ISWC 2008

The 7th International Semantic Web Conference

Christophe Guéret
Pascal Hitzler
Stefan Schlobach

Nature inspired Reasoning for the Semantic Web (NatuReS)

October 27, 2008

The 7th International Semantic Web Conference
October 26 – 30, 2008
Congress Center, Karlsruhe, Germany

ISWC 2008

Platinum Sponsors

Ontoprise

Gold Sponsors

BBN
eyeworkers
Microsoft
NeOn
SAP Research
Vulcan

Silver Sponsors

ACTIVE
ADUNA
Saltlux
SUPER
X-Media
Yahoo

Organizing Committee

General Chair
Tim Finin (University of Maryland, Baltimore County)

Local Chair
Rudi Studer (Universität Karlsruhe (TH), FZI Forschungszentrum Informatik)

Local Organizing Committee
Anne Eberhardt (Universität Karlsruhe)
Holger Lewen (Universität Karlsruhe)
York Sure (SAP Research Karlsruhe)

Program Chairs
Amit Sheth (Wright State University)
Steffen Staab (Universität Koblenz Landau)

Semantic Web in Use Chairs
Mike Dean (BBN)
Massimo Paolucci (DoCoMo Euro-labs)

Semantic Web Challenge Chairs
Jim Hendler (RPI, USA)
Peter Mika (Yahoo, ES)

Workshop chairs
Melliyal Annamalai (Oracle, USA)
Daniel Olmedilla (Leibniz Universität Hannover, DE)

Tutorial Chairs
Lalana Kagal (MIT)
David Martin (SRI)

Poster and Demos Chairs
Chris Bizer (Freie Universität Berlin)
Anupam Joshi (UMBC)

Doctoral Consortium Chairs
Diana Maynard (Sheffield)

Sponsor Chairs
John Domingue (The Open University)
Benjamin Grosof (Vulcan Inc.)

Metadata Chairs
Richard Cyganiak (DERI/Freie Universität Berlin)
Knud Möller (DERI)

Publicity Chair
Li Ding (RPI)

Proceedings Chair
Krishnaprasad Thirunarayan (Wright State University)

Fellowship Chair
Joel Sachs (UMBC)

Workshop Organization

Programme Chairs

Programme Committee

Table of Contents

Text-Based Ontology Enrichment Using Hierarchical Self-organizing Maps

Emil Şt. Chifu and Ioan Alfred Leţia

Technical University of Cluj-Napoca, Department of Computer Science, Bariţiu 28, RO-400027 Cluj-Napoca, Romania
{Emil.Chifu, letia}@cs.utcluj.ro

Abstract. The success of the Semantic Web research is dependent upon the construction of complete and reliable domain ontologies. In this paper we describe an unsupervised framework for domain ontology enrichment based on mining domain text corpora. Specifically, we enrich the hierarchical backbone of an existing ontology, i.e. its taxonomy, with new domain-specific concepts. The framework is based on an extended model of hierarchical self-organizing maps. As being founded on an unsupervised neural network architecture, the framework can be applied to different languages and domains. Terms extracted by mining a text corpus encode contextual content information, in a distributional vector space. The enrichment behaves like a classification of the extracted terms into the existing taxonomy by attaching them as hyponyms for the nodes of the taxonomy. The experiments reported are in the "Lonely Planet" tourism domain. The taxonomy and the corpus are the ones proposed in the PASCAL ontology learning and population challenge. The experimental results prove that the quality of the enrichment is considerably improved by using semantics based vector representations for the classified (newly added) terms, like the document category histograms (DCH) and the document frequency times inverse term frequency (DF-ITF) weighting scheme.

Keywords: taxonomy enrichment, unsupervised neural network, extended growing hierarchical self-organizing maps (Enrich-GHSOM), document category histograms (DCH), document frequency times inverse term frequency (DF-ITF) weighting scheme, centroid vector.

1 Introduction

The most important prerequisite for the success of the Semantic Web research is the construction of complete and reliable domain ontologies. Building ontologies is still a time consuming and complex task, requiring a high degree of human supervision and being still a bottleneck in the development of the semantic web technology.

The process of domain ontology enrichment has two inputs, an existing ontology – which plays the role of background knowledge – and a domain text corpus. The aim of our work is to automatically adapt the given ontology according to a domain specific corpus. We enrich the hierarchical backbone of the existing ontology, i.e. its taxonomy, with new domain-specific concepts extracted from the corpus [14].

Our framework for taxonomy enrichment is based on an extended model of hierarchical self-organizing maps, which represent an unsupervised neural network architecture. The candidates for labels of newly inserted concepts are terms collected by mining a text corpus. The term extraction process is based on recognizing linguistic patterns (noun phrases) in the domain corpus documents. Each term encodes contextual content information, in a distributional vector space. The context features of a term are the frequencies of its occurrence in different documents of the corpus. The classification of the extracted terms into the taxonomy of the given ontology proceeds by associating every term to one target node of the taxonomy, based on a similarity in the distributional vector space. That term becomes a new concept added to the taxonomy, and it is attached as hyponym (successor) under the target node.

Unsupervised hierarchical neural models in general start the growing of a dynamic tree-like topology from a single initial node. Our neural network model, called *Enrich-GHSOM*, is an extension of one of these existent systems, GHSOM [8], and it allows the growing to start from an initial tree. The taxonomy that is subject to enrichment is given as the initial state of the hierarchical self-organizing map. So, an essentially symbolic knowledge structure – taxonomic tree – is converted into a neural representation as an initial state of the hierarchical self-organizing map. The actual taxonomy enrichment takes place via an unsupervised training of the neural network by exposing the initialized hierarchical self-organizing map to the vector representation of the terms extracted from the domain corpus. A reverse, neural-symbolic translation is done after this enrichment process. This is actually the knowledge extraction step whose output is the final enriched taxonomy. Our taxonomy enrichment framework is a hybrid one, as it has to deal with neural-symbolic integration. The neural-symbolic translations in both directions have been naturally achieved, since our framework merely operates upon the taxonomic structure of the ontology, which is in agreement with the hierarchical structure of the self-organizing neural network.

In the rest of the paper, after a review of related work, section 3 presents the neural network learning solution chosen and adapted in our framework. Then section 4 details the architecture and implementation of the taxonomy enrichment framework and section 5 describes the experimental results. Conclusions and future directions are presented in section 6.

2 Related Work

There are two main categories of approaches for taxonomy enrichment [3]: methods based on *distributional similarity* and *classification of terms into an existing taxonomy* on one hand, and approaches using *lexico-syntactic patterns*, also known as Hearst patterns [10], on the other hand. Our enrichment approach belongs to the former category.

In the term classification approach, the terms extracted from a domain specific corpus of text are classified into an existent taxonomy [14, 6, 1, 16, 15]. In a top-down variant of this classification [14, 1, 16], there is a top-down search on the existent taxonomy in order to find a node under which a new term is to be inserted as a successor (hyponym). The classification of the terms is made according to a similarity measure in a distributional vector space. Each term is represented as a vector with information about different contexts of its occurrences in the corpus.

