Towards Lightweight and Robust Large Scale Emergent Knowledge Processing

Towards Lightweight and Robust Large Scale Emergent Knowledge Processing

Vít Nováček and Stefan Decker

DERI, National University of Ireland, Galway
IDA Business Park, Galway, Ireland
E-mail: vit.novacek@deri.org

Abstract. We present a lightweight framework for processing uncertain emergent knowledge that comes from multiple resources with varying relevance. The framework is essentially RDF-compatible, but allows also for direct representation of contextual features (e.g., provenance). We support soft integration and robust querying of the represented content based on well-founded notions of aggregation, similarity and ranking. A proof-of-concept implementation is presented and evaluated within large scale knowledge-based search in life science articles.

1 Introduction

On the Semantic Web, we often have to be able to represent and integrate statements coming from many resources with varying relevance in a bottom-up, emergent manner. Moreover, the statements themselves may be noisy and uncertain (e.g., inconsistent, potentially incorrect or having an explicit certainty degree). This is especially pertinent to a use case that has largely motivated our work – search for expressive statements instead of mere keywords in life science articles. More specifically, we want to allow life scientists to search for statements like *acute granulocytic leukemia : NOT is a : T-cell leukemia*, or *? : part of : immunization*. The former query is supposed to confirm whether acute granulocytic leukemia is different from T-cell leukemia by checking for similar statements in publications. Also, the result should provide articles supporting the query statement. The latter query is supposed to return everything that can be a part of the immunization process, plus any related statements and links to articles relevant to them.

Manual annotation of the publication knowledge to be exposed for such search is practically impossible in large scale. However, one can extract the knowledge from the article texts by ontology learning techniques [5] and link it to existing domain ontologies in order to increase the expressivity of the rather shallow extracted content. Such an approach still poses a couple of challenges, though: (i) The representation framework of choice should support uncertainty, as the extracted knowledge usually comes with explicit certainty degrees [5]. (ii) The representation should also straightforwardly support contextual features, namely at least provenance of statements (to link them to the respective source articles). (iii) Robust aggregation of the emergent statements based on

relevance of respective resources is necessary, since we have to integrate noisy extracted knowledge with presumably more accurate manually designed domain ontologies. (iv) The processed knowledge has to be accessible by means of intuitive (i.e., nearly natural language) query answering, since we target users with no or little technical expertise. The query evaluation should also be approximate in order to provide useful answers even for queries partially evaluated on a lot of potentially noisy data.

Approaches like [13, 11, 4, 7, 8, 15, 1, 12, 2] provide particular solutions apt for coping with the challenges separately, however, to the best of our knowledge there is no off-the-shelf framework tackling all of them at once on a well-founded basis. The main contribution of this paper is two-fold. Firstly, we introduce a general notion of similarity-based lightweight semantics, integrally addressing all the above challenges (Section 2). Secondly, Section 3 presents a particular application of the general framework to knowledge-based search in life science articles. Promising results of an evaluation performed with domain experts are reported in Section 4. We discuss related approaches and conclude the paper in Sections 5 and 6, respectively.

2 General Framework

In the following, we first informally outline the essential notions of the proposed framework and briefly comment on their interplay. The outline is then expanded by more rigorous and explanatory subsections 2.1 (entities and their grounding) and 2.2 (knowledge bases, aggregation and query answering).

Central to our framework is a notion of *entities* that represent real and/or conceivable objects using unique identifiers and sets of positive or negative uncertain *relations* to other entities. To give an example, let us consider the d, a, c, t identifiers representing the `dog`, `animal`, `cat` concepts and the `type` relationship, respectively. The `dog` entity can be further specified by binary relations $t(d, a)$ and $t(d, c)$ with a positive and negative certainty, respectively, meaning dogs are animals different from cats. To support contextual features of entity relationships (e.g., provenance or time-stamp), the relations may generally have arbitrary arities. A direct correspondence of sets of n-ary certainty-valued relations to n-dimensional tensors (generalisations of the scalar, vector and matrix notions) provides for a compact computational representation of entities. An entity E is then represented as $(e, \mathbf{E})$, i.e., its unique identifier and the respective compact representation of uncertain relations to other entities. To ensure accessibility for lay users, we link the somewhat abstract representation to corresponding natural language referents via a set of *grounding* functions. These may map, for instance, the `dog` entity to a preferred "dog" expression with a high certainty, but also to alternative synonyms like "doggy" or "hound", perhaps with a bit lower certainty. The other way around, a grounding would map the "mutt" word to the `dog` entity in the lexical domain of animals, but to a completely different entity in the domain of, say, humans. Thus the grounding provides a two-way bridge between the lexical (human-centric) and computational (machine-centric) aspects of the proposed lightweight semantics. The bridge is particularly important when

answering user queries—formulated as mostly natural language statements—by means of a query answering service dealing with abstract entity representations.

Building on the compact computational representation of entities, we introduce the *aggregation* and *querying* services in order to tackle the remaining challenges specified in the introduction. Entity aggregation employs linear combinations that naturally model merging of possibly conflicting statements coming from sources with varying relevance. For instance, imagine a statement that dogs eat meat, coming from a highly relevant source, and an opposite, yet relatively irrelevant statement (vegetarian dogs actually exist, however, the respective rather exceptional sources are presumably less relevant). The sum of the corresponding representations, weighed by the relative source relevance, will result in a claim that dogs eat meat with a positive, but slightly lower certainty (as the knowledge from more relevant source prevails in the aggregation).

Query answering makes use of two notions of entity similarity. Let us imagine entities of `dog` and `cow`, eating and not eating meat, respectively. Evaluation of a query for meat-eating animals first checks for entities fitting to the context of the query, i.e., being animals and linked by an "eat" relation to meat. Both `dog` and `cow` entities fit the query within this coarse-grained approximation of similarity. A finer grained notion of similarity, taking the certainty degrees into account, can be naturally coined as dual to a distance defined on the set of entity representations. Utilising this type of similarity results into meat-eating `dog` being a much more certain answer to the query than `cow`, which is an animal, but does not eat meat. In more complex cases, we also sort the query results according to their relevance employing a generalised IR measure based on numbers of outgoing and incoming relations among stored entities.

2.1 Entities and Their Grounding

Entities First we have to formalise certainty degrees, for which we use $\mathbb{R}$, i.e., real numbers. Positive and negative entity relationships are to be associated with positive and negative certainty values, respectively. 0 is of special importance, expressing absolute lack of certainty. We do not impose any restrictions on the range of certainty values, however, particular implementations may restrict the range to any set isomorphic with $\mathbb{R}$. A convenient variant (used throughout the paper) is $(-1, 1)$, which makes the certainty values compatible either with a recent approach to trust representation in RDF [7], or with general fuzzy and probabilistic formalisms (after transforming negative certainties into negative fuzzy/probabilistic statements). Openness of the certainty intervals reflects the fact that nothing is absolutely certain in emergent settings. Implementations may relax the assumption, though, and use a more traditional $[-1, 1]$ interval.

