Abstract

1 Introduction

To make use of the exciting possibilities that the Semantic Web offers, Artificial Intelligence techniques, including Evolutionary methods, have been used in many of its related tasks. Examples of these include query answering[1] and reasoning[?]. However, part of the great potential of the Semantic Web lies in the ability of software agents to use its semantic structure to collect relevant or related data. This paper explores the use of Evolutionary Programming to train such agents. The agents will be given a starting set of triples, and attempt to effectively query a dataset for related information. Entailments gained from the combined results is used as a scoring mechanism for the relevancy of results. First, the method and structure of the agents is discussed. Then, the software and hardware environment used in testing is specified. Finally, the results gained from testing are interpreted, and any conclusions that can be made regarding this method are given.

2 Method

Here, the approach used by the program to evolve the agents query construction is outlined. All agents are given a starting set of triples, and can load a limited amount of additional triples from the dataset each round.

2.1 Algorithm

The program's main structure is that of an Evolutionary Programming algorithm[3]. For mutation, bit flipping is used instead of Gaussian mutation, since the parameter values have no ordering. Initially, the population is filled with randomly generated agents. Each round, all the agents take a number of steps, and afterwards the round the program reads their scores. Agents keep their own score based on the amount entailments they make. Elitist selection is done by eliminating the ten percent worst achievers, and replacing them with the offspring of the ten percent best achievers.

2.2 Agents

Each agent consists of a personal memory, a genotype of two variably sized parameter lists, and a query list. Agents construct their query list by mapping terms from the personal memory into queries according to genotype. In addition to the personal memory, each agent has access to five common ontological terms for use in the queries. The size of the query list generated can differ in accordance with the parameters, but all agents have a shared maximum of results they can load into memory per round. To score, the agents conduct entailment over their memory and count the number of entailments gained. RDFS entailment patterns are used as the basis[7].

3 Environment

3.1 Dataset

The dataset used as a search space for the agents was the union of the DBpedia persondata file[9] together with the DBpedia ontology[10] to allow for more complex entailments. The DBpedia statistics [source] report 831,558 instances of type Person in its dataset.

At each iteration of searching, an agent can load a maximum of a fifty triples into personal memory in addition to the fifty triples of the initial memory already present. Thus, at each step, the search space of all solutions of a step is:

$$S = \{x : 100 \leq |x| \leq 200, I \subseteq x\} \tag{1}$$

Where G is the complete dataset and I is the initial memory.

3.2 Software

The Sesame framework was used both as a datastore and a querying engine. The entire program was written in Java. Jena was used for its entailment engine.

3.3 Hardware

For the experiment, the compute service of the DAS-4 VU cluster was used. The programs threads were run on twenty-four 2.4 Ghz Intel E5620 processors supported by 130GB of memory.

4 Results

The experiment was run for a thousand rounds over a population of 300 agents. Results were gathered on the changes in genotype and behaviour of the agents. Two clear trends were identified. The amount of queries conducted per round rose sharply, suggesting a strategy of loading heterogenous results from varying queries was favourable. In contrast, the usage of ontological terms in queries dropped sharply, implying a more general approach of searching for related triples trumped querying for specific ontological relations.

[Figures 1 and 2 graphs showing the trends in behaviour]

Surprisingly, these changes in behaviour did not lead to a significant change in performance. The amount of entailments made by both the entire population and the top performing agents was assessed. In no category was any noteworthy improvement gained.

[Figure 3 graph showing no improvement performance over rounds]

The lack of improvement despite the changes in behaviour suggest that the differences between agents in querying strategy do not affect performance. This can be explained by comparing the static behaviour all agents share with the behaviour that is evolutionary. All agents search based on terms from their personal memory. What kind of terms they select from their memory (subjects, predicates, objects) and where in the query clause they place these terms is evolutionary, in addition to the choice they make whether to use ontological predicates. Since the scoring is based on a generic total of all possible entailments, the collection of any set of related triples may constitute a near-optimal solution.

5 Conclusion

5.1 Related work

Recently, it has been argued that many Semantic Web related tasks can be more successfully approached as optimization problems suitable for Evolutionary and Swarm computing. Specifically, the scalability of Evolutionary solutions has been noted to be required to deal with the massive and opaque nature of Semantic Web data.[2] Evolutionary Computing has already been used to optimize Query Answering, and again advantages in memory usage and scalability have been noted. [1] For the task of appropriate Query Construction, its complexity has been noted and some solutions have been put forward for it regarding keyword search, although these do nut utilize Evolutionary methods.[4][5]

References

[1] Guret, Christophe, et al. *An evolutionary perspective on approximate rdf query answering*. Scalable Uncertainty Management. Springer Berlin Heidelberg, 2008. 215-228.

[2] Guret, Christophe, et al. "Evolutionary and swarm computing for the semantic web." *Computational Intelligence Magazine*, IEEE 7.2 (2012): 16-31.

[3] Eiben, Aguston E., and Marc Schoenauer. "Evolutionary computing." *Information Processing Letters* 82.1 (2002): 1-6.

[4] Zhou, Qi, et al. *SPARK: adapting keyword query to semantic search*. Springer Berlin Heidelberg, 2007.

[5] Zenz, Gideon, et al. "From keywords to semantic queriesIncremental query construction on the Semantic Web." *Web Semantics: Science, Services and Agents on the World Wide Web* 7.3 (2009): 166-176.

[6] Antoniou, Grigoris and Van Harmelen, Frank, *A semantic web primer*. MIT press 2004.

[7] W3C (2014), *RDF Schema 1.1*. Available at: http://www.w3.org/TR/rdf-schema/ (Accessed: 16 May 2014).

[8] W3C (2014), *RDF 1.1 Semantics*. Available at: http://www.w3.org/TR/2014/REC-rdf11-mt-20140225/ (Accessed: 17 May 2014).

[9] *Persondata*. Available at: http://downloads.dbpedia.org/3.9/en/persondata_en.nt.bz2 (Accessed: 11 May 2014).

[10] *DBpedia ontology*. Available at: http://downloads.dbpedia.org/3.9/dbpedia_3.9.owl.bz2 (Accessed: 11 May 2014).