Linked Data Meets Computational Intelligence
Position paper

Christophe Guéret
Vrije Universiteit Amsterdam
De Boelelaan 1105
1081 HV Amsterdam, Nederlands

Abstract

The Web of Data (WoD) is growing at an amazing rate and it will no longer be feasible to deal with it in a global way, by centralising the data or reasoning processes making use of that data. We believe that Computational Intelligence techniques provides the adaptiveness, robustness and scalability that will be required to exploit the full value of ever growing amounts of dynamic Semantic Web data.

Introduction

The Web of Data (WoD) is growing at an amazing rate as more and more data-sources are being made available online in RDF, and linked. Because of its size and dynamicity it is no more possible to consider the WoD as a large and static information system. It should instead be considered a complex system in constant evolution. Thus, new algorithms will have to be developed to cope with this new context.

The decentralized nature, and future, of the WoD

As the WoD grows we believe a centralized approach where data is aggregated into large databases will no longer be possible. This assertion is based on the following 3 observations:

1. **The massive amount of data and data sources**

   Data stores have become increasingly efficient and can now deal with billions of triples but the number of different data sources is also growing rapidly. As the number of data stores increases, incoherences, uncertainty and contradictions are more likely to appear. As an illustration of these size issues, Table 1 shows examples of publicly available datasets.

   The execution of federated queries over live SPARQL endpoints is known to be extremely expensive, because known optimizations (for example to deal with joins) do not work in the distributed case. Instead, snapshots are taken at intervals, dumped into gigantic repositories and made available in database style for querying. But both the exponential increase of figures and the risk for conflicts creates a serious threat to the viability of this approach.

Data source	Number of triples
US Census data	≈1 billion
DBPedia	≈274 million
MusicBrainz	≈36 million
DBLP	≈10 million
WordNet	≈2 million

Table 1: Example of publicly available data sets and their respective size.

2. **Privacy and coherence problems of centralisation**

   Data providers may see privacy issues in providing their data to a central store. This is particularly true for information related to social networks which is, by nature, highly personal. People providing social information about themselves and their friends want to keep control over their data. For these reasons, decentralisation is considered to be the future of social networks (Yeung et al. 2009).

   Additionally, the centralisation of data also raise compatibility issues. Data sources using different schemas are better off kept separated and made compatible through a translation layer rather than translated into a single schema and merged into one data store. A Peer Data Management System (PDMS) (Tatarinov et al. 2003) is an example of such a decentralised system. A PDMS is defined by a set of autonomous peers hosting data according to different schemas they relate through semantic mappings. For new peers willing to share data, PDMS has the advantage of a low entry cost, which is limited to the definition of mappings with one of the peers already present.

   One may also consider having data physically centralised but kept de-centralised. This model is that of a single data store hosting several named graphs or that of a cloud-based architecture hosting several stores. Although preserving the provenance/context information, this model leads to having the data being served by a single store. As such, privacy concerns may still be present.

3. **Opaque data locality**

   The recent development of cloud computing has highlighted an interesting fact about linked data consumers and providers: their interest is in having data available

somewhere and, more importantly, always accessible - not in knowing where or how it is stored or processed. And this is exactly what cloud computing provides: an abstraction over physical constraints related to data serving. Once put in a cloud, such as Amazon EC2 (Amazon Web Services 2009), data is made available in a scalable fashion. The addition/removal of nodes in the cloud is a transparent operation for the user which sees the data provider as a single entity.

Cloud computing comes close to the idea of Tuple-space based middleware such as the Triple Space Computing (Fensel 2004). Such middleware creates a persistent shared information space every peer can read from and write data to in an asynchronous way. Tasks such as sorting & storing that data or dealing with failure of storage components are in the charge of autonomic entities that exhibit several self-* behaviours (Kephart and Chess 2003).

We believe that the field of Computational Intelligence provides a set of techniques that can deal with the decentralised nature of the future WoD. Before discussing how Computational Intelligence applies to the Semantic Web, we briefly introduce the field.