Moving on to defining entities themselves, let I be a non-empty countable set of unique entity identifiers (e.g., integer numbers or URIs) and $n \in \mathbb{N}_0$ a so called rank of an entity. Rank expresses the maximal arity of relations associated to an entity. An entity with an identifier $c \in I$ and rank $n = 0$ can be written down as a tuple (c, d), meaning that c merely exists (or does not exist) with the certainty d. In practice, more expressive entities with a rank $n > 0$ are required, though. An entity E with an identifier $c \in I$ and rank $n > 0$ can be

written down as a set of tuples in the form $(c_1(c, c_2, \ldots, c_n), d)$, where $c_x \in I$ for $x \in \{1, \ldots, n\}$, $d \in \mathbb{R}$. The tuple elements encode a c_1 relation between c and other entities $c_2, \ldots, c_n$, and the relation's certainty, respectively. It is required that $\{(c_{i,1}, c_{i,2}, \ldots, c_{i,n}) | (c_{i,1}(c, c_{i,2}, \ldots, c_{i,n}), d_i) \in E\} = I^n$, meaning that the relations iterate through all possible identifier combinations. However, realistic entities are obviously associated only with a relatively small finite number of relations with a non-zero certainty. We distinguish a special zero-entity (denoted by $\mathcal{O}$ in the following text), which has all certainty degrees equal to 0, thus representing an absolutely uncertain object. $\mathcal{O}$ can be used namely to represent relations with an arity lower than n by n-ary ones (filling in the respective superfluous arguments as shown in Example 1).

Conceiving an entity as a set of relations associated with the entity's identifier is pretty intuitive. Such a form is, nonetheless, quite awkward for treating entities as compact objects. A more compact representation is possible using a direct correspondence between the sets of entity relations and the mathematical structure of tensor (multi-dimensional generalisation of the scalar, vector and matrix notions, which are tensors of ranks $0, 1, 2$, respectively). Using the tensor notation, an entity E with an identifier e and rank n can be represented as a tuple $E \equiv (e, \mathbf{E})$, where $e \in I$ and $\mathbf{E} \in T$ (a set of all tensors of rank n on the field $\mathbb{R}$). A tensor entity representation $(e, \mathbf{E})$ corresponds to a set of relation-degree tuples $\{(c_1(e, c_2, \ldots, c_n), \mathbf{E}_{c_1, c_2, \ldots, c_n}) | (c_1, c_2, \ldots, c_n) \in I^n\}$, where $\mathbf{E}_{c_1, c_2, \ldots, c_n}$ is the element of $\mathbf{E}$ with the respective indices.

Example 1. Here we illustrate the correspondence between the two entity notations (rank 2 is used to facilitate the presentation; higher ranks are direct generalisations of this case). Assuming `B` standing for `http://ex.org`, let $I = \{$ `B#null`, `B#type`, `B#cat`, `B#animal`, `B#eatsMeat` $\}$ abbreviated as $I = \{\bot, t, c, a, e\}$, respectively. `B#null` (or $\bot$) is an identifier of the zero entity $\mathcal{O}$. The `cat` entity E of rank 2 with an identifier c can be described by the following set of relation-degree tuples (omitting the ones with zero degrees): $\{(t(c, a), 0.99), (e(c, \bot), 0.99)\}$. The binary *type* relation says that cats are a type of animal, while the unary *eatsMeat* relation says that cats eat meat (both relations have a positive certainty). The respective tensor representation is $E = (c, \mathbf{E})$, where $\mathbf{E}$ is the following matrix:

	a	$\bot$
t	0.99	0
e	0	0.99

Note that we omit (also in the following examples) rows and columns with all degrees equal to zero when they are not required for facilitating the presentation.

Grounding Let L be a non-empty countable set of language expressions (e.g., words upon an alphabet). Entities are grounded in a language via a so called entity grounding mapping $g_{ind} : I \to (L \to \mathbb{R})$. $g_{ind}(x)$ are total functions that assign certainty values to each element from L. The functions support the synonymy and antonymy lexical phenomenons via positive and negative certainty assignments, respectively. Going the other way around, language expressions are

mapped to entity identifiers via a so called unique entity identifier assignment $g_{lan} : L \times I \to I$. The first argument of g_{lan} is an expression to be mapped to an entity identifier, while the second argument is a so called lexical domain – an entity providing a disambiguation context, catering for correct resolution of homonymous terms. Eventually, we need to ground the dimensions of the tensor representation (or argument positions in the relation-degree notation) to concept identifiers. This is done using a so called dimension grounding mapping $g_{dim} : \{1, \ldots, n\} \to I$, assigning an entity identifier to each entity index dimension.

Example 2. Assuming `B` standing for `http://ex.org`, let us extend the I set from Example 1 to $I = \{$ `B#null`, `B#type`, `B#cat`, `B#animal`, `B#eatsMeat`, `B#isVeggie`, `B#predicate`, `B#object`, `B#human`, `B#gld`, `B#sissy`$\}$. Furthermore, let $L = \{$*null entity, type, is a, cat, animal, pussycat, eatsMeat, isVeggie, meatEating, predicate, object, human, general lexical domain*$\}$. Let us consider functions $\mu_1, \ldots, \mu_{11}$ assigned by a sample entity grounding g_{ind} to the elements of I (in the order given in the beginning of the example). All the functions assign 0 to most elements of L, with the following exceptions: (i) $\mu_1(x) = 0.99$ for $x \in \{$*null entity*$\}$; (ii) $\mu_2(x) = 0.99$ for $x \in \{$*type, is a*$\}$; (iii) $\mu_3(x) = 0.99$ for $x \in \{$*cat*$\}$, $\mu_3(x) = 0.8$ for $x \in \{$*pussycat*$\}$; (iv) $\mu_4(x) = 0.99$ for $x \in \{$*animal*$\}$; (v) $\mu_5(x) = 0.99$ for $x \in \{$*eatsMeat, meatEating*$\}$, $\mu_5(x) = -0.99$ for $x \in \{$*isVeggie*$\}$; (vi) $\mu_6(x) = 0.99$ for $x \in \{$*isVeggie*$\}$, $\mu_6(x) = -0.99$ for $x \in \{$*eatsMeat, meatEating*$\}$; (vii) $\mu_7(x) = 0.99$ for $x \in \{$*predicate*$\}$; (viii) $\mu_8(x) = 0.99$ for $x \in \{$*object*$\}$; (ix) $\mu_9(x) = 0.99$ for $x \in \{$*human*$\}$; (x) $\mu_{10}(x) = 0.99$ for $x \in \{$*general lexical domain*$\}$; (xi) $\mu_{11}(x) = 0.99$ for $x \in \{$*pussycat*$\}$. Regarding the unique identifier assignment, the only ambiguous lexical expression is *pussycat*: $g_{lan}($*pussycat*, `B#human`$) =$ `B#sissy`, $g_{lan}($*pussycat*, `B#animal`$) =$ `B#cat`. All the other lexical expressions have obvious mappings to identifiers under the *general lexical domain*. Eventually, the dimension mapping g_{dim} can be defined as $g_{dim}(1) =$ `B#predicate`, $g_{dim}(2) =$ `B#object`. This roughly follows the RDF terminology in the sense that the first and second dimension of the tensor representation (i.e., the matrix row and column) correspond to predicate and object identifiers, respectively.

