A Linked Data Model for Facts, Statements and Beliefs

Ludivine Duroyon, François Goasdoué, Ioana Manolescu

A Linked Data Model for Facts, Statements and Beliefs

Ludivine Duroyon
Univ Rennes, Inria, CNRS, IRISA
Lannion, France
ludivine.duroyon@irisa.fr

François Goasdoué
Univ Rennes, Inria, CNRS, IRISA
Lannion, France
fg@irisa.fr

Ioana Manolescu
Inria and LIX (UMR 7161, CNRS and Ecole Polytechnique)
Palaiseau, France
ioana.manolescu@inria.fr

ABSTRACT

A frequent journalistic fact-checking scenario is concerned with the analysis of statements made by individuals, whether in public or in private contexts, and the propagation of information and hearsay ("who said/knew what when"). Inspired by our collaboration with fact-checking journalists from Le Monde, France's leading newspaper, we describe here a Linked Data (RDF) model, endowed with formal foundations and semantics, for describing *facts, statements*, and *beliefs*. Our model combines temporal and belief dimensions to trace propagation of knowledge between agents along time, and can answer a large variety of interesting questions through RDF query evaluation. A preliminary feasibility study of our model incarnated in a corpus of tweets demonstrates its practical interest.

KEYWORDS

Data journalism, linked data, fact-checking, tweet analysis

1 INTRODUCTION

An interesting class of questions investigated in journalistic fact-checking is the analysis of *who said what when*. Such an analysis may be made in order to determine where a public figure up for (re)election stands with respect to a given issue (a famous example is John Kerry's Senate voting history on the war in Irak[1]), or the public positions of members of a whole political party on an issue (e.g., the projected Wall between the US and Mexico[2]). Statements are made by individuals or organizations, on certain topics, and typically claim to refer to (real-world) facts. It is common for different actors to make different statements about the same fact or about each other statements. An actor may also make different statements about the same thing at different points in time. Professional standards of journalistic work lead to a high interest in modeling and being able to show statement *sources*, which extends our (informal) definition of data of interest to: *who said what when where*. The source can be public (e.g., a speech whose transcript is available on the Web, or a tweet) or it can be private (e.g., an email that journalists acquire through their sources, or which they create, e.g., transcribing a conversation with a source).

Many free or commercial systems and many research prototypes allow analyzing online media to answer specific questions, for instance, CrowdTangle allows to monitor social media and extract events, Twitonomy and TwitterAnalytics are specifically devoted to analyzing Twitter content etc. In this work, we establish a *generic data model for real-world facts, statements, and beliefs*, including (but not limited to) those expressed on social networks; after modeling data in this way, an endless spectrum of fine-granularity analyses can be applied by leveraging a database management system. *Time* plays an important role in our setting, since we must capture when events occur (or when facts hold), and when different statements are made about them; this serves, for instance, to track position reversals in time, or to keep track of promises[3]. Therefore, we incorporate and extend temporal database elements into our model. Further, we take inspiration from classical AI techniques for modeling *agents' beliefs* in order to correctly reflect the connections between actors and their statements. Finally, we adopt the W3C's Resource Description Framework (RDF) concrete graph data model to make databases described in our model *easy to share and combine (link)* with any other RDF dataset, e.g., one that classifies actors according to their political opinions, connections to companies etc., to enable even more analyses.

Beyond being "white-box" (as opposed to models not publicly shared, used by existing media analysis tools), the biggest advantage of our proposed model is to be *comprehensive* (modeling all the above aspects: facts, agents, beliefs, and information propagation), *interoperable* (being RDF), *extensible* (other data sources can be turned into instances of our model) and endowed with *formal semantics*. Our second contribution, beside the model, is to show how we can exploit it through *queries*, both interesting and feasible with off-the-shelf tools.

The remainder of this paper is organized as follows. We recall some preliminaries, present our model, and illustrate interesting queries on its instances. We describe preliminary experiments in a concrete use case, briefly discuss related works, then conclude.

2 PRELIMINARIES

We present next the notions of RDF graphs [16] and how they can be queried with the popular conjunctive fragment of SPARQL [17].

RDF graphs. An RDF graph is a set of triples (s, p, o) where *s* is termed the *subject*, *p* the *property*, and *o* the *object*; such a triple states that *s* is described with the property *p* that has value *o*. *Well-formed* triples, as per the RDF specification, belong to $(\mathcal{I} \cup \mathcal{B}) \times \mathcal{I} \times (\mathcal{I} \cup \mathcal{L} \cup \mathcal{B})$, where $\mathcal{I}$ is a set of *Internationalized Resource Identifiers* (IRIs in short), $\mathcal{L}$ is a set of *literals* (constants), and $\mathcal{B}$ is a set of *blank nodes* that, similarly to labeled nulls [1, 9], representing unknown IRI or literal values.