The top-down classification behavior in our framework is modeled by a growing hierarchical self-organizing map (GHSOM) architecture [8] extended with the possibility to set an initial state for the tree-like neural network. In our new extended neural model, called Enrich-GHSOM, the given taxonomy is set as the initial state of the neural network. The model allows to classify the extracted terms into the existing taxonomy by attaching them as hyponyms for the intermediate and leaf nodes of the taxonomy. Details of this process are given in section 4.2.

A similar, although non top-down approach is [15]. There is a search for a node to attach a new concept as a hyponym of, by finding a place in the existent taxonomy where the corpus derived semantic neighbors of the candidate concept are most concentrated. He supposes that at least some of the semantic neighbors are already in the taxonomy, and he defines a function to compute the class label for the set of neighbors – a hypernym for all the neighbors. This class label becomes the concept under which to attach the new term as hyponym. The similarity measure to find neighbors is based on a latent semantic analysis vector space [13].

3 Neural Network Learning Method

Our extended model of hierarchical self-organizing maps – Enrich-GHSOM – represents the unsupervised neural network based learning solution adopted by our taxonomy enrichment framework. This choice is suitable to the knowledge structure to be enriched – a taxonomy, i.e. an *is-a* hierarchy of concepts.

3.1 Self-organizing Maps

GHSOM is an extension of the Self-Organizing Map (SOM, also known as Kohonen map) learning architecture [12, 5], which is one of the most popular unsupervised neural network models. SOM can be seen as a projection method which maps a high dimensional data space into a lower dimensional one. The resulting lower dimensional output space is a rectangular SOM map, represented as a two-dimensional grid of neurons. Each input data item is mapped into one of the neurons in the map. SOM is also a clustering method, so that similar data items – represented as vectors of numerical values – tend to be mapped into nearby neurons.

The SOM map learns by a self-organization process. There is no initial knowledge about the membership of any input data item in a particular class or about the number of classes. The training proceeds with unlabeled input data like any unsupervised learning. Clusters (classes) are discovered and described by gradually detected characteristics during the training process. These gradually adjusted characteristics play the role of weights in the weight vector associated to each neuron. The role of a completely trained map is to represent all the available observations – the whole input data space – with optimal accuracy by using a restricted set of weight vectors associated to the map neurons.

The initial values for the weight vectors of the neurons can either be chosen depending on the problem domain or they can be taken randomly. Every iteration of the learning algorithm processes one input (training) vector as follows. Like usually for unsupervised neural networks, some form of competitive learning takes place: the winner neuron index *c*, which best matches the current input vector, is identified as the neuron whose weight vector is most similar to the current input vector in some metric. Then all the weight vectors or a subset of them that correspond to neurons centered around the winner neuron *c* – i.e. neurons in the neighborhood area of *c* –, including the winner itself, are updated in the direction of the input vector. This adaptation renders a globally ordered map in the process of learning. A neuron has four immediate neighbors in a rectangular map topology, which is our chosen map topology. This is merely a rectangular lattice type of the two-dimensional grid of neurons, and the SOM map is kept as a planar rectangle.

3.2 Growing Hierarchical Self-organizing Maps

Data spaces contain some latent structuring in the form of clusters. SOM maps can discover and illustrate this clustering. However, some *hierarchical structures* are also latent in data sets. To give an interesting example in the present context, a thesaurus is a data space consisting of terms in a language, represented as a lexical database. The main relation between the terms in a thesaurus is the taxonomic relation. However, because of their essentially flat topology, SOM maps have a limited capability to discover and illustrate hierarchical clusters in data sets. A solution for this problem is represented by the *hierarchical SOM maps*.

The growing hierarchical self-organizing map model consists of a set of SOM maps arranged as nodes in a hierarchy and it is able to discover hierarchical clusters [8]. The SOM's in the nodes can grow horizontally during the training by inserting either one more row or one more column of neurons. This happens iteratively until the average data deviation (quantization error) over the neurons in the SOM map decreases under a specified threshold τ_1. For one neuron, the quantization error is the dissimilarity of all the vectors of the data items mapped into the neuron versus the weight vector of the neuron.

The SOM's in the nodes can also grow vertically during the training, by giving rise to successor nodes. Each neuron in the SOM map could be a candidate for expansion into a successor node SOM map (see Fig. 1). The expansion takes place whenever the data deviation on the current neuron is over a threshold τ_2. This sounds like a zoom into the data subspace mapped into the parent neuron, because the successor SOM map is trained merely with data items in that subspace. Further node expansions continue recursively on successor nodes, and the training of the whole GHSOM model finally stops (converges) when both thresholds are satisfied. The training begins with a single-neuron SOM map having the whole input data set mapped into its only neuron. This becomes the root of the final, completely trained GHSOM model.

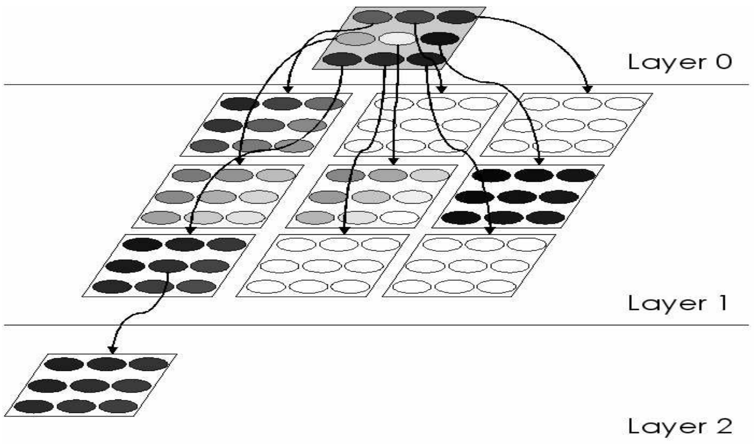


Fig. 1. The GHSOM neural network model.

The thresholds τ_1 and τ_2 control the granularity of the hierarchy learned by GHSOM in terms of depth and branching factor. A low τ_1 with a much lower τ_2 leads to a deep hierarchy with an increased number of neurons into the SOM nodes, and consequently an increased branching factor also. A high τ_1 with a lower τ_2 leads to deep hierarchies with small SOM nodes (with few neurons), and consequently a reduced branching factor corresponding to the reduced number of neurons in SOM nodes. When both thresholds are low and comparable, then the hierarchy is flat with a high branching factor. If both thresholds are high and comparable, then the hierarchy is flat with a low branching factor.

Each level in a learned GHSOM model displays a more detailed clustering of the data space as compared to the parent level. This corresponds to a top-down process of hierarchical clustering of the input data space items.

3.3 Enrich-GHSOM

The growth of a GHSOM is a completely unsupervised process, being only driven by the unlabeled input data items themselves together with the two thresholds and some additional learning parameters. There is no way to suggest from outside any initial paths for the final learnt hierarchy. We have extended the GHSOM model with the possibility to force the growth of the hierarchy along with some predefined paths of a given hierarchy. Our new extended model, *Enrich-GHSOM*, is doing a classification of the data items into an existing taxonomic structure. This initial tree plays the role of an initial state for the tree-like neural network model. The classical GHSOM model grows during the training by only starting from a single node. The top-down growth in our extended model starts from a given initial tree structure and inserts new nodes attached as successors to any of its intermediate and leaf nodes.

