.

2. Connections due to S3:relatedTo *statements* Whenever I comprises an annotation a of a fragment f (of document d) by a source (author) src, with a specialization of the keyword k, we record the resulting connection between d and k, as well as the

[4] This is a slight abuse of notation (tags do not have fragments): if $t \in T$ we use $Frag(t) = \{t\}$.

fact that the connection is due to the source of the annotation on f, through the tuple $(\text{S3:relatedTo}, f, src) \in \text{S3:relatedTo}(d, k)$.

Moreover, if the annotation a itself is related to a keyword k (directly or indirectly, due to a source src', and no matter what the relationship type is: we denote this by $(\_, a, src') \in rel(a, k)$), and a is about the fragment f of document d, then $rel(d, k)$ includes the tuple $(\text{S3:relatedTo}, f, src')$. This records that a connects (relates) f and all its ancestors, to the keyword k, and keeps the source src' of the connection.

3. Connections due to S3:commentsOn *statements* Finally, if a relation source tuple connects a comment c on a document d to the keyword k due to some relation source src, then $rel(d, k)$ includes $(\text{S3:commentsOn}, c, src)$. Intuitively, social connections to a comment on d also extend to d.

Social proximity Our second ingredient for a score function is a comprehensive social proximity measure on I. To each pair of vertices connected by at least one social path, it associates a *global* measure (of *all* the paths betwen them).

First, for each path we define a *proximity function* $path\_prox$, returning values in $[0, 1]$, satisfying the following conditions:

- $path\_prox([\,]) = 1$, that is: the proximity is maximal on an empty path;
- for any two paths p_1, p_2 such that the end point of p_1 is the start point of p_2:
$$path\_prox(p_1||p_2) \leqslant min(path\_prox(p_1), path\_prox(p_2))$$
where $||$ denotes path concatenation. Again, this follows the intuition of a proximity measure: it can only decrease as the path gets longer.

Based on this, we express the social proximity $prox : (\Omega \cup D \cup T)^2 \to [0, 1]$, which is a measurement along every possible path:
$$prox(a, b) = \oplus_{path}(\{path\_prox(p),\ p\ \in a \rightsquigarrow b\})$$
where $\oplus_{path}$ is a function which aggregates a set of values in $[0, 1]$ in a single scalar value. We consider all paths and not just the shortest, because different paths traverse different nodes, users, documents and relationships, all of which must be taken into account when computing the score.

Based on the rel tuples and the proximity measure introduced above, we define:

Definition 6 (Generic score). *Given a document d and a query $q = (u, \phi)$, the score of d for q is: $score(d, (u, \phi)) = \oplus_{gen}(\{(k, type, pos(d, f), prox(u, src)) \,|\, k \in \phi, (type, f, src) \in rel(d, k)\})$, where $\oplus_{gen}$ aggregates four-tuples of the form (keyword, relationship type, importance of fragment f in d, social proximity) into a value from $\mathbb{R}$.*

Let us comment on how various aspects of the graph reflect in the generic score.

First, $\oplus_{gen}$ aggregates over all the keywords in ϕ. Further, recall that tuples from $rel(d, k)$ account not only for k but also for all keywords $k' \in Ext(k)$. This is how *semantics* is injected into the score.

Second, the score of d takes into account the relationships between fragments f of d, and the keyword k (or some $k' \in Ext(k)$). We use $pos(d, f)$ as an indication of the structural importance of the fragment within the document. Document structure is thus taken into account both directly through pos, and indirectly, since the rel tuples previously introduced also propagate relationships from fragments to their documents.

Third, the score takes into account the *social* component of the graph, through $prox$. This function accounts for the relationships between the seeker u, and the various parties (users, documents and tags), denoted src, due to which f is related to k.

We outline concrete methods for implementing the $\oplus_{path}, path\_prox$ and $\oplus_{gen}$ functions in our technical report [1].

4 Query evaluation algorithm

The core idea of our algorithm is the following. The graph is explored starting from the seeker and moving to vertices (users, documents, or tags) at increasing distance, looking for *candidate* fragments that may be part of the top-k answer. We compute $Ext(k)$ for each query keyword, since the members of these extensions will be used to compute scores. For each candidate c we maintain a *score interval* with its lowest and highest possible real scores, which are refined during the exploration. When a candidate fragment is discovered, we are aware of at least one path from the seeker to it, but not of *all* of them, and the candidate's score w.r.t. the query may be significantly impacted by a path not visited yet. During the search, candidate documents are kept sorted *by their highest possible score*. The exploration ends when we are certain that no candidate document outside the current top-k candidates may have an *upper bound* above the *minimum lower bound* within the current top k.

For the particular proximity and score functions we considered (based on the literature and adapted to our setting), we have the guarantee that our algorithm correctly produces the query result (Definition 5), as we show in [1].

5 Conclusion and main related work.

To the best of our knowledge, this is the first attempt to model social media with both structured and semantis content. Past research has considered keyword search on structured XML documents, e.g., [2]. Other studies have focused instead on social keyword-based search, ignoring structure and semantics [4,5,9]. In these works, data items are seen as unstructured collections of keywords (tags) assigned by users, and top-k search ranks higher items close to the seeker. [6] studies top-k search over unstructured text documents, exploiting a measure of semantic similarity between keywords, on which query expansion is based in an early-termination algorithm. Such a model ignores both social and structural features. We have previously considered querying interconnected corpora of XML documents and RDF triples [3]; however, that work ignored any social aspect o top-k search.

Our work seeks to extend and integrate the aforementioned models, to support the three dimensions (structured, semantic and social), in the most common query paradigm: the one of keyword query.

References

1. R. Bonaque, B. Cautis, F. Goasdoué, and I. Manolescu. S4: Social, Structured and Semantic Search. `http://pages.saclay.inria.fr/raphael.bonaque/p/9efac4ffa`.
2. L. J. Chen and Y. Papakonstantinou. Supporting top-k keyword search in XML databases. In *ICDE*. IEEE, 2010.
3. F. Goasdoué, K. Karanasos, Y. Katsis, J. Leblay, I. Manolescu, and S. Zampetakis. Growing triples on trees: an XML-RDF hybrid model for annotated documents. *VLDB Journal*, 2013.
4. S. Maniu and B. Cautis. Network-aware search in social tagging applications: instance optimality versus efficiency. In *CIKM*, 2013.
5. R. Schenkel, T. Crecelius, M. Kacimi, S. Michel, T. Neumann, J. X. Parreira, and G. Weikum. Efficient top-k querying over social-tagging networks. In *SIGIR*, 2008.
6. M. Theobald, R. Schenkel, and G. Weikum. Efficient and self-tuning incremental query expansion for top-k query processing. In *SIGIR*, 2005.
7. Resource Description Framework. `http://www.w3.org/RDF`.
8. Uniform Resource Identifier. `http://tools.ietf.org/html/rfc3986`.
9. S. A. Yahia, M. Benedikt, L. V. Lakshmanan, and J. Stoyanovich. Efficient network aware search in collaborative tagging sites. *PVLDB*, 2008.