Notation. Blank node names start with the letter *b* and literals are written as string between quotes, e.g., "string".

An RDF graph models a set of *assertions*, each of which expresses either an instance of *class* (unary relation) or an instance of *property*

[1] https://www.factcheck.org/2013/09/kerry-spins-his-record-on-iraq/
[2] https://www.politifact.com/truth-o-meter/statements/2019/jan/09/donald-trump/trump-democrats-reverse-border-wall-position/
[3] https://www.lemonde.fr/les-decodeurs/visuel/2018/05/07/un-an-apres-son-election-emmanuel-macron-tient-il-ses-promesses-de-campagne_5295281_4355770.html

RDF statement	Triple	Short notation
Class assertion	$(s, \text{rdf:type}, o)$	(s, τ, o)
Property assertion	(s, p, o) with $p \neq \text{rdf:type}$	(s, p, o)

Table 1: RDF assertions.

(binary relation). The syntax of these assertions is shown in Table 1; we introduce the shorthand notation τ to denote the standard rdf:type property, recommended by the W3C for specifying the class(es) to which a resource belongs.

Example 2.1 (RDF triples). The facts comprised in the phrase "Presidential candidate Trump visits Moscow; Michael D. Cohen works for him"[4] can be modeled with the three triples:
(D.Trump, candidateFor, president), (D.Trump, visits, Moscow) and (M.D.Cohen, worksFor, D.Trump).

Querying RDF. SPARQL is the W3C's standard to query RDF graphs. We consider the popular SPARQL conjunctive queries, a.k.a. Basic Graph Pattern Queries (or BGPQs, in short), a core subset of SPARQL 1.1. In particular, we consider BGPQs extended with *property paths*.

A BGPQ q is of the form $q(\bar{x}) \leftarrow t_1, \ldots, t_n$ where $t_1, \cdots, t_n$ are generalized triples called *triple patterns*, in which variables may be used as subject, property or object; $\bar{x}$ is the set of q's *answer variables*, which is a subset of the variables used in $t_1, \cdots, t_n$.

Example 2.2 (BGPQ). The query below asks for the presidential candidates and their collaborators:

$$q(x, y) \leftarrow (\text{x, candidateFor, president}), (\text{y, worksFor, x})$$

Its answer on the triples in Example 2.1 is: (D.Trump, M.D.Cohen).

Further, in our BGPQs, we allow the use of a property paths in the property position of triple patterns. A property path is recursively defined as either a URI, a variable, or a regular expression on property paths among: $p_1|p_2$ for alternative property paths, p_1/p_2 for a sequence of property paths, $p_1?$ for a property path that occur at most once, p^+ for a property path that occur at least once, and p^* for a property path that may not occur or may occur at least once.

Example 2.3 (BGPQ with property path). The query below asks for those who work for sombody visiting Moscow:

$$q(x) \leftarrow (\text{x, worksFor/visits, Moscow})$$

Its answer on the triples in Example 2.1 is: M.D.Cohen.

3 DATA MODEL

We now describe our data model capable of modeling (timed) beliefs, facts and statements.

3.1 Agents, Time, Facts and Beliefs

Agents can be individuals, organizations (e.g., companies, media etc.), or other "party" which make statements or learn about them. In the following, we model agents as RDF resources; for simplicity of the examples, we may simply use people names to denote them, e.g. Alice, Bob etc. In general, agents are those resources having the type *Agent*.

Time We consider a set T of *time intervals*, each of which has a *start* time and an *end* time. Further, we distinguish three special constants $-\infty$, *now*, $+\infty$ that are used to model time bounds $(-\infty, +\infty)$, as well as the current time. A finite union of time intervals can always be equivalently written as a *disjoint* union of time intervals. By a slight abuse of notation, in the sequel, we will write $t \in T$ to denote any such finite union of disjoint time intervals. Each start time and end time can be represented e.g. following the W3C's XML Schema `dateTime` type[5], under the form YYYY-MM-DD[THH:MM].

Temporal normalization To simplify working with unions of time intervals, we use their normalized forms, which are equivalent smallest unions of non overlapping intervals [5]:

Definition 3.1 (Time normalization). Let $I = \{\iota_1, \ldots, \iota_k\}$ be a set of temporal intervals. The time normalization of I is the set of intervals $I' = \{\iota'_1, \ldots, \iota'_l\}$, where l the smallest integer such that:

- $\iota'_1, \ldots, \iota'_l$ are pairwise disjoint and
- $\iota_1 \cup \ldots \cup \iota_k = \iota'_1 \cup \ldots \cup \iota'_l$.