In Enrich-GHSOM, the nodes of the predefined hierarchy are labeled with some data item labels from the input data space used for training. The training data items propagate top-down throughout the given tree hierarchy structure. When the propagation process hits a parent SOM of a tree node, then the weight vector of the corresponding parent neuron in that parent SOM is initialized with the data item vector of that successor node label. The weight vectors of the SOM neurons with no successor are initialized with random values. Then the training of that SOM proceeds by classifying the training data items against the initialized neurons. Training data items that are similar (distributionally similar as vectors) to the predefined initialized neurons are propagated downwards to the associated successor SOM nodes to continue the training (recursively) on that predefined successor SOM. Data items that are not similar to the initialized neurons are mapped to other, non-initialized, neurons in the same SOM, and they are not propagated downwards into the predefined hierarchy. They remain as mapped into that SOM, and are considered as classified into the parent neuron of that SOM, i.e. as successor of that parent.

For instance, consider the parent neuron of a current SOM node is labeled *mammal*, and there are two predefined successor nodes labeled *feline* and *bear*, which correspond to two predefined initialized neurons in the current SOM. Then the training data item vector *dog* is not similar to any of the two neuron initializer weight vectors associated to *feline* and *bear* (see Fig. 2, where the neuron initializers are marked with bold). So *dog* will remain as classified into that SOM – mapped on another, non-initialized neuron – i.e. as successor (hyponym) of *mammal* and twin of the existent nodes *feline* and *bear*. Also, a data item labeled *tiger* – similar with the weight vector of the predefined "*feline*" neuron – will be propagated into the associated predefined successor SOM map together with other terms that correspond to felines, which will all become direct or indirect hyponyms of the concept *feline*. The process continues top-down for all the SOM nodes in the predefined initial tree hierarchy, ending at the leaves. The data item vector representations of the labels of the given initial tree play the role of *predefined initializer weight vectors* of our neural model.

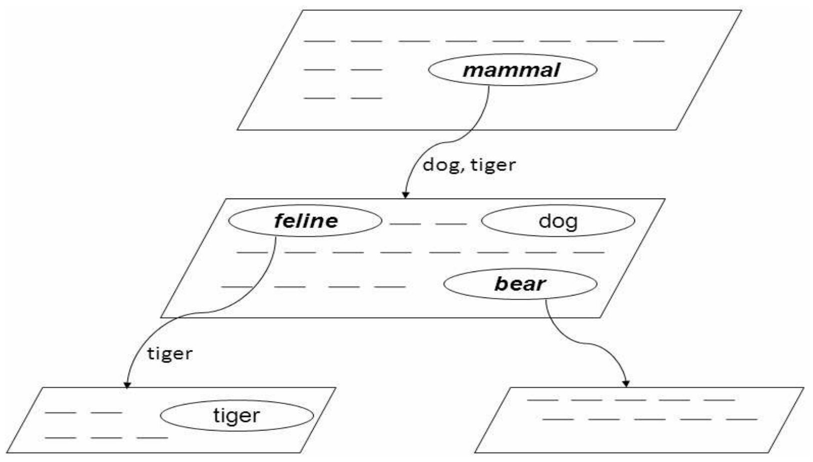


Fig. 2. The Enrich-GHSOM neural network model.

4 A Neural Model for Unsupervised Taxonomy Enrichment

The architecture of our framework is implemented as a pipeline with several linguistic and machine learning processing stages. The whole processing can be divided in two main steps: the *term extraction* step and the *taxonomy enrichment* step.

4.1 Extraction of Terms

The candidates for the labels of new concepts inserted during the taxonomy enrichment are terms representing *noun phrases*, identified by mining the domain text corpus. In order to identify the terms by a linguistic analysis of the corpus documents, our framework relies on several processing resources offered by the ANNIE module for analyzing English texts in the GATE framework [7]: morphological analyzer (stemmer), tokenizer, sentence splitter, the Hepple part-of-speech tagger, and a JAPE [7] transducer. The transducer has the role to identify noun phrase constructs, based on regular expressions over different parts of speech of the component words.

4.2 Taxonomy Enrichment

The terms extracted from the domain text corpus are mapped to classes (concepts) of the existing taxonomy. The taxonomy enrichment algorithm proceeds by "populating" the given taxonomy with the terms collected from the corpus. The *Enrich-GHSOM* neural network drives a top-down hierarchical classification of the terms along with the given taxonomy branches and inserts new nodes (concepts) corresponding to these classified terms. Every new concept is attached as successor of an intermediate or a leaf node of the given taxonomy and becomes a hyponym of that node.

In order to use our Enrich-GHSOM neural network to induce such a taxonomy enrichment behavior, a symbolic-neural translation is first done by parsing a textual representation of the initial taxonomy (*is_a*(*concept*, *superconcept*) assertions or OWL format). The result of this parsing is the initial internal tree-like state of the neural network. In order for the initialized network to be able to classify terms into this initial taxonomic structure, apart from the vector representation of the classified terms, a representation as a numerical vector is also needed for each node in the initial taxonomy. This vector plays the role of initial weight vector for the neural network (see section 3.3). It is the vector representation for the noun phrase concept label associated to the node, computed as will be described in section 4.3. The acquisition of this vector takes place in the same way as the acquisition of the vector representation of the classified terms (section 4.3).

We assume that the concept labels of the initial taxonomy are terms – noun phrases – extractable from the domain text corpus from which the classified terms themselves have also been extracted. Their vectors are then computed in the same way as the vectors of all the corpus extracted terms which are classified during the enrichment. Using the same corpus from a specialized domain to acquire the feature vectors of the concepts in the initial taxonomy and the terms to be classified is a reasonable choice, since it will reduce the problems with ambiguous (multiple) senses of one and the same term.

4.3 Vector Representation for Terms

Since Enrich-GHSOM is a connectionist system, the terms classified by Enrich-GHSOM and the concepts of the given taxonomy have to be represented as vectors. In our framework, the attributes (features) of the vector representation of a term or concept encode contextual content information, in a *distributional vector space*. Specifically, the context features are the frequencies of the occurrence of the term – classified term or concept label term – in different documents of the corpus. The number of component attributes of such a term vector coincides with the number of documents of the text corpus out of which all the terms have been extracted. Every attribute in the vector of a term is essentially the number of occurrences of the term in one document. This representation is inspired from the latent semantic analysis [13]. A similar semantics-based dimensionality reduction effect as the one obtained in the latent semantic analysis by singular value decomposition is achieved in our framework by the *document category histograms* (*DCH*), defined in what follows.

The vector representation in the current framework satisfies Harris' distributional hypothesis [6, 3]: the meaning of each classified term (or concept label) is related to the meanings of the contexts in which the term (or the concept label) occurs. In such a setting, we use the *distributional similarity* which asserts that the meaning of semantically similar terms and concept labels is expressed by similar vectors in the distributional vector space. The Euclidean distance is used in the current framework to compute the dissimilarity among vectors.