Computational intelligence

Computational Intelligence (CI) is a branch of AI focused on heuristic algorithms for learning, adaptation and evolution. Evolutionary computation and collective intelligence are among the most popular paradigms of CI. They provide particular advantages :

- **Learning and adaptation**: the performance of CI algorithms improves during their execution. As good results are found, they learns why these results are good and/or how to find similar ones. This learning can also cope with changing conditions and consequently adaptat.
- **Simplicity**: CI design principles are simple. For instance, an evolutionary algorithm is based on the survival of the fittest: in an iterative process, solutions are guessed, verified and deleted if they are not fit. The expected result, that is to find an optimal solution to the problem, comes as a consequence of the basic mecanism. This bottom-up approach differs from the complex top-down approaches commonly used to deal with hard problems.
- **Interactivity**: CI techniques are used in a continously running, typically iterative, process. This implies two things: First, at any-time, the best result found so far can be returned as an answer to the posed problem. Secondly, CI algorithms can be made interactive and incorporate a user into the loop. This interaction is of great benefit when the quality of a result is difficult to appreciate in an automated way (Takagi 2001).
- **Scalability, robustness and parallelisation**: all of these three advantages result from using a population of co-evolving solutions instead of focusing on only one. Because each member of the population is independent, algorithms are easy to parallelise. Also, the bad performance of some members will be compensated by the global efficiency of the entire population, making the population robust against individual failures.

It is widely recognised that new adaptive approaches towards robust and scalable reasoning are required to exploit the full value of ever growing amounts of dynamic Semantic Web data (Fensel and Harmelen 2007). Computational Intelligence provides the features that algorithms dealing with the WoD will have to exhibit.

This paper

This paper highlights some of the research that has been conducted so far to apply CI to Semantic-Web problems. Although this combination is a recent field of study, some achievements have already been made: the eRDF evolutionary algorithm makes it possible to query live SPARQL endpoints without any pre-processing, such as the creation of an index or the retrieval of data dumps; by gossiping semantic information a PDMS can self-organise and improve its interoperability at large; a swarm of agents taking care of the localisation of data within a Triple Space produces optimised data clusters and query routes.

All of these applications and some others will be detailed in the rest of the paper. A particular emphasis will be put on the joint work currently conducted by the CI and Knowledge Representation and Reasoning groups of the Vrije Universiteit of Amsterdam. We are particularly interested in fostering emergence and self-organisation (Wolf and Holvoet 2005) and turning data-related tasks into anytime optimisation problems.

Emergence and self-organisation in decentralised knowledge systems

One can define emergence as the apparition of global behaviour/structure/pattern from local interaction between elements of a system (Wolf and Holvoet 2005). The use of emergence can be compared to the divide&conquer-like approach commonly used for distributed computing. In divide&conquer, a supervisor divides a global task into subtask, send it to a set of computing resources and coordinate the results (Figure 1).

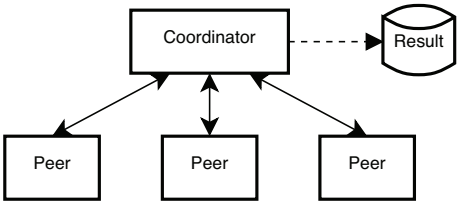

Figure 1: Divide and conquer resolution. A central coordinator direct peers toward a specific goal

Architectures focused on emergence define the local interaction between entities. The global goal to achieve is unknown to the entities, nor does one entity have as a complete view of the solution at any moment. The expected goal emerges from the local interactions (Figure 2).

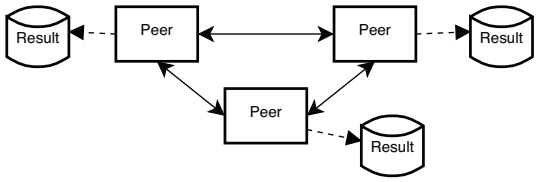

Figure 2: Global properties emerge from local interaction among the peers

In the following, we introduce work that has been done to design emergence of **semantic interoperability**, **structure** (semantic overlays and data clusters) and **knowledge** (new triples) in distributed semantic networks.