2.2 Knowledge Bases, Aggregation and Query Answering

Knowledge base of rank n is a tuple $(\mathcal{E}, n, I, L, \mathcal{G})$. I, L are the sets of entity identifiers and language expressions as introduced before. $\mathcal{E}$ is a set of entities $(e, \mathbf{E})$ such that $e \in I$ and $\mathbf{E} \in T$, where T is a set of all tensors of rank n defined on $\mathbb{R}$. $\mathcal{G}$ is a set of particular grounding mappings $g_{ind}, g_{lan}, g_{dim}$. Furthermore, a knowledge base must satisfy certain restrictions. Let $ind : \mathcal{E} \to I, ind((e, \mathbf{E})) = e$, $rep : \mathcal{E} \to T, rep((e, \mathbf{E})) = \mathbf{E}$ be projections mapping entities in a knowledge base to their identifiers and tensor representations, respectively. Then it is required that $ind(E) = ind(F)$ if and only if $rep(E) = rep(F)$ for every $E, F \in \mathcal{E}$ (consequently, $E = F$ iff $ind(E) = ind(F)$ or $rep(E) = rep(F)$). Also, the ind projection has to be a bijection. Thus, every entity has a unique identifier and each identifier maps to an entity in a particular knowledge base.

As knowledge is often inherently context-dependent, we have to introduce an appropriate notion of context in our representation. We do so using so called

contextual scopes, which are non-empty sets $S \subseteq I^n$ for a knowledge base $(\mathcal{E}, n, I, L, \mathcal{G})$. Briefly put, contextual scopes divide $\mathcal{E}$ into classes of entities associated with particular relations of non-zero certainty in direct correspondence to the elements of S. Each non-zero entity fits into at least one contextual scope. We refer to the minimal contextual scope fully covering a non-zero entity $\mathcal{E}$ by $scp : \mathcal{E} \setminus \mathcal{O} \to 2^{I^n}, scp(E) = \{(v_1, \ldots, v_n) | (v_1, \ldots, v_n) \in I^n \wedge rep(E)_{v_1,\ldots,v_n} \neq 0\}$. It is simply a set of indices of all non-zero elements in the respective entity representation. We define fitness of a non-zero entity E w.r.t. a general contextual scope S as $fit : \mathcal{E} \setminus \mathcal{O} \times 2^{I^n} \to [0, 1], fit(E, S) = max(\frac{|scp(E) \cap S|}{|S|}, \frac{|scp(E) \cap S|}{|scp(E)|})$. Maximal fit of 1 is achieved if either all non-zero element indices of the entity are covered by the contextual scope, or if all elements of the contextual scope are covered by the entity's non-zero elements. Minimal fit of 0 is achieved if no index of any non-zero entity element appears in the contextual scope.

Example 3. In order to illustrate practical treatment of contextual scopes, let us extend the I set from previous examples to $I =$ { `B#null`, `B#type`, `B#cat`, `B#animal`, `B#eatsMeat`, `B#dog`, `B#feline`, `B#canine`}, abbreviated as $I = \{\bot, t, c, a, e, d, f, cn\}$, respectively. Let E and F be `cat` and `dog` entities, such that

$$rep(E) = \begin{array}{c|cccc} & a & f & cn & \bot \\ \hline \mathrm{t} & 0.99 & 0.99 & 0 & 0 \\ \mathrm{e} & 0 & 0 & 0 & 0.99 \end{array} \text{ and } rep(F) = \begin{array}{c|cccc} & a & f & cn & \bot \\ \hline \mathrm{t} & 0.99 & 0 & 0.99 & 0 \\ \mathrm{e} & 0 & 0 & 0 & 0.99 \end{array}.$$

Contextual scopes corresponding to felines and canines can be defined as $S_1 = \{(t, f)\}, S_2 = \{(t, cn)\}$, respectively. Similarly, feline and canine animals correspond to contextual scopes $S_3 = \{(t, a), (t, f)\}, S_4 = \{(t, a), (t, cn)\}$. Consistently with common sense, $fit(E, S_1) = fit(F, S_2) = fit(E, S_3) = fit(F, S_4) = 1$, meaning that cats are in the context of felines and feline animals (similarly for canine dogs). Also, $fit(E, S_2) = fit(F, S_1) = 0$ meaning that cats do not fit in the context of canines and vice versa for dogs. However, $fit(E, S_4) = fit(F, S_3) = 0.5$, meaning that cats share certain properties (i.e., relations) with canine animals (i.e., being a type of animal), and vice versa for dogs.

In the following, we will need an auxiliary operator for entity trimming according to a contextual scope. It is defined as $\tau : T \times 2^{I^n} \to T, \tau(\mathbf{E}, S) = \mathbf{F}$, where $\mathbf{F}_{i_1,\ldots,i_n} = \mathbf{E}_{i_1,\ldots,i_n}$ for all $(i_1, \ldots, i_n) \in S$, otherwise $\mathbf{F}_{i_1,\ldots,i_n} = 0$. The trimming cuts all the relations not "belonging" to a contextual scope off an entity, rendering their certainty zero. Apparently, $\tau(rep(E), S) = rep(E)$ iff $scp(E) \subseteq S$. The operator is to be used when one needs to focus only on particular features of entities within their computational processing (e.g., aggregation or querying).