It is easy to see that I' is well-defined, that is: for any I, I' exists and it is unique. Clearly, I' has at most as many intervals as I, i.e., $l \leq k$. For instance, if $k = 3$, $\iota_1 = [1998, 1999]$, $\iota_2 = [2002, 2005]$ and $\iota_3 = [2003, now]$, we have $l = 2$, $\iota'_1 = [1998, 1999]$ and $\iota'_2 = [2002, now]$. We call *temporal normalization* the procedure which given a set of intervals $\{\iota_1, \iota_2, \ldots, \iota_k\}$ computes the disjoint set of intervals $\{\iota'_1, \iota'_2, \ldots, \iota'_l\}$ described above. From now on, we consider that time is normalized set of intervals; we use ι, possibly with indices, to denote them.

We model intervals as RDF ressources having the type Interval. Each interval has a *begin* and an *end* property specifying the interval bounds, whose values are either of type `dateTime`, or a special constant called *now*. To represent time as a set of intervals, we use an RDF ressource of type Time and a property *hasInterval* taking values of type Interval.

Example 3.2. The time t_0 is the union of the two intervals $\iota_0 =$ [2018-04-07T16:00;2018-04-07T19:30], i.e., that starts on April 7, 2018 at 4:00 PM and ends on April 7, 2018 at 7:30 PM, and $\iota_1 =$ [2018-04-07T21:00;2018-04-07T22:30]. This is stated in RDF as:
$(t_0, \tau, \text{Time}), (t_0, \textit{hasInterval}, \iota_1), (t_0, \textit{hasInterval}, \iota_2)$
$(\iota_1, \tau, \text{Interval}), (i_2, \tau, \text{Interval}),$
$(\iota_1, \textit{begin}, \text{2018-04-07T16:00}), (\iota_1, \textit{end}, \text{2018-04-07T19:30}),$
$(\iota_2, \textit{begin}, \text{2018-04-07T21:00})(\iota_2, \textit{end}, \text{2018-04-07T22:30}).$

For simplicity, if a time is a single interval, we directly associate a begin and an end to a time. For instance if t_0 is only ι_1 then we write : $(t_0, \textit{begin}, \text{2018-04-07T16:00}), (t_0, \textit{end}, \text{2018-04-07T19:30})$.

Fact is used to designate in a generic way any real-life event recorded in the database. In the following, we denote facts by their IDs F_1, F_2 etc. and model each fact as an RDF resource of type *Fact*. Each fact has a *time* property specifying when the fact is supposed to occur. Further information about the fact itself is the value of the property *description*; this can be e.g., a text, or an RDF resource having more properties etc. We assume that time and sign information are always attached to facts, e.g., an interval $[-\infty, +\infty]$ for

[4] This and the next examples are inspired from https://www.nytimes.com/interactive/2019/01/26/us/politics/trump-contacts-russians-wikileaks.html.

[5] See https://www.w3.org/TR/xmlschema11-2/#dateTime

facts that are always true etc. If the database features (F, time, t_1) and (F, time, t_2) with $t_1 \neq t_2$, we represent this as a single fact F with time $t_1 \cup t_2$.

Example 3.3 (Fact). The fact "Trump conducted business in Moscow on July 2-3, 2016" is represented as:
(F_1, τ, Fact), (F_1, *time*, t_1), (t_1, *begin*, 2016-07-02),
(t_1, *end*, 2016-07-03), (F_1, *description*, u_1), (u_1, *who*, D.Trump),
(u_1, *what*, conductsBusiness), (u_1, *where*, Moscow).

This small RDF graph together with those corresponding to the next two examples are sketched in Figure 1; oval nodes denote URIs, while text nodes correspond to literals. Some information such as edge labels, triples stating the time of t_1, t_2 etc. are ommitted to avoid clutter; URIs representing agents are signaled by a "user" pictogram. The reason why some nodes have a yellow background will be discussed shortly.

Beliefs relate agents with the things they believe; we model them as resources of the dedicated RDF type Belief. In this work, we use "believe" to model any among: *having knowledge (being informed) of*, *thinking* or *believing* something etc. A belief can be a positive belief (the agent believes something) or a negative one (the agent doesn't believe it). A belief is characterized by: (*i*) the agent holding the belief, which is the value of a *from* property whose subject is the belief; (*ii*) the time when the agent holds the belief, represented by a *time* property; (*iii*) the belief subject, which is the value of a *believes* property, can be a fact (either one we consider to hold, or one of which we only know that it is stated or believed by some agent), a communication, or another belief; (*iv*) finally, a *sign* property whose values can be + or −, indicating whether the agent actually believes the subject of the belief, or not. For simplicity, we assume the *sign* property is present only when its value is −; when it is not present, we assume its value is +, i.e., the agent does hold the belief.

Example 3.4 (Beliefs). J. Mueller believes since January 5, 2017, and up to now that D. Trump conducted business Moscow in July 2016. Building on the fact F_1 from Example 3.3, we denote this by:
(B_1, τ, Belief), (B_1, *sign*, +), (B_1, *from*, J.Mueller), (B_1, *time*, t_2),
(t_2, *begin*, 2017-01-05), (t_2, *end*, *now*), (B_1, *believes*, F_1).