Semantic interoperability

A set of different data sources is not guaranted to share a common schema for their data. The semantic interoperability of such a set requires setting up schema mediation allowing any participant to communicate with any other.

Semantic gossiping (Cudré-Mauroux 2006) as been proposed as an alternative to the common strategy of defining a global schema every participant should map to. Local agreements in the form of pair-wise schema mappings are established among the peers. Gossiping (Demers et al. 1987) is a probabilistic message passing algorithm where peers exchanges part of their knowledge with their neighbours. By exchanging information about the local agreements they have established, peers are able to improve their own set of mappings and thus enhance their interoperability with other peers. As a global result of these local enhancements, semantic interoperability can be achieved at network scale (Cudre-Mauroux and Aberer 2004).

Self-organising structure

The use of non-centralised RDF spaces poses the problem of the localisation of data. Every peer is a potential provider of any given triple. Overlay structure and optimized data distribution have to be used to provide a more efficient use of this decentralised data.

A common example of self-organising optimized data distribution is that of Distributed Hash Tables (DHT) that assign data to a particular peer based on a specific identifier space. The SwarmLinda project combines a Tuple space approach with an artificial ants algorithm to create a self-organising system showing the emergence of data clusters and optimised routes for querying that data. Artificial ants applies a sorting algorithm inspired by the brood sorting ability of real ants: they take some triples at one location and move them to an other one where similar triples are found (Graff 2008). When they are moved, traces are left by the triples on the network connections used for their journey. These traces are used later on to do query routing (Koske 2009).

The PIAF system (Guéret, Monmarché, and Slimane 2007) also make use of traces left by pieces of information moved from peer to peer. Instead of being used for query routing, these traces are used to efficiently disseminate information. Ants carying a piece of information will follow paths on which traces are the most similar to the carried item.

Note that collective intelligence has already proven to be efficient for network routing (Dorigo and Caro 1998) and data clustering (Deneubourg et al. 1990). The work reported in this section shows that these techniques can be combined with Semantic Web technology in order to make a better use of decentralised data.

Emergence of Knowledge

Knowledge can also by designed to emerge from local interaction between different RDF spaces. We make a distinction between two types of knowledge one can expect to see emerging in a network:

- The computation of a closure under RDFS semantics, for instance, resulting in the creation of new triples thus materializing information that was implicit.
- In a more generic way, the application of rules lead to the creation of new explicit knowledge that was not anticipated nor implicit.

Both are essentially the result of some forward-chaining reasoning based on axioms of different expressivity. Because of this, it is possible to design a single architecture able to exhibit both type of emerging properties.

In our work on swarm-based reasoning, reasoning tasks are distributed among a number of agents, i.e. autonomous micro-reasoning processes that are referred to as “beasts”, which explore data distributed across distinct locations. In contrast to batch-oriented reasoning, their exploration is a continuous process. The generic model hence defined is that of an constantly evolving anthill inhabited, used and modified by restless beasts.

- Implicit knowledge: In (Dentler, Guéret, and Schlobach 2009), we applied this model to the computation of closure over a decentralised RDF graph. Such a graph is seen as a network, where each subject and each object is a node and each property an edge. A path is composed of several nodes that are connected by properties, i.e. edges. The beasts, each representing an active reasoning rule, which might be (partially) instantiated, move through the graph by following their paths. Swarms of independent light- weight beasts travel from RDF node to RDF node and from location to location, checking whether they can derive new information according to the information that they find on their way. Whenever a beast traverses a path that matches the conditions of its rule, it locally adds a new derived triple to the graph. Given an added transition capability between (sub-)graphs, it can be shown that the method converges towards closure.
- Explicit knowledge: when dealing with truly decentralised data, that each user keeps locally, there is more to be done than computing the closure. Beasts can be designed to apply arbitrary rules and extend personal data. Bibliographic data is an example of such data that would ideally be maintained by its authors and directly reasoned

over. By looking at the publications made by different people, a social network can be infered when assuming that two authors of a same paper know each other. Beasts can be instanciated with this simple rule and sent walking over the graph. Whenever an author of a paper and then a second author of the same paper are found, a *knows* relation is created between the two. After some time, a social network emerges.