The framework allows multiple ways to encode the frequencies of occurrence, starting from simple *flat counts of occurrences*. Another variant is the *DF-ITF weighting scheme*, which means "document frequency times inverse term frequency". We propose this weighting scheme, which is a transposed of TF-IDF [2] relative to a term/document occurrence matrix. TF-IDF is used in document classification (text categorization) and information retrieval. Now we rather classify terms, by using DF-ITF. By using this weighting scheme, we consider that long documents, which talk about too many terms, should have a lower weight when classifying terms, since they have a reduced discrimination power among the meanings of different terms. This effect is achieved by our DF-ITF weighting scheme and is confirmed by the experimental results reported in section 5.

A third way to encode the vector representation is one in which we propose the vector to be a *document category histogram* (*DCH*). Specifically, first a SOM [12] is trained having the corpus documents as input data space to arrive at approximately 200 semantic document categories. Documents similar in meaning are clustered together by the unsupervised SOM neural network. In this SOM training, the documents are represented as vectors of frequencies for the terms they talk about. Equally like the term vectors, the document vectors are collected from the same term/document matrix, but after transposing this matrix. As we want a number of approximately 200 semantic document categories, we impose the training of a rectangular SOM map of dimension 16x12. Then, by summing up the frequencies of a term in different documents of the same category, and merely keeping the summed frequencies in different document categories as vector components, we arrive at a reduced dimensionality for the vector representation. In our experiments reported in section 5 with the "Lonely Planet" tourism data set, the reduction induced by such a vector representation as a histogram on semantic document categories is from 1801 (which represents the number of documents in the "Lonely Planet" corpus) to 180 and 179 (in two different experimental runs described in section 5).

Data Sparseness. The *dimensionality reduction* achieved by using the document category histogram (DCH) representation is important since it removes the semantic noise caused by minor differences in semantic content for different corpus documents. Such documents now belong together to the same semantic category. This intuition is already confirmed by our experiments reported in previous work [4]. Moreover, the term/document occurrence matrix is sparse (with many zeros), and reducing the dimensionality by using histograms leads to less sparse vectors. A more natural behavior of the neural network model is expected by using reduced and less sparse vectors.

A source of data sparseness is represented by *terms with very few occurrences* in the text corpus. Among such terms are the most generic terms that label the roots of the main trees in a given initial taxonomy and usually the concepts which are very high in a taxonomy. When in the *Enrich-GHSOM* neural network such an overly generic term with a very sparse vector labels the concept of one of the roots, and also when using the flat count vector representation instead of the histogram representation, then the main tree rooted by that concept is unable to attract and classify a relevant quantity of training terms. Thus the top-down search during the classification is misled. It is the case of the root concepts *spatial_concept*, *intangible*, and *thing* in the ontology of the "Lonely Planet" tourism dataset used in the present experiments. Some of the branches of these main trees are populated by no training term, which leads to the *starvation* of the neural network. Starvation means that the neural network enters an infinite loop when trying to tune the quantization error on a neuron below the thresholds (see section 3.2). Many of our experiments which used a flat count vector representation failed by starvation. As opposed, all the experiments using the reduced, histogram vector representation (DCH) converged to a result.

A way of reducing the number of zeros in the vector representation of the generic terms that label the generic concepts in the initial taxonomy is the *centroid vector* [14, 6]. We have used the idea of centroid in the following way: the average vector of the vector representations of all the concepts in the sub-tree rooted by the given concept, including the root itself. Using the centroid representation method has led us to a significant improvement of the experimental results, partially reported in [4], where we rather proposed a similar approach: one of the more specific concepts in a main tree becomes a substitute for the too generic concept in the root of the tree. So, the label of every main tree root was one representative and more specific concept in the tree, for instance *course* was a substitute for *activity*, and *staff* was a substitute for *person* (in the "4 universities" domain). The improvement obtained by using the centroid vector representation for concepts is reported in [14, 6].

5 Experimental Results

The experiments carried out in what follows are in the tourism domain, consisting of a corpus and a given taxonomy (the "Lonely Planet dataset") [9]. The associated corpus consists of 1801 text descriptions of tourist destinations from different countries around the world.

5.1 Experimental Setup

In order for the corpus extracted terms to actually become domain specific concepts, they have to be noun phrases with enough frequency of occurrence in the domain specific corpus. In the term extraction process, we have set a threshold for the extracted noun phrases to occur in at least 0.5% of the number of documents in the corpus. Having set this threshold, we have extracted and acquired the corresponding numerical vector representations for 1241 noun phrases. These extracted terms are classified against the taxonomy of a tourism ontology consisting of 73 concepts, which is proposed in the PASCAL ontology learning and population challenge [9].

The evaluation of the enrichment means evaluating the quality of the mapping from corpus extracted terms into target concepts of the given initial taxonomy. An extracted term becomes a new concept added to the taxonomy, and it is attached as hyponym (successor) under its associated target node. In order to evaluate the taxonomy enrichment, we followed a *cross-validation strategy* [14, 16, 15]. In every experimental run, exactly one node in the given initial taxonomy of 73 concepts was removed from the taxonomy, together with the whole subtree rooted by that node. The classification process was run against the result taxonomy, and the position of the held out concept, as classified like any corpus extracted term is assessed. The correct (direct hit) classification of the concept corresponds to its initial position in the taxonomy before its removal. In other words, the concept should be mapped to a target concept which was its direct hypernym (parent node) before its experimental removal. The process should be repeated 72 times, for every concept in the taxonomy except its very root, named *root*. Actually we repeated this experimental run 43 times, since we only had corpus statistical data to build the distributional vector representation for 43 of the taxonomy concepts. (We need a statistical distributional vector for every term to be classified.)

5.2 Evaluation Measures

The most appropriate measure for evaluating the taxonomy enrichment task is the *learning accuracy*, defined and evaluated in [9, 6, 14, 1, 16]. By choosing this measure, we consider correct classifications of the new concepts with different levels of detail. For instance, the new concept *cat* can be mapped to the target concept *feline*, *carnivore*, *mammal* or *animal* with different levels of detail, as a consequence of different hypernym-hyponym taxonomic distances between the target concept as chosen by the system and the direct hypernym of the classified concept before its removal. Before removal, *cat* was direct hyponym of the *feline* concept. Classifying *cat* as *feline*, by associating it to the *feline* target concept is a direct hit, since *cat* is correctly a *direct hyponym* of *feline*, i.e. 100% classification accuracy. Though, classifying *cat* as *carnivore*, *mammal*, or *animal* are near hits, since *cat* is correct only as an *indirect hyponym* of *carnivore*, *mammal*, or *animal*, corresponding say to 50%, 30%, 20% classification accuracy respectively.