Entity Aggregation Let $+, \cdot$ be operations of vector addition and scalar multiplication defined on T and $\mathbb{R}$ (e.g., element-wise tensor addition and scalar multiplication as a generalisation of the respective matrix operations). Then T forms a vector space and as such can provide a natural framework for weighed entity aggregation by means of linear combinations. An aggregation of entities $E_1, \ldots, E_k$ with rank n is a function $agg : 2^T \to T$ operating on the respective

tensor representations:

$$agg(\{rep(E_1), \ldots, rep(E_k)\}) = \sum_{v \in V} \sum_{j \in J} r_{v,j} \tau(rep(E_j), \{v\}),$$

where $V = \{(i_1, \ldots, i_n) | \exists x.x \in \{1, \ldots, k\} \wedge rep(E_x)_{i_1,\ldots,i_n} \neq 0\}$, $J = \{x | x \in \{1, \ldots, k\} \wedge rep(E_x)_{v_1,\ldots,v_n} \neq 0\}$, such that $(v_1, \ldots, v_n) = v$. $r_{v,j} \in \mathbb{R}_0^+$ are weights reflecting the relevance of the particular summation elements and τ is the entity trimming operator defined before. The generic aggregation definition flexibly covers intuitively applicable aggregation mechanisms, as shown in the following example.

Example 4. Assuming the I set from the previous examples, imagine two different `dog` entity representations $rep(E_1), rep(E_2)$, such that

$$rep(E_1) = \begin{array}{c|c|c} & a & \bot \\ \hline \mathrm{t} & 0.99 & 0 \\ \hline \mathrm{e} & 0 & -0.5 \end{array} \text{ and } rep(E_2) = \begin{array}{c|c|c} & a & \bot \\ \hline \mathrm{t} & 0.99 & 0 \\ \hline \mathrm{e} & 0 & 0.99 \end{array}.$$

Let the entity representations come from sources with relevance weights 0.2 and 1, respectively (the source conceiving a dog as a kind of vegetarian having much lower, although non-zero relevance). $agg(\{rep(E_1), rep(E_2)\})$ then expands as:

$$r_{(a,t),1} \begin{array}{c|c|c} & a & \bot \\ \hline \mathrm{t} & 0.99 & 0 \\ \hline \mathrm{e} & 0 & 0 \end{array} + r_{(a,t),2} \begin{array}{c|c|c} & a & \bot \\ \hline \mathrm{t} & 0.99 & 0 \\ \hline \mathrm{e} & 0 & 0 \end{array} + r_{(e,\bot),1} \begin{array}{c|c|c} & a & \bot \\ \hline \mathrm{t} & 0 & 0 \\ \hline \mathrm{e} & 0 & -0.5 \end{array} + r_{(e,\bot),2} \begin{array}{c|c|c} & a & \bot \\ \hline \mathrm{t} & 0 & 0 \\ \hline \mathrm{e} & 0 & 0.99 \end{array}.$$

Various mechanisms of aggregation can be achieved by setting the $r_{(a,t),1}, r_{(a,t),2}$, $r_{(e,\bot),1}$, $r_{(e,\bot),2}$ weights accordingly. E.g., $r_{(a,t),1} = r_{(a,t),2} = 0.5, r_{(e,\bot),1} = 0.2/1.2$, $r_{(e,\bot),2} = 1/1.2$ keeps equal elements unchanged, however, computes weighted mean for conflicting certainty values with the source relevances as particular weights, thus letting the statement from a more relevant source prevail.

Query Answering We support soft anytime retrieval of entities from knowledge bases according to their *similarity* to so called primitive queries Q, with the results sorted by their *relevance*. Primitive queries are simply entities with an unknown identifier (i.e., variable). First approximation of the similarity is the fitness $fit(E, scp(Q))$. Assuming a knowledge base $(\mathcal{E}, n, I, L, \mathcal{G})$, entities $E \in \mathcal{E}$ with $fit(E, scp(Q)) > 0$ are plausible (possibly partial) answers for the query Q.

A more fine grained notion of similarity can be naturally defined using a metric $d : T^2 \to \mathbb{R}$ on the set T of entity representations. d can be any function satisfying the following properties for all $\mathbf{E}, \mathbf{F}, \mathbf{G} \in T$: (i) positive definiteness – $d(\mathbf{E}, \mathbf{F}) \geq 0$, $d(\mathbf{E}, \mathbf{F}) = 0$ if and only if $\mathbf{E} = \mathbf{F}$; (ii) symmetry – $d(\mathbf{E}, \mathbf{F}) = d(\mathbf{F}, \mathbf{E})$; (iii) triangle inequality – $d(\mathbf{E}, \mathbf{G}) \leq d(\mathbf{E}, \mathbf{F}) + d(\mathbf{F}, \mathbf{G})$. Similarity is conceptually dual to distance (i.e., metric). Therefore we can define similarity of entities $E, F \in \mathcal{E}$ as a a function $sim : \mathcal{E}^2 \to (0, 1], sim(E, F) = \frac{1}{1+d(rep(E),rep(F))}$. The duality of sim and d is ensured by their apparent inverse proportionality. Moreover, sim has the following intuitively expected properties: $sim(E, E) = 1$ and $\lim_{x \to \infty} sim(E, F) = 0$, where $x = d(rep(E), rep(F))$.

Apart of similarity of candidate answers to queries, we establish the notion of entity relevance, which can be effectively used for ranking query results. Informally, relevance of an entity E in our framework is given by the number and certainty of relations that are associated to it, but also by the number and certainty of relations that reference it. Such a measure tells us how important E is w.r.t. determining the meaning of other entities. This is directly related to the hubs and authorities algorithm designed for ranking of web pages [9]. We only need to generalise it to support n-ary links with arbitrarily weighed relations and argument positions. The generalised hub measure of entities in a knowledge base $(\mathcal{E}, n, I, L, \mathcal{G})$ is recursively defined as $h : \mathcal{E} \to \mathbb{R}_0^+$ such that:

$$h(E) = \sum_{(u_1,\ldots,u_n)\in scp(E)} |rep(E)_{u_1,\ldots,u_n}| w_{arg}(1) w_{rel}(u_1) \sum_{k=2}^{n} w_{arg}(k) a(F),$$

where $F = ind^{-1}(u_k)$ is the entity referenced in the respective relation. Similarly, the generalised authority measure is defined as $a : \mathcal{E} \to \mathbb{R}_0^+$, such that:

$$a(E) = \sum_{F\in R} \sum_{(u_1,\ldots,u_n)\in V} |rep(F)_{u_1,\ldots,u_n}| w_{arg}(1) w_{rel}(u_1) h(F) \sum_{x\in Y} w_{arg}(x),$$

where $R = \{G | \exists G.rep(G)_{u_1,\ldots,u_n} \neq 0 \wedge \bigvee_{i=1}^{n} ind(E) = u_i\}$ is a set of all entities referencing E, $V = \{(v_1,\ldots,v_n) | rep(F)_{v_1,\ldots,v_n} \neq 0 \wedge ind(E) \in \{v_1,\ldots,v_n\}\}$ and $Y = \{y | y \in \{u_1,\ldots,u_n\} \wedge y = ind(E)\}$. $w_{rel} : I \to \mathbb{R}_0^+$ and $w_{arg} : \{1,\ldots,n\} \to \mathbb{R}_0^+$ are weights of particular relations and relation argument positions (generally including also the "zeroth" argument position, i.e., the relation identifier itself). Using the generalised measures, we can compute the hub and authority scores for entities $E \in \mathcal{E}$ with the iterative algorithm given in [9] (normalising the scores in each iteration to ensure convergence). The relevance of an entity E is then defined as $rel : \mathcal{E} \to \mathbb{R}_0^+, rel(E) = m(h(E), a(E))$, where $m : \mathbb{R}^2 \to \mathbb{R}$ is any aggregation function such that for all $x, y \in \mathbb{R}_0^+$, $min(x, y) \leq m(x, y) \leq max(x, y)$. Examples are min, max, or an arithmetic mean.