An important aspect of swarm based reasoning is that it is not the data that is sent to a reasoner but it's instead the reasoning process that moves towards the data. The amount of data disclosed by every peer is minimal (only what is needed by the beast) and is kept in control by the user (nothing prevents a beast to authenticate itself to the peer). The progress of this work can be followed online at `http://www.beast-reasoning.net/`.

Anytime resolution of complex optimisation problems across decentralised data sources

Several problems within the Semantic Web can be expressed as optimisation problems. For instance, ontology mapping and SPARQL query answering can be seen as a combinatorial optimisation problems which consist of assembling an optimal set of entities. CI techniques are suited for this type of task, in particular, when the search space is very large (e.g. costly to explore exhaustivly) or changing. The algorithms presented in this section are all population-based. That is, they optimise a set of candidate solutions instead of optimising only one. The way this set is used differs from one algorithm type to the other, depending on whether a Evolutionary algorithm or a Particle Swarm Optimisation is used.

Evolutionary algorithms define a competition metaphor: only the best solution(s) can survive and be improved. The basic loop (see Figure 3) consists of the creation of new candidate solutions, their evaluation and selection of the best candidates for the next loop (Eiben and Smith 2003).

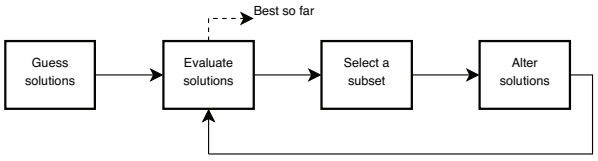

Figure 3: A basic evolutionary loop

Particle Swarm Optimisation is instead inspired from bird swarms and fish schooling were all the candidate solutions survive and evolve. The competition taking place in the EAs is replaced by social metaphors were the the best candidate solution attracts others to it and leads the swarm (Engelbrecht 2006). At every iteration, individuals move towards some direction, evaluate the quality of their current position and compare it with how they perceive the quality of their neighbours' positions. Then, a decision is taken on which direction to go for the next step (see Figure 4).

We now present population-based algorithms applied to the classic problems of ontology matching and query answering over an RDF dataset.

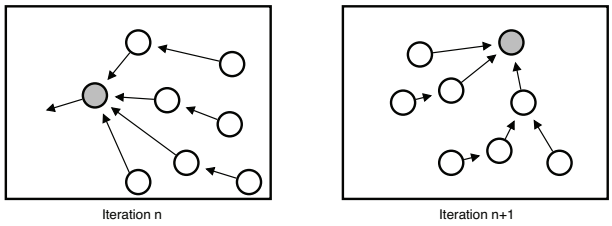

Figure 4: The population follows its best neighbour. The globally best individual is greyed. When this best changes, the entire population changes direction.

Ontology matching

Ontology matching, that is establishing mappings between concepts of different ontologies, is a common task Semantic Web developers are confronted with. Considering two ontologies of n_1 and n_2 each, the task consists in finding which concept pairing is the best out of the $n_1 \times n_2$ possible pairs. Several algorithms exist to explore this search space efficiently and output a set of optimal mappings. These can be found on the Ontology Alignment Evaluation Initiative website[1].

The continuous optimisation feature of evolutionary algorithms and particle swarm techniques enables the progressive construction of a set of mappings. GAOM (Wang, Ding, and Jiang 2006) and GOAL (Martinez-Gil, Alba, and Montes 2008) are two initatives making use of a genetic algorithm (GA). They both use a feature-based similarity function to compare two ontologies. This function is a weighted combination of measures used to evaluate a mapping according to different aspects such as the edit distance between labels or the number of shared instances. During the evolutionary process, candidate solutions having the highest similarity score are kept and improved. Going from one generation to the next one, bad mappings are dropped and replaced by new ones whose quality will be evaluated in the next evaluation phase of the evolutionary loop.