For a given classified term *i*, if *pi* is the target concept assigned (predicted) by the system, and *ci* the correct target concept according to the given initial taxonomy, the *learning accuracy* is the average over all the classified terms *i* of the function *LA*(*pi*, *ci*), where the function *LA* is defined as

$$LA(p, c) = \frac{\delta(top, a) + 1}{\delta(top, a) + \delta(a, c) + \delta(a, p) + 1} \quad \textbf{(1)}$$

top is the root of the taxonomy, and *a* is the least common subsumer of the concepts *p* and *c* (i.e. the most specific common hypernym of *p* and *c*). $\delta(a, b)$ is the taxonomic distance between the concepts *a* and *b*, i.e. the number of taxonomy edges to be traversed when going from the taxonomy node labeled *a* towards node *b*. This is the most used formula to compute the learning accuracy. In the context of the Pascal ontology learning and population challenge, it is actually called *symmetric learning accuracy*, and the term *learning accuracy* is used for a historically initial version of the learning accuracy measure, as introduced by [11]:

$$\begin{aligned} LA'(p, c) &= \frac{\delta(top, a) + 1}{\delta(top, c) + 1} && \text{if } p \text{ is ancestor of } c \text{ (then also } a = p) \\ LA'(p, c) &= \frac{\delta(top, a) + 1}{\delta(top, a) + 2 * \delta(a, p) + 1} && \text{otherwise} \end{aligned} \quad \textbf{(2)}$$

According to formulae (1) and (2) to compute both variants of the learning accuracy, the same number of edges in the taxonomic distance between the predicted and the correct target concept means a better accuracy when the edges are lower in the taxonomy. This is due to the intuition that the same number of edges between two concrete (lower in the taxonomy) concepts means an increased similarity (a reduced semantic distance), as compared to the same number of edges between two abstract concepts (higher in the taxonomy).

Another quantitative evaluation measure similar in spirit to the learning accuracy is the *edge measure*. It actually counts the average deviation (in terms of taxonomic distance) between the system predicted target concept and the correct one according to the given initial taxonomy. Consequently, as opposed to the first two learning accuracy measures (formulae (1) and (2)), the edge measure means a better classification for a lower edge measure value.

5.3 Evaluation Results

A first set of experimental runs is based on a document category histogram (*DCH*) vector representation for the extracted terms and concept label terms. Also, the concept label terms of the given initial taxonomy are represented using the *centroid* method for the whole sub-tree of a given concept node, as described in section 4.3. The improvements gained by using DCH and centroid are already confirmed qualitatively by our experiments reported in [4]. Furthermore, not only the training of the Enrich-GHSOM neural network is less efficient on flat count vectors with 1801 attributes (corresponding to the 1801 corpus documents) compared to the 180 attributes (for the 180 semantic document categories) in DCH's, but also using flat count (unreduced) vectors often leads to the starvation of the neural network.

In a second set of experiments, we first applied the *DF-ITF* weighting scheme on the flat count term vectors of 1801 attributes. The result vectors were then converted into *DCH* histograms, thus reducing the term vector dimension to 179.

[6] and [14] used the centroid vector to reperesent the concept nodes. [14] found out that their best results were achieved when taking into account only the first three levels of successors in the sub-tree of the concept in order to compute the centroid. The experiments in [6] consider only the direct successors of the concept to compute the centroid. Driven by these results, we ran a third set of experiments, in which we considered only the first level of successors to represent the centroid of any concept in the given taxonomy, like in [6]. We didn't also try the three-level version of [14], since the results would be similar with our results for whole sub-trees. This is because the average depth of the taxonomy to be enriched in our experiments is 4, and the majority of the nodes don't have sub-trees of depth greater than 3. In this third set of experiments we kept the DF-ITF and DCH settings like in the second experiment.

We evaluated the three learning accuracy measures on placing the 43 concepts in their actual position in the given initial ontology from the Pascal challenge [9]. The results are illustrated in Table 1.

Table 1. Learning accuracy of the taxonomy enrichment when using DCH, DF-ITF, and different variants of centroid.

Vector Representation	DCH	DF-ITF + DCH	DF-ITF + DCH
Concept Label Centroid	whole subtree centroid	whole subtree centroid	first-level centroid
Learning Accuracy	33.565%	39.654%	37.679%
Symmetric Learning Accuracy	33.742%	40.437%	38.016%
Edge Measure	3.023	2.651	2.907

All the three learning accuracy measures are considerably improved by using the DF-ITF weighting measure, and keeping the DCH histogram vector representation. These results prove that the quality of the enrichment is improved by using our contributed semantics based vector representations (DCH and DF-ITF) for the classified terms and the concept label terms in the initial taxonomy.

Another finding is that limiting the depth of the sub-concepts for the computation of the centroid vector representation for taxonomy concepts leads to a slight degradation of the learning accuracy. The experiments in [16] also confirm that using whole sub-trees to represent the centroid of the concepts improve the performance of the taxonomy enrichment.

Named Entity Classification. In a last set of experiments, instead of classifying terms represented by common noun phrases extracted from the "Lonely Planet" corpus, we rather classified noun phrases for proper names – i.e. named entities – extracted from the same corpus. The majority of the named entities occur few times in the corpus, and many of them only occur once, in a singe document. This is why, in the experiments reported in what follows, we have reduced the frequency threshold to zero. It was 0.5% in the preceding experiments (see section 5.1).

Having no more frequency threshold for the corpus extracted noun phrases, we found and extracted a total of 43006 noun phrases, compared to 1241 in the preceding three taxonomy enrichment experiments. Some of them are common nouns and the other are named entities. We will refer in what follows to this experiment as *the maximal experiment*. To reduce the dimensionality of the data, and consequently the inherent noise, one of our experiments was trying to keep only what is absolutely necessary for the classification. We kept a minimum of common noun phrases corresponding to the concept labels in the taxonomy, and a minimum of proper noun phrases representing the set of named entities asked to be classified in the PASCAL ontology learning and population challenge [9]. The total number of common and proper noun phrases extracted is reduced to 631. We will call this experimental run *the minimal experiment*.

We evaluated these last experiments automatically by using the PASCAL challenge site online evaluation system[1]. This evaluation system is based on a *gold standard*, i.e. an ontology populated with the set of named entities that are asked to be classified in the PASCAL challenge. In other words, the PASCAL competition target set of named entities are considered as correctly mapped to the different concepts in the gold standard ontology. In *the maximal experiment*, a number of 623 named entities extracted from the "Lonely Planet" corpus are classified against an ontology consisting of 74 concepts, which is proposed in the PASCAL challenge [9]. Actually there are much more named entities extracted by our framework, but only 623 of them are also included in the set of named entities asked to be classified in the PASCAL ontology learning and population challenge. In the *minimal experiment*, 417 named entities are classified into a taxonomy consisting of 96 concepts. Table 2 illustrates these last two experiments, as evaluated automatically with the PASCAL challenge online evaluation system.

There are two explanations for the lower classification quality values in the maximal experiment as compared to the minimal one. First, the minimal experiment uses the DCH histogram vector representation as compared to the flat counts of the maximal experiment, and second is the noise caused by the much bigger quantity of noun phrases classified in the maximal experiment – 43006 versus 631. Also, an explanation for an overall degraded quality of the *named entity classification* as

[1] http://olc.ijs.si/eval.html.

compared to the *taxonomy enrichment* in the preceding experiments is that the classified named entities have very low frequency of occurrence as compared to the classified terms (common nouns) from the taxonomy enrichment, and consequently they have a very sparse vector representation. This misleads their classification.