3.2 Belief sharing: communications

Information (beliefs) spread through time through communications. Each communication is characterized by: (*i*) an agent who is the transmitter, indicated by its *from* property; (*ii*) optionally, one or more agents who are the receivers, indicated by the *to* property; (*iii*) a subject, which is the value of the *communicates* property, which can be a fact, a belief or another communication; (*iv*) a sign, whose value is given by a *sign* property + or −, indicates whether the agent actually agrees or disagrees with the subject of the communication; (*v*) a time encoded by the *time* property. If the receiver is not specified, we assume it is a *public* communications. These are considered available to anyone, e.g., anyone can have access to the newspaper, TV, Web source where the communication was made. Communications with one or more specific receivers are considered private; only the transmitter and the receiver are assumed aware of this communication.

Example 3.5 (Communications). On Dec 1st, 2018, M. D. Cohen states that D. Trump did not conduct business in Moscow in July 2016. We denote this by:
(C_1, τ, Communication), (C_1, *sign*, −), (C_1, *time*, t_3),
(t_3, *begin*, 2018-12-01), (t_3, *end*, 2018-12-01),
(C_1, *communicates*, F), (C_1, *from*, M.D.Cohen)
On Dec 15, 2018, J. Mueller says M. D. Cohen knows, since July 2, 2016, that D. Trump conducted business in Moscow in July 2016:
(C_2, τ, Communication), (C_2, *sign*, +), (C_2, *time*, t_4),
(t_4, *begin*, 2018-04-16), (t_4, *end*, 2018-04-16), (C_2, *communicates*, B_2),
(B_2, τ, Belief), (B_2, from, M.D.Cohen), (B_2, believes, F), (B_2, time, t_5),
(t_5, begin, 2016-07-06), (t_5, end, now).

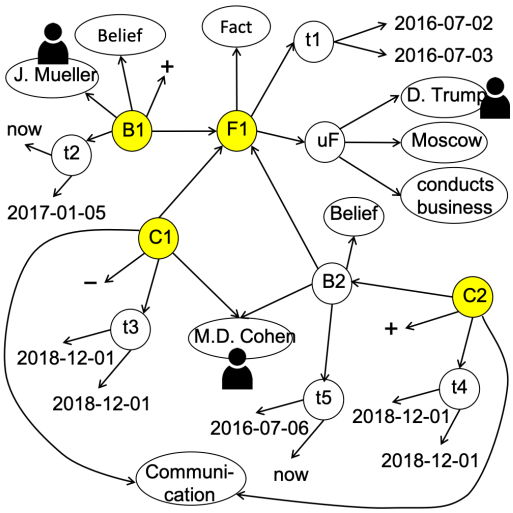

Figure 1: Illustration of the Examples 3.3, 3.4 and 3.5.

3.3 Records and databases

The above examples show that some facts are considered stated as such in the database, e.g., F_1 in Example 3.3, whereas others are not, hence *not declared true* in the database, but merely *believed* or *communicated by* some agents whether they are actually true or not. For instance, the database may state M. D. Cohen believes that D. Trump was on holidays in Moscow in July 2016, in which case the fact that D. Trump was on holidays in Moscow in July 2016 just holds *according to M. D. Cohen* within the database. Similarly, the database may state that an agent *A* holds a belief, or makes a communication, but this is different from the database stating that *according to agent B*, agent *A* believes, respectively communicates it. In the former case, something holds *according to the database*, i.e., is considered an undisputed assertion; in the latter, the database merely states that something just holds according to *B*.

Records We introduce a special type Record which we attach to any fact, belief or communication that holds according to the database, i.e., that the database declares to be true. Note that each record may be the root of a potentially long chain of beliefs and communications; each such chain ends in a Fact[6]. We illustrate this below by revisiting the examples above.

[6]This follows the natural interpretation that any belief or communication carries over something, thus the chain must end in a Fact.

Example 3.6 (Records). In Example 3.3, F_1 holds according to the database. Thus, the triple $(F_1, \tau, \text{Record})$ is also part of the encoding of the fact F in our data model; in Figure 1, nodes of type Record are shown on a yellow background.

In Example 3.4, the belief B_1 holds according to the database. Thus, the triples $(B_1, \tau, \text{Record})$ is also part of it.

Similarly, in Example 3.5, the communications C_1, C_2 took place according to the database, hence the triples $(C_1, \tau, \text{Record})$ and $(C_2, \tau, \text{Record})$ are part of it. However, the belief B_2 is not a database record, because it only holds according to J. Mueller, who communicates on it.