Having introduced all the necessary notions, we can finally specify the set of answers to a query $Q \in \mathcal{E}$ w.r.t. a knowledge base $(\mathcal{E}, n, I, L, \mathcal{G})$ as a function $ans : \mathcal{E} \to 2^{\mathcal{E}}$ such that $ans(Q) = \{s_1 A_1, \ldots, s_k A_k\}$. $A_1, \ldots, A_k \in \mathcal{E}$ and $s_i = sim(\tau(rep(A_i),\ scp(Q)), rep(Q))$ for $i \in \{1,\ldots,k\}$. Note that to simplify the notation, we assume that $sA = s(a, \mathbf{A}) = (a, s\mathbf{A})$ for a multiplication of an entity (i.e., an identifier-tensor tuple) by a scalar value. It is required that $fit(A_1,\ scp(Q)) \geq \cdots \geq fit(A_k, scp(Q)) > 0$. Moreover, every sequence $s_{i_1} A_{i_1}, \ldots, s_{i_l} A_{i_l}$ such that $i_1, \ldots,\ i_l \in \{1,\ldots,k\}$, $i_1 \leq \cdots \leq i_l$ and $fit(A_{i_1}, scp(Q)) = \cdots = fit(A_{i_l}, scp(Q))$ must be lexicographically ordered according to the respective $(s_x,\ rel(A_x))$ measures. Thus, $ans(Q)$ is a set of entities from $\mathcal{E}$ multiplied by their actual similarity to Q, taking only the minimal contextual scope covered by Q—i.e., $scp(Q)$—into account, though. The answers also must be ordered first regarding the fitness measure w.r.t. $scp(Q)$, then according to their similarity to the query (in the query's context), and finally according to their relevance.

Example 5. In the following, we employ similarity based on particular metric $d(\mathbf{E},\mathbf{F}) = \frac{1}{|V|}\sum_{(u_1,\ldots,u_n)\in V} |\mathbf{E}_{u_1,\ldots,u_n} - \mathbf{F}_{u_1,\ldots,u_n}|$, where $V = scp(E) \cup scp(F)$. The metric simply sums up absolute values of differences across the representation indices referring to a non-zero value in $\mathbf{E}$ or in $\mathbf{F}$, normalising the result by the size of the summation range. The respective similarity $\frac{1}{1+d(rep(E),rep(F))}$ is essentially a very simple formalisation of the contrast model [14] (more sophisticated alternatives may, e.g., put specific weights on particular elements within the metric computation to reflect intensity and context in the sense of [14]).

Consider now the particular `cat` and `dog` entities E and F as given in Example 3 and a query Q asking for canine animals. The set of answers $ans(Q)$ then equals $\{A_1, A_2\}$, where

$$rep(Q) = \begin{array}{c|cc} & a & cn \\ \hline \mathrm{t} & 0.99 & 0.99 \end{array},\ A_1 = (d,\ \begin{array}{c|ccc} & a & cn & \bot \\ \hline \mathrm{t} & 0.99 & 0.99 & 0 \\ \mathrm{e} & 0 & 0 & 0.99 \end{array}),\ A_2 = (c,\ \begin{array}{c|ccc} & a & f & \bot \\ \hline \mathrm{t} & 0.497 & 0.497 & 0 \\ \mathrm{e} & 0 & 0 & 0.497 \end{array}).$$

When aggregating the hub and authority values using the *max* function and setting all weights to 1, except for the unary e relation weight set to 0, the relevance of the `cat, dog, animal, feline, canine` entities is $0.5, 0.5, 0.5, 0.25, 0.25$, respectively. However, apparently we do not need relevance in this simple example, as the fitness and similarity are enough to sort the results.

Raw sets of answers might not be particularly interesting for users in practical query-answering application scenarios. Therefore the implementations of the proposed framework may present just the corresponding ordered list of the answer entity identifiers (or their appropriate lexical labels). To provide additional information, such results may be associated with an aggregation of the respective fitness and similarity values, such as in the following: $\{dog : 1, cat : 0.5\}$ (using the corresponding lexical labels and *min* for the aggregation). Such an answer contains all the intuitively expected information – dogs are canine animals, while cats are animals, however, not canines. Therefore cats are present in the result, too, but with a lower explicit relevance.

3 Particular Implementation and Deployment

We have implemented a proof-of-concept prototype of the theoretical principles introduced so far, called EUREEKA (may be read as an acronym for *Efficient, Universal, Reasonable and Easy-to-use Emergent Knowledge Acquisition*). As mentioned in Section 1, the development and current deployment of the prototype has been motivated by the use case of knowledge-based search in life science articles. In order to realise this in an economically feasible way, we have to *extract* the respective knowledge from the texts, *represent* it in an appropriate manner, *integrate* it and *expose* it to the users in a robust and meaningful way. To address these tasks, we have recently delivered CORAAL (cf. `http://coraal.deri.ie:8080/coraal/`), which is a comprehensive life science publication search engine deployed on the data provided by Elsevier within their Grand Challenge contest (cf. `http://www.elseviergrandchallenge.com/`). EUREEKA forms the

engine's crucial back-end part, catering for the *representation*, *integration* and *exposure* tasks, thus enabling the knowledge-based search functionalities.