Table 2. Learning accuracy of the named entity classification.

Experiment	*maximal experiment*	*minimal experiment*
Vector Representation	flat counts	DCH
Concept Label Centroid	whole subtree centroid	whole subtree centroid
Learning Accuracy	22.3%	31.2%
Symmetric Learning Accuracy	21.2%	28.5%
Edge Measure	3.78	4.767

6 Conclusions and Further Work

We have presented an unsupervised top-down neural network based approach and framework for taxonomy enrichment. The framework can be applied to different domains and languages. The experimental results obtained in the "Lonely Planet" tourism domain prove that our contributed semantics based vector representations, i.e. the *document category histograms* and the *DF-ITF weighting scheme* are suitable for the task of taxonomy enrichment.

The comparison of taxonomy enrichment systems (and of named entity classifiers) is problematic. Different systems use different domains and, even for the same domain, they use different corpora of different sizes and different ontologies. [6] present such a comparison of existent systems, and the conclusion is that the classification quality degrades with the increase in the size of the ontology.

Another interesting point is that sometimes given taxonomic structures are not reflecting correctly some fine-grained meanings. For instance, in the initial taxonomy used in our experiments, *forest* is hyponym of *area*. However the context in which the term *forest* occurs in the corpus are rather specific to plants (*plant* concept), which is far in the taxonomy from *area*. Our system "incorrectly" classified *forest* as *plant*.

The data sparseness remains a problem for the task of taxonomy enrichment. Terms (or named entities) represented by sparse vectors have an increased chance to be wrongly classified, because of the reduced power of attraction towards the correct branches and nodes of the taxonomy. Thus the top-down search during the classification is misled, and this phenomenon is mostly encountered in the case of named entity classification, where named entities have very sparse vector representations. Consequently, as further work, we will try to change the statistical distributional vector representation of the terms to further reduce the dimensionality of the vectors. We will try using pseudo-syntactic dependencies as representation of the terms, in the spirit of [6].

References

1. Alfonseca, E., Manandhar, S.: Extending a lexical ontology by a combination of distributional semantics signatures. In A. Gómez-Pérez, V.R. Benjamins (Eds.), *13th* International Conference on Knowledge Engineering and Knowledge Management, LNAI. Springer, pp. 1-7 (2002)
2. Buitelaar, P., Cimiano, P., Grobelnik, M., Sintek, M.: Ontology learning from text. Tutorial at ECML/PKDD workshop on Knowledge Discovery and Ontologies (2005)
3. Buitelaar, P., Cimiano, P., Magnini B.: Ontology learning from text: an overview. In P. Buitelaar, P. Cimiano, B. Magnini (Eds.), Ontology Learning from Text: Methods, Evaluation and Applications, Frontiers in Artificial Intelligence and Applications Series. IOS Press, pp. 1-10 (2005)
4. Chifu, E.Şt., Leţia, I.A.: Unsupervised ontology enrichment with hierarchical self-organizing maps, In: IEEE 2nd International Conference on Intelligent Computer Communication and Processing, pp. 3-9, IEEE Press, Cluj-Napoca (2006)
5. Chifu, E.Şt., Leţia, I.A.: Web mining with self-organizing maps, 8th IEEE International Conference on Intelligent Engineering Systems, pp. 93-98 (2004)
6. Cimiano, P., Völker, J.: Towards large-scale, open-domain and ontology-based named entity classification. In *RANLP'05,* International Conference on Recent Advances in Natural Language Processing, pp. 166-172 (2005)
7. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: GATE: a framework and graphical development environment for robust NLP tools and applications. In 40th Anniversary Meeting of the ACL (2002)
8. Dittenbach, M., Merkl, D., Rauber, A.: Organizing and exploring high-dimensional data with the Growing Hierarchical Self-Organizing Map. In L. Wang, *et al.* (Eds.), 1st International Conference on Fuzzy Systems and Knowledge Discovery, vol. 2, pp. 626-630 (2002)
9. Grobelnik, M., Cimiano, P., Gaussier, E., Buitelaar, P., Novak, B., Brank, J., Sintek, M.: Task description for PASCAL challenge. Evaluating ontology learning and population from text (2006)
10. Hearst, M.A.: Automatic Acquisition of Hyponyms from Large Text Corpora. In: 14th International Conference on Computational Linguistics, pp. 539-545 (1992)
11. Hahn, U., Schnattinger, K.: Towards text knowledge engineering. In: 15th National Conference on Artificial Intelligence and the 10th Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI), pp. 524-531 (1998)
12. Kohonen, T., Kaski, S., Lagus, K., Salojärvi, J., Honkela, J., Paatero, V., Saarela, A.: Self-organization of a massive document collection. IEEE Transactions on Neural Networks 11, pp. 574-585 (2000)
13. Landauer, T., Dumais, S.: A solution to Plato's problem: the latent semantic analysis theory of acquisition, induction and representation of knowledge. Psychological Review 104, 211–240 (1997)
14. Pekar, V., Staab, S.: Taxonomy learning – factoring the structure of a taxonomy into a semantic classification decision. In COLING'02, 19th International Conference on Computational Linguistic*s*, pp.786-792 (2002)
15. Widdows, D.: Unsupervised methods for developing taxonomies by combining syntactic and statistical information. In HLT-NAACL Conference, pp. 197-204 (2003)
16. Witschel, H.F.: Using decision trees and text mining techniques for extending taxonomies. In Learning and Extending Lexical Ontologies by using Machine Learning Methods, Workshop at ICML-05, pp. 61-68 (2005)

Genetic Algorithms for RDF Query Path Optimization

Alexander Hogenboom, Viorel Milea, Flavius Frasincar, and Uzay Kaymak

Erasmus School of Economics, Erasmus University Rotterdam
P.O. Box 1738, 3000 DR Rotterdam, The Netherlands
alexander.hogenboom@gmail.com
{milea, frasincar, kaymak}@few.eur.nl

Abstract. In this paper we present an approach based on genetic algorithms for determining optimal RDF query paths. The performance of this approach is benchmarked against the performance of a two-phase optimization algorithm. For more complex queries, the genetic algorithm RDFGA generally outperforms two-phase optimization in solution quality, execution time needed, and consistency in performance. Setting a time limit improves the overall performance of RDFGA compared to two-phase optimization even more.

1 Introduction

The potential of the Semantic Web has been demonstrated by different proof-of-concept applications, generally focussing on small domains. This limited focus, however, results in a Semantic Web that seems to be scattered into small pieces. Being available only on a small scale and for very specific domains, the access to the Semantic Web seems rather limited from the perspective of the average user.

Addressing the average user could be achieved by offering something that the current Web cannot offer: the possibility to query significant heaps of data from multiple heterogeneous sources more efficiently, returning more relevant results. In the context of the Semantic Web, the keyword is meta-data: describing the context of data and enabling a machine to interpret it. Semantic data is commonly represented using the Resource Description Framework (RDF), a World Wide Web Consortium (W3C) framework for describing and interchanging meta-data [1].