Database Based on the elements introduced above, a *database* can be seen as a *set of fact, belief and communication records*, encoded by a set of RDF triples as discussed above. Figure 1 displays such a database, in which F_1, B_1, C_1, C_2 are the database records.

3.4 Use case: political Twitter scenario

As a particular instantiation of this data model, we have built a database for a French political tweet analysis scenario, as follows. We have subscribed to, and archived, tweets from elected officials, ministries, politicians etc.; over Sept-Dec 2018, we obtained a total of 900.000 tweets (**4.57GB** in JSON format). Out of these, we produced an instance of our RDF data model of size **4.5GB**, as follows. Each tweet is a public Communication, by an agent who is the twitter account. Since each tweet has certainly occurred, it is also a Record. The time of the tweet is a point interval (with the beginning equal to its end). What the tweet communicates depends on the nature of the tweet. (*i*) The content of a *simple tweet* is a text message and possibly photos, links,hashtags and/or agents mentioned; all these can be easily extracted from the corresponding fields within the JSON tweet format. We represent such a tweet content by a Fact which is also a Record. Each such Fact has an ID described by the above mentioned features extracted from the JSON tweet, using the following properties: *text* for the tweet text message, *urlPhoto* for the photos, *urlMedia* for the links and *hashtag* for the hashtags. Further, if the tweet t, by agent $@a$, whose content is modeled by the fact f, mentions another agent (Twitter user) $@b$, we create a communication from $@a$ to $@b$, with the same time moment as t, whose content is the same fact f. (*ii*) The other tweets are either *retweets*, *answers* to a tweet, or tweets that *quote* another tweet; each of this is a tweet t_1 based on another tweet t_0. From t_1, we build a Communication with time and author as above; however, it will be based on the communication corresponding to the content of t_0. Further, answers and quotes (but not retweets) may also have some content of their own; we model that by a new Comment subclass of Fact, and an *addsComment* property which goes from the Communication which models t_1, to the actual text of the answer (or quote) t_1.

Example 3.7 (Tweets). Tweet t_0 from the PublicSénat TV station reads: "#Benalla affair : @CCastaner accuses senators of threatening the republic" with a link to his webpage. A communication c_0 from @publicsenat is created, associated to the fact f_0 of the text, the link, and the hashtag. Another communication c_{0bis} from @publicsenat to @CCastaner on f_0 is created. Twitter user @PadrePio quotes t_0 in his tweet t_1, adding "The new Minister of the Interior, also!" This leads to a communication c_1 from @PadrePio, on c_0 (the declaration of @publicsenat). c_1 has also a comment co_1 with his text.

Further, we used an open dataset about the French National Assembly[7], which we also converted in RDF; this includes the Twitter account information, enabling to connect the tweets to more information about their authors. We added a database of French politician Twitter accounts that journalists from Le Monde shared with us, leading to a total of 48.877 triples describing French political Twitter users.

Our complete corpus (dominated by the triples describing tweets) has 20.697.338 RDF triples. All its Facts and Communications are Records, i.e., there is no "hearsay" (nothing is hypothetical). However, this scenario is sufficient to enable studying the performance of queries that go along chains of Communications: these are materialized by retweets, answers, and/or quotes.

4 QUERYING OUR DATA MODEL

Many interesting queries can be written on an instance of our data model. A few examples include: (*i*) What did M. X. state on date D in his interview with journal J? (*ii*) What is the fact on which members of the political party PA communicate most often, that members of the political party PB do not mention at all? (*iii*) What are the facts on which M. X and Mrs. Y disagree (he believes it whereas she does not)? (*iv*) Which are the facts from a given knowledge base KB that Mrs. Y does not believe (belief with a − sign)? (*v*) Who changed her/his mind, or made incoherent statements, on some fact within a time period (beliefs, or communications, with opposite signs but on a same fact)? (*vi*) What are the informations (beliefs, communications) shared with M. X about fact F before the time moment T? Such queries can be written directly by someone who knows SPARQL, or formulated with the help of a GUI which allows specifying how many facts (beliefs, communications) the query is about, how they connect, fill in the known values (e.g., X, D, J, Y etc. in the above examples), then the GUI generates the corresponding SPARQL statement.

Below, we highlight a family of queries that are interested in tracing the *propagation* of information across chains of beliefs and communications. These queries are theoretically interesting as they translate to property paths, a relatively relatively recent SPARQL feature, yet, as we will show, they are feasible for RDF engines freely available today.