For the initial knowledge extraction in CORAAL, we used a NLP-based heuristics stemming from [10, 16] in order to process chunk-parsed texts into subject-predicate-object-score quads. The scores were derived from absolute and document frequencies of subject/object/predicate terms aggregated with subject/object co-occurrence measures. If a relation's score is not available for any reason (e.g., when importing legacy knowledge from crisp resources instead of extracting it from text), we simply set it to 1 (or -1) in the implementation. The extracted quads encoded three major types of ontological relations between concepts: (i) taxonomical—*type* or *same as*—relationships; (ii) concept difference (i.e., negative *type* relationships); and (iii) "facet" relations derived from verb frames in the input texts (e.g., *has part*, *involves* or *occurs in*). We imposed a taxonomy on the latter, considering the head verb of the respective phrase as a more generic relation (e.g., *involves expression of* was assumed to be a type of *involves*). Also, several artificial relation types were introduced to specify the semantics of some most frequent relations. Namely, (positive) *type* was considered transitive and anti-symmetric, and *same as* is set transitive and symmetric. Similarly, *part of* was assumed transitive and being inverse of *has part*.

After the initial knowledge extraction in CORAAL, EUREEKA comes into play in order to integrate the emergent statements, link them to precise domain thesauri and expose them to users via intuitive approximate querying. The remainder of this section outlines the most important features of the EUREEKA implementation that enabled its efficient deployment in CORAAL.

3.1 Relational Storage of Knowledge Bases

For low-level storage, we chose to employ a relational database, since it is a state of the art technology for scalable data management, which in addition allows for quite straightforward implementation of our framework. Considering a knowledge base $(\mathcal{E}, n, I, L, \mathcal{G})$, we can represent $\mathcal{G}, \mathcal{E}$ as two relational tables `grounding` and `entities`. The former serves for mapping of natural language inputs to unique internal identifiers and vice versa, while the latter supports the entity storage and operations according to Section 2.2.

The `grounding` table consists of the columns `lemma`, `identifier`, `scope`, `certainty` of `VARCHAR`, `INTEGER`, `INTEGER`, `FLOAT` types, and of indices `ls = (lemma,scope)`, `ic = (identifier,certainty)`. The sets I, L are given by the `identifier`, `lemma` columns, respectively (we store terms in their lemmatised, i.e., canonical lexical form). The table indices allow for a convenient and efficient implementation of the g_{ind} and g_{lan} mappings in $\mathcal{G}$ via the respective `SELECT` operations. Note that inclusion of `certainty` into `ic` allows for direct access to, e.g., lexical expressions attached to an identifier with maximal positive or negative certainty. This retrieves an entity's preferred synonyms or antonyms, respectively. To save space, we use integer entity identifiers, however, these can be directly mapped to a respective URI scheme if required by an application. In the current deployment of EUREEKA, the `grounding` table is filled in according to

the terms (and possibly their synonyms) coming from two sources: (i) EMTREE and NCI life science thesauri (cf. `http://www.embase.com/emtree/`, `http://nciterms.nci.nih.gov`, respectively); (ii) statements extracted from the Elsevier life science articles. The only lexical domains we currently distinguish are those corresponding to auxiliary *relation* and *generic* (i.e., non-relation) entities.

The `entities` table stores particular entities. These can be expressed as sets of relations associated with the respective certainty, as introduced in the beginning of Section 2. Such a notation can be directly transformed into a set of rows in a relational database table. However, a direct transformation of n-ary relations may be inadequate if n is not set firmly and/or if we have to store many relations with arities lower than n. These situations lead either to problems with maintenance, or to wasted space in the table.

Nevertheless, we process *subject-predicate-object* triples, all of which have a *provenance* (either an article, or a domain thesaurus), so we can explicitly represent the respective ternary relations in the `entities` table without wasting any space. For the representation of possible additional relation arities (such as location or other types of context), we associate each row in the `entities` table with a unique statement identifier `stid`. Then we can represent, e.g., quaternary relations in the form *bindsTo(drugX,proteinY,docID,bindingSiteZ)* as a ternary relation *at(*$stid_i$*,bindingSiteZ,docID)*, assuming *at* grounding the fourth "location" argument. $stid_i$ is a statement identifier of *bindsTo(drugX,proteinY,docID)*. This procedure can be obviously generalised to arbitrary arities.

Following the design considerations, the `entities` table consists of columns `stid`, `predicate`, `subject`, `object`, `provenance`, `certainty`. All columns are `INTEGER`, except for the latter one, which is `FLOAT`. Provenance is modelled as a special entity linked to an article ID, title, text, etc. Besides the primary key `(stid)`, indices on `(subject,predicate,object)`, `(subject,object)`, `(object,predicate)`, `(predicate,object)` are defined. Explicit querying for provenance is not necessary in our use case (we only need to retrieve provenance as a function of particular statements), therefore we do not maintain respective indices. An entity $E \in \mathcal{E}$ directly corresponds to rows with `subject` equal to $ind(E)$ and to rows with `subject` referencing the respective `stid` values. The corresponding tensor entity representation $rep(E)$ can be directly constructed from the content of the rows as a multidimensional array of floats, with the necessary tensor-based operations implemented on the array data structure.

Regarding the particular implementation of entity ranking, we employ $w_{arg} = 1$ for `predicate, object` and $w_{arg} = 0$ for all other arguments (results in rather traditional binary hub and authority score computation). The relation weighing function makes use of the frequency of particular relation instances (i.e., number of statements having the relation identifier as a predicate): $w_{rel}(r) = \frac{1}{ln(e+f(r)-L)}$ if $f(r) \geq L$, $w_{rel}(r) = 0$ otherwise, where $f(r)$ is the absolute frequency of r. Relations with frequency below the limit are not taken into account at all. The heuristic weighing is designed to reduce the influence of very frequent, but rather generic relations (e.g., *type*), in favour of less frequent, but potentially significant ones (e.g., *involved in*). The L limit (set to 25 in the current implementation)

serves for cutting accidental noise off the result. For the aggregation of the $h(E), a(E)$ hub and authority scores into $rel(E)$, we use the arithmetic mean.

3.2 Aggregating and Accessing the Emergent Knowledge

EUREEKA can smoothly merge facts extracted from different resources. This is done via decomposition of each entity into entities containing subject-predicate-object statements with equal provenance. The decomposed entities with same identifiers are merged using the *agg* operation into single entities with respective compound provenances. *agg* is implemented as weighted arithmetic mean (similarly to Example 4), with relevances $1, 0.2$ for the thesauri and article provenance, respectively. This ensures significantly higher relevance of the manually designed thesauri in case of conflict with the automatically extracted knowledge.

In order to access the aggregated emergent knowledge, we implemented a service evaluating simple conjunctive queries with negation (for the query language specification, see `http://smile.deri.ie/projects/egc/quickstart`). The query evaluation and presentation of the answers is implemented essentially following Example 5[1]. In addition to the ranking of the answer entities, statements associated to an entity are sorted according to the relevance of their arguments in descending order. Example queries and selected top answer statements are (answer certainties in brackets): *Q: ? : type : breast cancer* $\rightsquigarrow$ **cystosarcoma phylloides** *TYPE breast cancer (1); Q: rapid antigen testing : part of : ? AND ? : type : clinical study* $\rightsquigarrow$ **dicom study** *USE protein info (0.8),* **initial study** *INVOLVED patients (0.9).* The examples abstract from the result provenance, however, full-fledged presentation of answers to the above or any other queries can be tried live with CORAAL at `http://coraal.deri.ie:8080/coraal/`, using the *Knowledge* search tab or the guided query builder.