Despite current efforts, a successful implementation of an application that is able to query multiple heterogenous sources still seems far away. An interesting research field in this context is the determination of query paths: the order in which the different parts of a specified query are evaluated. The execution time of a query depends on this order. A good algorithm for determining the query path can thus contribute to quick and efficient querying.

In the context of the Semantic Web, some research in this field has already been done: the iterative improvement (II) algorithm followed by simulated annealing (SA), also referred to as the two-phase optimization (2PO) algorithm, addresses the optimal determination of query paths [2]. This implementation aims at optimizing the query path in an RDF query engine. However, other algorithms have not yet been used for RDF query path determination, while genetic algorithms (GA) have proven to be more effective than SA in cases with some similar characteristics. For example, a GA performed better than SA in solving the circuit partitioning problem, where components have to be placed on a chip in such a way, that the number of interconnections is optimized [3]. The query path determination problem is somewhat similar to this problem, since the distinctive parts of the query have to be ordered in such a way, that the execution time is optimized. Furthermore, genetic algorithms have proven to generate good results in traditional query execution environments [4]. Therefore, we seek to apply this knowledge from traditional fields to an RDF query execution environment, which differs from traditional ones in that the RDF environment is generally more demanding when it comes to response time; entirely new queries should be optimized and resolved real-time. In the traditional field of query optimization for relational databases, queries considered for optimization tend to be queries which are used more frequently and/or queries for which the duration of the optimization process is not that big of an issue.

The main goal we pursue consists of investigating whether an approach based on genetic algorithms performs better than a two-phase optimization algorithm in determining RDF query paths. The current focus is on the performance of such algorithms on a single source, and not in a distributed setting.

The outline of this paper is as follows. In Section 2 we provide a discussion on RDF and query paths, the optimization of which is discussed in Section 3. Section 4 introduces the genetic algorithm employed for the current purpose. The experimental setup and obtained results are detailed in Section 5. Finally, we conclude in Section 6.

2 RDF and Query Paths

Essentially, an RDF model is a collection of facts declared using RDF. The underlying structure of these facts is a collection of triples, each of which consists of a subject, a predicate and an object. These triples can be visualized using an RDF graph: *"a node and directed-arc diagram, in which each triple is represented as a node-arc-node link"* [1]. The relationship between a subject node and an object node in an RDF graph is defined using an arc which denotes a predicate. This predicate indicates that the subject has got a certain property, which refers to the object.

An RDF query can be visualized using a tree. The leaf nodes of such a query tree represent inputs (sources), whereas the internal nodes represent relational algebra operations, which enable a user to specify basic retrieval requests on these sources [5]. The nodes in a query tree can be ordered in many different ways, which all produce the same result. These solutions all depict an order in which operations are executed in order to retrieve the requested data and are referred to as query plans or query paths.

When querying RDF sources is regarded as querying relational databases, computing results for paths from partial results resembles computing the results of a chain query. In a chain query, a path is followed by performing joins between its sub paths of length 1 [2]. In the context of the Semantic Web, such queries can be expressed as a set of node-arc-node patterns which can be chained (joined). Each arc is to be interpreted as a predicate. Each node represents a concept and is to be interpreted as a subject associated with the predicate following this node and as an object associated with the predicate preceding this node. The join condition used in joining the node-arc-node patterns is that the object of the former pattern equals the subject of the latter pattern.

In an RDF context, bushy and right-deep query trees can be considered [2]. In bushy trees, base relations (containing information from one source) as well as results of earlier joins can be joined. Right-deep trees, which are a subset of bushy trees, require the left-hand join operands to be base relations. See Figure 1 for an example of a bushy tree and a right-deep tree, where concepts $(c_1, c_2, c_3, c_4, c_5, c_6, c_7)$ are joined and a $\bowtie$ represents a join.

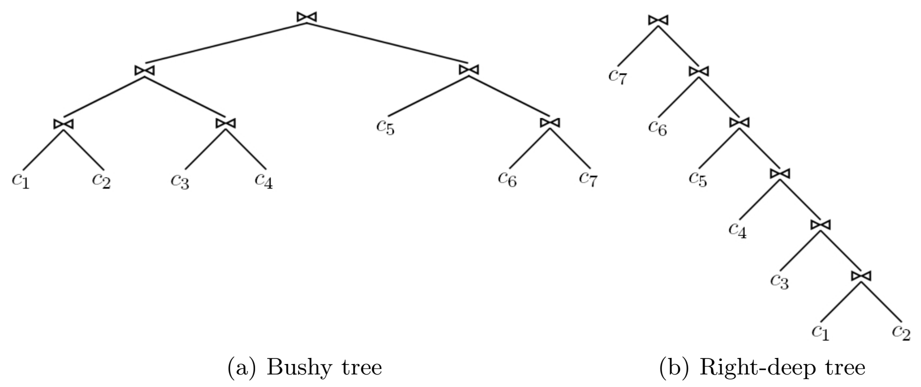


Fig. 1. Examples of possible trees

3 RDF Query Path Optimization

The order of joins of sub paths in a query path is variable and affects the time needed for executing the query. In this context, the join-order problem arises. The challenge is to determine the right order in which the joins should be computed, hereby optimizing the overall response time. In this process, each join is associated with costs, which are influenced by the number of elements in each operand (their cardinalities) and the method used in the join operation. Several methods can be used for implementing (two-way) joins, as discussed in [5].

The relevance of query path optimization can be demonstrated using a simplified example, in which only the number of results a join yields is considered in determining costs associated with that join. Let us consider an RDF model of the CIA World Factbook [6] containing various data about 250 countries, defined in over $100,000$ statements, generated using QMap [7]. Suppose a company, currently located in South Africa, wants to expand its activities to a country already in a trading relationship (in this example an import partnership) with South Africa. In order to assess the risks involved, the board wants to identify the candidates that have one or more neighbours involved in an international dispute. This query can be expressed in SPARQL, an RDF query language, in the following way:

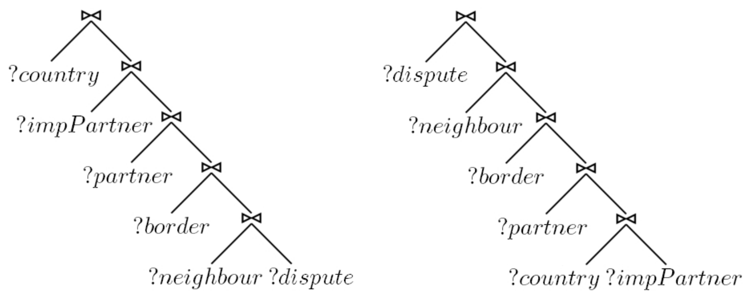
```
PREFIX ont: <http://www.daml.org/2003/09/factbook/factbook-ont#>
SELECT ?partner
WHERE { ?country ont:conventionalShortCountryName ?countryName .
        FILTER regex(?countryName, "^south africa$", "i") .
        ?country ont:importPartner ?impPartner .
        ?impPartner ont:country ?partner .
        ?partner ont:border ?border .
        ?border ont:country ?neighbour .
        ?neighbour ont:internationalDispute ?dispute .
      }
```

This query is a simple example of a chain query and can be subdivided into five parts: the query for information on the import partners of the specified country, the query for countries actually associated with other countries as import partners, the query for the borders of the latter countries, the query for countries associated with a country border as neighbours, and finally the query for the international disputes the neighbouring countries are involved in. The results of these sub queries can be joined in order to resolve the complete query. Here, the number of statements resulting from a join is equal to the number of statements compliant with both operands' constraints.