We start with a set of **generic queries**, which can be written on any instance of the data model previously described. Q_0 captures who has heard of what, when, and how. We consider an agent *hears about* something when either the agents believes it, or the agent is the recipient of a communication about it. Q_0 returns: the agent a, the communication c, its time (as an interval spanning from b to e), and its subject s. Note the regular path expression (of potentially unbounded length) going from c to the subject s: it captures the propagation of hearsay, i.e., if c is about a belief b (or another communication c') which is about topic t, then Q_0 returns t together with a and c, since it is through c that a heard of t:

[7] https://github.com/regardscitoyens/nosdeputes.fr/blob/master/doc/api.md#liste-des-parlementaires

$Q_0(?a, ?c, ?b, ?e, ?s) \leftarrow (?c, \tau, \text{Record}),$
$(?c, (from|to), ?a),$
$(?c, (believes|communicates)+, ?s)$
$(?c, time, ?t), (?t, begin, ?b), (?t, end, ?e)$

Building on Q_0, Q_1 finds out who has heard of a specific fact f, and when. It returns the agents, the communications which propagated f to them, and the communication time:

$Q_1(?a, ?c, ?b, ?e) \leftarrow Q_0(?a, ?c, ?b, ?e, f), (f, \tau, \text{Fact})$

Q_2 is a selection on Q_0. It finds what a given agent a has heard of, how, and when:

$Q_2(?c, ?b, ?e, ?s) \leftarrow Q_0(a, ?c, ?b, ?e, ?s)$

Q_3 finds who has heard of what, through which communication, during a given time interval $t0$=$[b0, e0]$:

$Q_3(?a, ?c, ?s) \leftarrow Q_0(?a, ?c, ?b, ?e, ?s), OVERLAP([b, e], [b0, e0])$
$FILTER(?b2 > b), FILTER(?e2 < e)$

Next, we present a few queries **specific to our political Twitter scenario**. To find out how a given hashtag h disseminated on Twitter, query Q_4 returns all the agents reached by the hashtag and the communication which brought the hashtag to them:

$Q_4(?a, ?c, ?b, ?s) \leftarrow Q_0(?a, ?c, ?b, ?s), (?s, hashtag, h)$

Because a tweet has a zero-length temporal interval (b=e), in Q_4 we only use the interval beginning. A restriction of Q_4, query Q_5 ensures that c has taken place in a given time interval:

$Q_5(?a, ?c, ?f) \leftarrow Q_0(?a, ?c, ?b, ?s), (?s, hashtag, h)$
$FILTER(?b2 > b), FILTER(?b2 < b)$

Finally, query Q_6 uses the political party affiliations of agents to aggregate the hashtag targets (agents) by their political party:

$Q_6(?pa, count(*)\ as\ ?n) \leftarrow Q_0(?a, ?c, ?b, ?s), (?s, hashtag, h)$
$FILTER(?b2 > b), FILTER(?b2 < b)$
$(?a, name, ?name), (?a, party, ?pa),$
$GROUP\ BY\ ?pa$

5 EXPERIMENTS

We describe experiments we carried out with instances of our data model built as we explained in Section 3.4.

Platform and settings We aimed to check the feasibility of loading and querying instances of our data model using an off-the-shelf RDF data management system (RDB, in short). We had a particular interest in the capacity of the RDB to handle property path queries. Following a recent benchmark of property path support [13], we have chosen **RDF4J** (formerly known as Sesame) v2.4.2. RDF4J provides B+-tree indexes over the stored triple; an index is defined by an order over four attributes s, p, o (for the standard RDF subject, property, object) and c (for context, i.e., the RDF graph from which a triple comes). By default, the *spoc* and *posc* indexes are built; the application can modify the index set. We added three more indexes *psoc*, *cosp*, and *opsc* to speed up query evaluation. We ran our tests on a Linux computer with an Intel Xeon CPU @ 2.67GHz, 4 Cores and 40GB RAM.

Loading We loaded our corpus of 2×10^7 triples and built the six indexes on it (two default and four we added) in 976 seconds.

Queries We timed the execution of a set of queries on increasingly larger subsets of our dataset, from 1/6th of the it (150.000 tweets) to the entire dataset obtained from 900.000 tweets; the query evaluation times (in ms) appear in Figure 2.

As a first baseline, we ran Q_0 (from Section 4); this query traverses all the chains of information propagation in our dataset, and

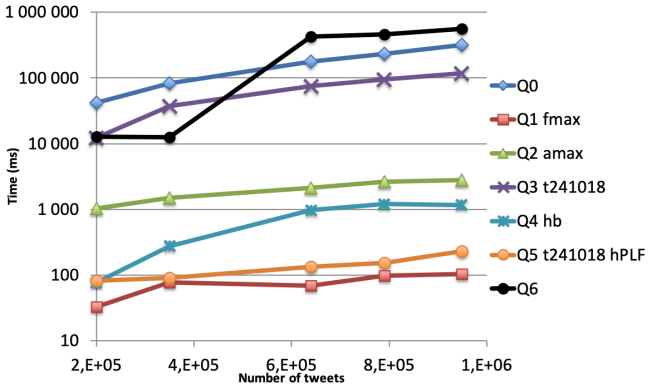

Figure 2: Query time for increasing numbers of tweets.

returns a very large answer (from 1 million to tens of millions of answers); it is not very interesting per se, but it is one of the most expensive we could think of (taking 318 seconds when run over the whole dataset), yet RDF4J is capable of handling it. A close inspection of the times shows that Q_0 scale-up is *super-linear in the number of tweets (database size)*, but it is *linear in the number of results*, which is overall good news (the query engine can hardly escape the cost of enumerating results).