Currently the main means for accessing the EUREEKA deployment is the intuitive user-centric front-end in CORAAL. Applications may get RDF corresponding to the results presented in CORAAL from its Exhibit presentation layer, however, this is rather awkward. Therefore we are working on an API allowing for import and processing of arbitrary texts and RDF data in the N3 notation (cf. `http://www.w3.org/DesignIssues/Notation3`). The processed data are to be exported as N3 RDF, with the certainties and provenance represented according to the W3C note at `http://www.w3.org/TR/swbp-n-aryRelations/`.

4 Evaluation with Sample Users

In the CORAAL deployment, EUREEKA provides access to more than 15 million statements about ca. $350,000$ unique entities that are referred to by about

[1] The translation from the query language into entity representations is quite straightforward – positive and negative crisp query statements form triple relations that are associated with maximal and minimal certainty values, respectively. Statements with variables in the "object" position are inverted, so that the query can be translated as a single entity. The answer candidates and their fitness measures are then computed on the top of (possibly nested for inverted statements) `SELECT` queries on the `entities` table, with `WHERE` conditions corresponding to the query statements.

620,000 natural language terms. The knowledge base is covering ca. 11,700 Elsevier articles mostly related to cancer research and treatment. With an assistance of a three-member domain expert evaluation committee, we assessed issues deemed to be most important by the committee regarding applicability of the framework: (i) ease of use, real-time response; (ii) quality of answers to queries (users want to have as many good results entailed by the articles and thesauri as possible); (iii) appropriateness of the result ranking (users want to find the relevant results on the top). Note that we do not discuss evaluation of the document retrieval here, since it is related to the CORAAL search engine as such, but not to the main contribution of this paper (presentation of the general emergent knowledge processing framework).

Ease of use was addressed by the simple queries close to natural language, guided query builder and faceted browsing (supported by Exhibit, cf. `http://simile-widgets.org/exhibit/`), all offered within the EUREEKA front-end in CORAAL. The response is actually not an issue – results are presented within units of seconds in CORAAL (up to 90% of the lag owing to the HTML rendering overhead, not to the query processing itself). The two remaining issues were mapped to these tasks: (i) assessing correctness (i.e., precision) and completeness (i.e., recall) of variable instances provided within answers to significant queries; (ii) assessing number of relevant statements as a function of their rank in answers. The latter task was evaluated using significant entities as queries (such results in effect provide statements assumed to be related to the query entities based on the fitness, similarity and relevance in direct correspondence to raw results in Example 5).The significance of queries and entities to be used for the evaluation was determined as follows. First we picked 100 random entity names and generated 100 random queries based on the extracted content. We let the evaluation committee assess the significance of respective concept and statement queries by 1-5 marks (best to worst). We used the following best-scoring queries—Q_1 : *? : type : breast cancer;* Q_2 : *? : part of : immunization;* Q_3 : *? : NOT type : chronic neutrophilic leukemia;* Q_4 : *rapid antigen testing : part of : ? AND ? : type : clinical study;* Q_5 : *? : as : complementary method AND ? : NOT type : polymerase chain reaction*—and entities—E_1 : *myelodysplastic syndrome;* E_2 : *p53;* E_3 : *BAC clones;* E_4 : *primary cilia;* E_5 : *colorectal cancer*.

For a base-line comparison, we employed the open source edition of OpenLink Virtuoso (cf. `http://tinyurl.com/cf8ga2`), a triple store with database backend supporting rule-based RDFS inference and querying[2]. The content fed to EUREEKA was transformed to crisp RDFS, omitting the unsupported negative statements and provenance arguments before import to the base-line. EURE-

[2] Alternatives [13, 7] capable of either arbitrary meta-knowledge, or explicit trust representation in RDF were considered, too. However, the respective implementations allow neither for soft aggregation of emergent entities, nor for inherent exploitation of certainty in approximate answering of queries close to natural language. They can only expose the certainty and/or meta-knowledge via extended SPARQL queries. Therefore their capabilities are essentially equal to the "plain" Virtuoso RDF store base-line regarding our use case, while Virtuoso handles the relatively large amount of data more efficiently, presumably due to more mature data management engine.

EKA queries were mapped to statements with unique entity identifiers as per the `grounding` table and then translated to respective SPARQL equivalents to be executed using the base-line.

Approach	**Correctness and completeness**						**Relevance per answer ranking**				
	P	R	F	P_{nn}	R_{nn}	F_{nn}	1-10	11-50	51-100	101-200	201-...
EUREEKA	0.719	0.583	0.586	0.532	0.305	0.310	0.780	0.668	0.430	0.227	0.091
BASE	0.169	0.053	0.067	0.281	0.088	0.111	0.300	0.229	0.293	0.172	0.188

Table 1. Summary of the results

The evaluation results are summed up in Table 1. P, R, F columns contain precision, recall and F-measure ($\sim \frac{2(PR)}{P+R}$), respectively, averaged across the results of all evaluated queries. $X_{nn}, X \in \{P, R, F\}$ relate to average results of non-negative queries only (Q_1, Q_2, Q_4). Particular P, R values were computed as $P = \frac{c_r}{a_r}, R = \frac{c_r}{c_a}$, where c_r, a_r is a number of *relevant* and all answer entities returned, respectively. c_a is the number of all entities relevant to the query, as entailed by the documents in the CORAAL corpus (determined by the evaluation committee by means of manual analysis of full-text search results related to the entities occurring in the evaluated queries). The columns in the right hand part of Table 1 contain average values $\frac{s_r}{sz}$, where s_r, sz is the number of *relevant* and all statements in a given ranking range, respectively. The average goes across results corresponding to E_{1-5} query entities. The *relevance* was determined by unequivocal agreement of the evaluation committee. Results with certainty lower than 0.5 were disregarded (i.e., a statement was considered as a false positive iff it was deemed irrelevant and its absolute certainty value was 0.5 or more).

Regarding *correctness and completeness*, our approach offers almost three-times better results in terms of F-measure than the base-line. That holds for the negation-free queries supported by both frameworks. Obviously, the difference is even bigger for generic queries having no base-line results in two out of five cases. The increase in EUREEKA's precision was directly due to its two novel features unsupported by the base-line: (i) relevance-based aggregation of the initially extracted input; (ii) explicitly presented certainty of the results allowing for disregarding presumably uncertain ones. The increase in recall was caused by the approximate query evaluation that included also some correct results from answers with fitness lower than 1 (similar behaviour is not directly supported by the base-line). The relevance of EUREEKA answers is a clearly decreasing function of the ranking. However, no similar pattern can be seen for the base-line.