The MapPSO (Bock and Hettenhausen 2008) algorithm follows a similar strategy based on a similarity function but makes use of PSO instead of a GA. A set of candidate solutions fly above a search space defined by the similarity function in a quest for positions with the highest correctness. The main difference between the GA approach is the social behaviour of individuals in a PSO. Instead of a fighting for survival, they co-operate and share knowledge about their environment.

The shared advantage of using a Genetic Algorithms or a PSO algorithm for ontology mapping is the ability to incorporate new similarity measures at limited cost. This is because similarities are computed only on candidate solutions and are not pre-computed over the entire search space.

[1] `http://oaei.ontologymatching.org/`

RDF query answering

Giving an answer to a query expressed over an RDF dataset is basically a pattern matching problem. One has to find triples matching a particular set of graph patterns. The classical way of dealing with this is to retrieve, and then merge, partial results for every single graph pattern to match.

We introduced evolutionary RDF (eRDF) query answering (Oren, Guéret, and Schlobach 2008) as an alternative to these approaches. Complete variable assignments are randomly generated and checked against the basic graph patterns defined by the query. The quality of a solution is proportional to the correctness of instanciated graph patterns. This evaluation leaves room for customisation as any metric can be used to appreciate the correctness of a given instanciated graph pattern. For instance, a generated triple <uri, foaf:name, ''Amsterdam''> can be checked for existance with a boolean outcome (0=non existant, 1=exist). It can also be compared to an other triple <uri, foaf:name, ''A'dam''> using an bounded string-based distance and a real-valued outcome. As for the ontology matching case, changes on the measure for the validity of a candidates solutions have a limited impact on the performances. Based on such well-defined, and user-specified, notions of similarity eRDF returns "perfect" answers if possible, and approximate answers if necessary. The ongoing work on eRDF and can be followed online on http://www.erdf.nl.

A speciality of eRDF is to allow distributed queries over live data-sources as only very simple unary queries are needed (assert correctness). Additionally, eRDF can issue all of its queries in a fully parallel fashion. There is no theoretical restriction on the number of data-sources and their data-size only marginally increases individual response times. Of course, increasing data-size in combination with a constant population size will increase convergence time. However, given the any-time character of evolutionary methods good answers are still returned comparatively quickly. This makes eRDF an interesting alternative for exploration and discovery for the Web of Data. In order to illustrate this feature, we recently developped on top of eRDF the Like? discovery engine (Guéret, Groth, and Schlobach 2009). Using that engine a user can submit a query to discover what is like something else, the likeness being related to the number of shared properties. Like? is a simple application: once a user has indicated a keyword and pressed "go", a query is sent to Sindice in order to find a resource related to that keyword. The set of triples describing that resource are turned into a SPARQL query sent to eRDF which, in turns, streams the results found. These results are resources sharing at least one property with the query resource. This application is accessible at http://www.like.nu.

As demonstrated by the Like? application, eRDF is able to query not only the sets presented in Table 1 but also the billion triple set provided in the context of the Billion Triple Challenge[2] of ISWC2009.

[2] http://challenge.semanticweb.org/

Conclusion

In this position paper, we described how the WoD can and needs to be seen as an ever changing, dynamic, complex system. We believe that it will no longer be feasible to deal with the WoD in a global way, by centralising the data or reasoning processes making use of that data. New adaptive, robust and scalable techniques are required to exploit the full value of ever growing amounts of dynamic Semantic Web data. We argue for using Computational Intelligence techniques as the basis for these new algorithms and gave an overview of the research that has been conducted in this direction.

References

Amazon Web Services. 2009. Public data sets on aws. http://aws.amazon.com/publicdatasets/.

Bock, J., and Hettenhausen, J. 2008. MapPSO results for oaei 2008. In Shvaiko, P.; Euzenat, J.; Giunchiglia, F.; and Stuckenschmidt, H., eds., *OM*, volume 431 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Cudre-Mauroux, P., and Aberer, K. 2004. A necessary condition for semantic interoperability in the large. In *In Ontologies, DataBases, and Applications of Semantics for Large Scale Information Systems (ODBASE*.