Next, we ran instantiations of the query templates introduced in the previous section, as follows. First, we manually identified the fact $f_{\max}$ which is at the origin of the longest chain of Communications (retweets, comments etc.). Then, we ran Q_1 setting f to $f_{\max}$: find everyone who heard of this fact directly or indirectly; this query has 374 results. Thanks to our indexes, Q_1 with f=$f_{\max}$ ran extremely fast (from 50 ms to 99 ms, the lowest curve in the figure). This is very good news, as it shows that RDF4J is capable of restricting the exploration of "has heard of" chains to just those ending in $f_{\max}$; the query is three orders of magnitude faster than Q_0. However, using just the default indexes, Q_1 performed no better than Q_0! This is because with our modeling (recall Q_0 from Section 4, and also Figure 1), information propagation chains go "backward" (a fact is the object of a triple connecting it to a belief or communication about it, which in turn are objects (property values) of higher-order beliefs or communications etc). This is why the indexes we added to RDF4J made an important difference here.

Subsequently, we picked an agent $a_{\max}$ who authored the most tweets in the smallest (scale 1/6) fragment of our database, then ran Q_2 setting a to $a_{\max}$. Q_2 takes more time than Q_1, since it needs to look for that agent at many levels in the information propagation chains; however, it is much faster than Q_0, (also) because it returns less results. Next, we ran Q_3 for the time interval t_{241018} from 9 to 10 am on Oct 24, 2018; it is quite expensive, showing that the time selection was not very efficiently exploited by RDF4J to prune the search. The tweet dates are triple objects, thus we hoped that indexes starting with the o attribute would be exploited by RDF4J to speed up significantly these queries, or, this does not appear to be the case. RDF4J does not provide an "explain" functionality, thus for now we can only guess at an imperfect optimization strategy.

We picked the hashtag #Benalla (a name involved in a political scandal in France), which we denote *hb* in the sequel, and ran Q_4

setting h to hb; Q_4 is moderately expensive, as it also had to traverse all "has heard of" chains starting in a relevant Communication We ran Q_5 using t_{241018} and a popular hashtag h_{PLF} of the moment, PLF19[8]; combining the selections on the hashtag and on the time significantly reduced the amount of data manipulated by the engine, thus Q_5 performed almost as well as Q_1.

Finally, we ran Q_6 with h set to #PLF2019 and the time interval t_{241018}. This query becomes more expensive than Q_0 for large datasets; it combines the disadvantage of (not-so-efficient) temporal selection and the extra cost of joining the communications with the agent political party affilations, and of computing the group-by.

From the experiments, we conclude that RDF4J handles reasonably well complex queries, featuring complex path expressions (which lead to exploring potentially long information dissemination chains). Our study focused on such queries since information propagation is both interesting for data journalism scenarios and a recent, challenging SPARQL feature. While we used publicly shared Tweets, such queries can be great tools in investigative journalism scenarios based (also) on information accessible only to journalists. Modeling data in RDF aims (also) at facilitating the integration of all the data sources journalists can acquire.

6 RELATED WORK AND CONCLUSION

Modelling facts, statements and beliefs to further search and analyze them has recently gained interest in the setting of computational journalism [4], especially for fact-checking purposes, like monitoring sources, extracting claims, checking them w.r.t. reference sources and publishing obtained results (see [2] for a survey).

In this setting, we follow a *database-oriented approach* for representing timed facts, statements and beliefs using the RDF data model, and then exploiting them through SPARQL queries.

Belief databases [6], representing (i) facts and (ii) positive or negative beliefs of agents on facts or on agent beliefs, have been investigated in the setting of the relational data model and conjunctive queries. Such databases build on multi-agent epistemic logic, in particular to infer and query (a possibly infinite set of) implicit agent beliefs. We do not consider belief inference; we focus on storing information and how it propagates between agents.

Temporal databases, long known for relational data [14], have been revisited for RDF. [10] attach time points or time intervals to triples; [12] allows intervals with unknown bounds on which constraints can be set or derived using Allen's algebra. Finally, [15] models a timed RDF graph as a set of its snapshots over time. We build on the time modelling approach of [12] to attach a normalized union of time intervals to facts, beliefs and communications.