The absolute EUREEKA results may still be considered rather poor (F-measure around 0.3), but the evaluation committee unequivocally considered the ability of EUREEKA to perform purely automatically as an acceptable trade-off for the presence of some noise in the not-entirely-complete results. In conclusion, the evaluation with sample users confirmed that the innovative principles of the proposed approach lead to a better applicability in the current use case, when compared to a base-line state of the art solution.

5 Related Work

An implemented approach [4] generalising Description Logics in order to support vagueness as one form of uncertainty exists, however, it does not allow

for straightforward representation of contextual features. Moreover, logics-based approaches are usually not able to infer many meaningful conclusions from the rather sparse and noisy emergent inputs [3], which renders the querying in our use case practically unachievable if based on the logical inference.

The works [13, 7] propose generic framework for representing contextual features like certainty or provenance in RDF. These features are considered rather as "annotations" of RDF triples and thus can be merely queried for. It is impossible to use the certainty as a first class citizen for robust entity integration and/or query answering, unless one builds an ad hoc application tackling that on the top of either [13], or [7]. Similarity-based query post-processing with imprecision support is tackled by [8], however, the suggested iSPARQL framework handles uncertainty merely concerning query result filtering, disregarding a priori imprecise knowledge. This makes it rather inapplicable both to partial query evaluation and processing of the emergent uncertain knowledge before the actual querying. The work [11] extends the crisp RDF semantics by fuzzy degrees, but supports neither robust querying nor integration capabilities, nor context representation. Integration of RDF ontologies based on graph theory is tackled in [15], but incorporation of certainty degrees and contextual features into the presented method is non-trivial, since [15] is based on crisp binary relations.

Papers [1, 12] research techniques for ranking ontology concepts and anytime RDF query answering, respectively. The former approach is applicable for relevance-based sorting of query results, while the latter is apt for general robust, approximate and scalable query answering. However, both [1, 12] lack explicit support for uncertainty and contextual features.

All the approaches discussed so far also neglect as clearly defined and universal interface between the lexical and computational aspects of semantics as proposed in our approach. The Textrunner framework [2] provides an expressive search service based on natural language, which is very similar to the deployment of our framework in CORAAL. However, the framework provides neither for extracted knowledge integration, nor for complex (i.e., conjunctive or negative) querying, lacking an appropriate underlying computational semantics model.

Conceptual spaces [6], a geometrical formalisation of meaning, shares some similarities with our approach, namely uncertainty-aware, non-logical nature of representation, and multi-dimensionality of concept features. However, exploitation of emergent relational statements is not particularly straightforward within the framework, since it is tailored primarily to non-symbolic connectionist input. Moreover, there is neither a standardised implementation, nor a universal and intuitively applicable querying mechanism available for conceptual spaces.

6 Conclusions and Future Work

We have introduced a framework that addresses all the challenges specified in Section 1 on a well-founded basis. The framework has been implemented in the form of a respective EUREEKA prototype. We applied and evaluated the prototype within a practical use case of knowledge-based life science publication search. Our approach is novel and promising regarding practical emergent kno-

wledge processing, which has been proven not only by the results presented here, but also by our successful participation in the Elsevier Grand Challenge contest (cf. `http://www.elseviergrandchallenge.com/`).

In the near future, we are going to extend the user-centric query language by contexts and release the extended EUREEKA implementation as an open source module. In longer term, we have to investigate import of more complex ontologies into EUREEKA – so far we have covered only rather simple RDFS semantics of life science thesauri. Last but not least, we intend to provide means for distributed implementation of the principles introduced here in order to scale the framework up to arbitrarily large data.

Acknowledgments We have been supported by the 'Líon II' project funded by SFI under Grant No. SFI/08/CE/I1380. Deployment of EUREEKA within CORAAL would not be possible without the great work of Tudor Groza and much appreciated senior support of Siegfried Handschuh. Finally, we are very grateful to our evaluators and testers: Doug Foxvog, Peter Gréll, MD, Miloš Holánek, MD, Matthias Samwald, Holger Stenzhorn and Jiří Vyskočil, MD.

References

1. H. Alani, C. Brewster, and N. Shadbolt. Ranking ontologies with AKTiveRank. In *Proceedings of ISWC'06*, 2006.
2. M. Banko and O. Etzioni. The tradeoffs between open and traditional relation extraction. In *Proceedings of ACL-08: HLT*, pages 28–36. ACL, 2008.
3. S. Bechhofer et al. Tackling the ontology acquisition bottleneck: An experiment in ontology re-engineering, 2003. At `http://tinyurl.com/96w7ms`, Apr'08.
4. F. Bobillo and U. Straccia. fuzzyDL: An expressive fuzzy description logic reasoner. In *In Proceedings of FUZZ-08*, 2008.
5. P. Buitelaar and P. Cimiano. *Ontology Learning and Population*. IOS Press, 2008.
6. P. Gärdenfors. *Conceptual Spaces: The Geometry of Thought*. MIT Press, 2000.
7. O. Hartig. Querying Trust in RDF Data with tSPARQL. In *ESWC'09*, 2009.
8. C. Kiefer, A. Bernstein, and M. Stocker. The fundamentals of isparql: A virtual triple approach for similarity-based semantic web tasks. In *ISWC/ASWC*, 2007.
9. J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5), 1999.
10. A. Maedche and S. Staab. Discovering conceptual relations from text. In *Proceedings of ECAI 2000*. IOS Press, 2000.
11. M. Mazzieri. A fuzzy RDF semantics to represent trust metadata. In *Proceedings of SWAP'04*, 2004.
12. E. Oren, C. Guéret, and S. Schlobach. Anytime query answering in RDF through evolutionary algorithms. In *Proceedings of ISWC'08*, 2008.
13. B. Schueler, S. Sizov, S. Staab, and D. T. Tran. Querying for meta knowledge. In *Proceedings of WWW 2008*. ACM, 2008.
14. A. Tversky. Features of similarity. *Psychological Review*, 84(2):327–352, 1977.
15. O. Udrea, Y. Deng, E. Ruckhaus, and V. S. Subrahmanian. A graph theoretical foundation for integrating RDF ontologies. In *Proceedings of AAAI'05*, 2005.
16. J. Voelker, D. Vrandecic, Y. Sure, and A. Hotho. Learning disjointness. In *Proceedings of ESWC'07*. Springer, 2007.