Cudré-Mauroux, P. 2006. *Emergent semantics*. Ph.D. Dissertation, EPFL, Lausanne. Prix EPFL de doctorats - EPFL doctorate award (2007).

Demers, A.; Greene, D.; Hauser, C.; Irish, W.; Larson, J.; Shenker, S.; Sturgis, H.; Swinehart, D.; and Terry, D. 1987. Epidemic algorithms for replicated database management. In *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing*, 1–12.

Deneubourg, J. L.; Goss, S.; Franks, N.; Sendova-Franks, A.; Detrain, C.; and Chrétien, L. 1990. The dynamics of collective sorting robot-like ants and ant-like robots. In *Proceedings of the first international conference on simulation of adaptive behavior on From animals to animats*, 356–363. Cambridge, MA, USA: MIT Press.

Dentler, K.; Guéret, C.; and Schlobach, S. 2009. Semantic web reasoning by swarm intelligence. In *Proceedings of the 5th International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS2009), collocated at the 8th International Semantic Web Conference (ISWC2009)*.

Dorigo, M., and Caro, G. D. 1998. Antnet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research* 9:317–365.

Eiben, A., and Smith, J. 2003. *Introduction to evolutionary computing*. Springer.

Engelbrecht, A. P. 2006. *Fundamentals of Computational Swarm Intelligence*. John Wiley & Sons.

Fensel, D., and Harmelen, F. v. 2007. Unifying reasoning and search to web scale. *IEEE Internet Computing* 11(2):96–95.

Fensel, D. 2004. Triple-space computing: Semantic web services based on persistent publication of information. *Intelligence in Communication Systems* 43–53.

Graff, D. 2008. Implementation and evaluation of a swarm-linda system. Technical report, Freie Universitat Berlin.

Guéret, C.; Groth, P.; and Schlobach, S. 2009. erdf: Live discovery for the web of data. ISWC2009 Billion Triple Challenge.

Guéret, C.; Monmarché, N.; and Slimane, M. 2007. Sharing resources in a p2p network with artificial ants. *Journal of Mathematical Modelling and Algorithms (JMMA)* 6:345–360.

Kephart, J. O., and Chess, D. M. 2003. The vision of autonomic computing. *Computer* 36(1):41–50.

Koske, S. 2009. Swarm approaches for semantic triple clustering and retrieval in distributed rdf spaces. Technical report, Freie Universitat Berlin.

Martinez-Gil, J.; Alba, E.; and Montes, J. F. A. 2008. Optimizing ontology alignments by using genetic algorithms. In Guéret, C.; Hitzler, P.; and Schlobach, S., eds., *Nature inspired Reasoning for the Semantic Web (NatuReS2008)*, volume 419 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Oren, E.; Guéret, C.; and Schlobach, S. 2008. Anytime query answering in rdf through evolutionary algorithms. In *7th International Semantic Web Conference (ISWC2008)*.

Takagi, H. 2001. Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. *Proceedings of the IEEE* 89(9):1275–1296. Invited Paper.

Tatarinov, I.; Ives, Z. G.; Madhavan, J.; Halevy, A. Y.; Suciu, D.; Dalvi, N. N.; Dong, X.; Kadiyska, Y.; Miklau, G.; and Mork, P. 2003. The piazza peer data management project. *SIGMOD Record* 32(3):47–52.

Wang, J.; Ding, Z.; and Jiang, C. 2006. Gaom: Genetic algorithm based ontology matching. In *APSCC '06: Proceedings of the 2006 IEEE Asia-Pacific Conference on Services Computing*, 617–620. Washington, DC, USA: IEEE Computer Society.

Wolf, T. D., and Holvoet, T. 2005. Emergence versus self-organisation: Different concepts but promising when combined. In *Engineering Self-Organising Systems*, 1–15. Springer-Verlag.

Yeung, C.; Liccardi, I.; Lu, K.; Seneviratne, O.; and Berners-Lee, T. 2009. Decentralization: The future of online social networking. In *W3C Workshop on the Future of Social Networking Position Papers*.