RDF has also been used in fact-checking. In [3], claims are RDF triples to be checked against a knowledge base such as DBPedia. Claim accuracy is assessed based on the (shortest and most specific) paths that connect the claim subject to its object within the knowledge base. FactMinder [7] uses an XML-RDF hybrid data model [8] to link the entities found in documents (web pages, etc.), hence in the claims made there, to these entity descriptions in a knowledge graph, in order to guide manual fact-checks. Finally, [11] proposes a complete fact-checking systems, from claim extraction to analysis and publication of the results, which notably check claims against several knowledge bases. However, it does not use a fine-grained temporal model for facts, beliefs, and their propagation.

As the next step of this work, we are currently devising a form-based GUI to help journalists writing meaningful but complex queries for their investigations.

[8] *Projet de la loi de finance* 2019, or the French state budget for 2019.

ACKNOWLEDGMENTS

This work is funded by the ANR ContentCheck project (contract ANR-15-CE23-0025).

REFERENCES

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. 1995. *Foundations of Databases*. Addison-Wesley.
[2] Sylvie Cazalens, Philippe Lamarre, Julien Leblay, Ioana Manolescu, and Xavier Tannier. 2018. A Content Management Perspective on Fact-Checking. In *WWW*. https://doi.org/10.1145/3184558.3188727
[3] Giovanni Luca Ciampaglia, Prashant Shiralkar, Luis M. Rocha, Johan Bollen, Filippo Menczer, and Alessandro Flammini. 2015. Computational fact checking from knowledge networks. *PloS one* 10, 6 (2015).
[4] Sarah Cohen, James T. Hamilton, and Fred Turner. 2011. Computational Journalism. *Commun. ACM* 54, 10 (2011). https://doi.org/10.1145/2001269.2001288
[5] Andreas Dohr, Christiane Engels, and Andreas Behrend. 2018. Algebraic Operators for Processing Sets of Temporal Intervals in Relational Databases. In *TIME*. https://doi.org/10.4230/LIPIcs.TIME.2018.11
[6] Wolfgang Gatterbauer, Magdalena Balazinska, Nodira Khoussainova, and Dan Suciu. 2009. Believe It or Not: Adding Belief Annotations to Databases. *PVLDB* 2, 1 (2009). http://www.vldb.org/pvldb/2/vldb09-115.pdf
[7] François Goasdoué, Konstantinos Karanasos, Yannis Katsis, Julien Leblay, Ioana Manolescu, and Stamatis Zampetakis. 2013. Fact checking and analyzing the web. In *SIGMOD*. https://doi.org/10.1145/2463676.2463692
[8] François Goasdoué, Konstantinos Karanasos, Yannis Katsis, Julien Leblay, Ioana Manolescu, and Stamatis Zampetakis. 2013. Growing triples on trees: an XML-RDF hybrid model for annotated documents. *VLDB J.* 22, 5 (2013). https://doi.org/10.1007/s00778-013-0321-2
[9] François Goasdoué, Ioana Manolescu, and Alexandra Roatis. 2013. Efficient query answering against dynamic RDF databases. In *EDBT*.
[10] Claudio Gutiérrez, Carlos A. Hurtado, and Alejandro A. Vaisman. 2005. Temporal RDF. In *ESWC*. https://doi.org/10.1007/11431053_7
[11] Naeemul Hassan, Gensheng Zhang, Fatma Arslan, Josue Caraballo, Damian Jimenez, Siddhant Gawsane, Shohedul Hasan, Minumol Joseph, Aaditya Kulkarni, Anil Kumar Nayak, et al. 2017. ClaimBuster: The First-ever End-to-end Fact-checking System. *Proceedings of the VLDB Endowment* 10, 7 (2017).
[12] Carlos A. Hurtado and Alejandro A. Vaisman. 2006. Reasoning with Temporal Constraints in RDF. In *Principles and Practice of Semantic Web Reasoning (PPSWR)*. https://doi.org/10.1007/11853107_12
[13] Daniel Janke, Adrian Skubella, and Steffen Staab. 2017. Evaluating SPARQL 1.1 Property Path Support. In *2nd Int'l Workshop on Benchmarking Linked Data, next to ISWC*. http://ceur-ws.org/Vol-1932/paper-04.pdf
[14] Christian S. Jensen and Richard T. Snodgrass. 1999. Temporal Data Management. *IEEE Trans. Knowl. Data Eng.* 11, 1 (1999). https://doi.org/10.1109/69.755613
[15] Jonas Tappolet and Abraham Bernstein. 2009. Applied Temporal RDF: Efficient Temporal Querying of RDF Data with SPARQL. In *ESWC*. https://doi.org/10.1007/978-3-642-02121-3_25
[16] w3c-rdf. RDF. https://www.w3.org/TR/rdf11-concepts. (????).
[17] w3c-sparql. SPARQL 1.1 Query Language. https://www.w3.org/TR/sparql11-query. (????).