In this case, the collection of considered concepts is (?*country*, ?*impPartner*, ?*partner*, ?*border*, ?*neighbour*, ?*dispute*). The model contains 226, 1177, 186, 616, 186, and 548 elements respectively associated with these concepts. However, since the ?*country* concept is constrained to South Africa, the model only contains 1 compliant element.

An example of a query path consisting of joining the concepts in a particular order for this case is shown in Figure 2a. This query path starts with joining the last two concepts, yielding 181 compliant statements. These results are then joined with the ?*border* concept, which yields 2412 compliant statements. Joining these results with the ?*partner* concept yields 156 results. After a consecutive join of these results with the ?*impPartner* concept, 2434 statements are still compliant. A final join with the ?*country* concept yields 7 results. The sum of elements considered in every sub path thus equals 5190.

However, another order of joins is much more efficient. This order is depicted in Figure 2b. A first join of the first two concepts yields 8 results. Joining these results with the ?*partner* concept again yields 8 compliant statements. Joining



(a) Inefficient join-order (b) Efficient join-order

Fig. 2. Possible query paths for the international disputes case

these results with the ?*border* concept results in 38 triples satisfying all conditions. The model contains 33 triples compliant with a join between all previously joined concepts and the ?*neighbour* concept. Finally, a join between these resulting triples and the ?*dispute* concept yields 7 triples. The sum of elements considered in every sub path of this query path equals a mere 94. The order of joins of sub queries can thus make a big difference.

Two solution spaces can be distinguished for the join-order problem in an RDF context: a solution space consisting of bushy trees and a subset of that solution space, containing right-deep trees. The solution space of bushy trees contains $\binom{2n}{n}\frac{n!}{2^n}$ points representing possible permutations of join-orders, for a path length of n. There are 2^{n-1} possible query paths in the subset of right-deep trees [2]. Algorithms for identifying neighbouring solutions in the solution space differ per solution space [4]. If only right-deep query trees are considered, identifying neighbours can be done using the Swap algorithm or the 3Cycle algorithm [8]. However, if the complete solution space (containing bushy query trees) is considered, neighbouring solutions can be found by transforming a solution using transformation rules [9].

Since not every query path is as efficient as others, the challenge in determining which query path should be selected is to optimize query response time and/or execution costs. When utilizing a relational view on RDF sources, queries on these sources could be translated into algebraic expressions. Using some transformation rules for relational algebraic expressions, several heuristics for algebraic query optimization have been developed [5, 10].

However, in complex solution spaces, these simple heuristics are not sufficient; randomized algorithms (e.g. the iterative improvement algorithm and the simulated annealing algorithm) and genetic algorithms have proven to generate better results in traditional query execution environments [4]. Applying these algorithms in determining the order of *select* and *project* operations would not be very interesting due to the lack of complexity in the associated solution spaces and due to the sufficiency of the heuristics mentioned above. The real challenge lies in optimizing the order and nature of the joins, indicating randomized or genetic algorithms as promising approaches in this area.

In the context of the Semantic Web, the query path determination problem has already been addressed using an II algorithm followed by SA, also referred to as the two-phase optimization (2PO) algorithm [2]. The II algorithm randomly generates a set of initial solutions, which are used as starting points for a walk in the solution space. These walks only consist of steps to neighbouring points in the solution space that yield improvement. If no better neighbour can be found in a specified number of tries, the current point is assumed to be a local optimum. The number of times the algorithm tries to find a better neighbour (i.e. randomly selects a neighbour) is limited to the number of neighbours of that solution. The described process is repeated for all starting points.

The best local optimum thus found is subsequently used as a starting point for the SA algorithm, which tends to accept (with a declining probability) moves not yielding improvement. The latter algorithm thus searches the proximity of possibly sub-optimal solutions, hereby reducing the risk for a local optimum. Inspired by the natural process of annealing of crystals from liquid solutions, SA simulates a continuous temperature reduction, enabling the system to cool down completely from a specified starting temperature to a state in which the system is considered to be frozen. Just like II, the algorithm always accepts moves in the solution space yielding lower costs. However, SA can also accept moves leading to higher costs, hereby reducing the chances for the algorithm to get stuck in a local optimum. The probability for accepting such moves depends on the system's temperature: the higher the temperature, the more likely the system is to accept moves leading to higher costs. However, for every state of the algorithm applies that the more the costs associated with a solution exceed the current costs, the less likely the system is to accept such a move [8].

4 A Genetic Algorithm for Determining RDF Query Paths

As discussed in Section 1, GAs tend to perform better in query optimization. Based on these results, we propose a GA for determining RDF query paths: RDFGA. A GA is an optimization algorithm which simulates biological evolution according to the principle of survival of the fittest. A population (a set of chromosomes, representing solutions from the solution space) is exposed to evolution, consisting of selection (where individual chromosomes are chosen to be part of the next generation), crossovers (creating offspring by combining some chromosomes) and mutations (randomly altering some chromosomes). In this process, the fitness of a chromosome (expressing the quality of the solution) determines the chances of survival. Equation 1 depicts that higher the fitness F_s of a chromosome s in relation to the total fitness of n chromosomes, the bigger the probability that this chromosome and/or its offspring will make it to the next generation. Evolution is simulated until either the maximum number of iterations is reached or several generations have not yielded any improvement.

$$\mathrm{Pr}\,(\texttt{s selected}) = \frac{F_s}{\sum_{c=1}^{n} F_c} \quad (1)$$

Since a GA utilizes a randomized search method rather than moving smoothly from one solution to another, a GA can move through the solution space more abruptly than for example II or SA, by replacing parent solutions by offsprings that may be radically different from their parents. Therefore, a GA is less likely to get stuck in local optima than for example II or SA. However, a GA can experience another problem: crowding [11]. An individual with a relatively high fitness compared to others could reproduce quickly due to its relatively high selection probability, hereby taking over a large part of the population. This reduces the population's diversity, which slows further progress of the GA.

Crowding can be reduced by using different selection criteria, sharing a solution's fitness amongst similar solutions or controlling the generation of offspring. Another option is using a hybrid GA (HGA), which essentially is a GA with some additional, embedded heuristics. For instance, the initial population could be generated using heuristics for finding (sub-optimal) solutions, heuristics could be embedded in the crossover process or heuristics could (locally) optimize results generated by the crossover process. In these processes, local optimization techniques such as II could be used. However, high quality solutions are not guaranteed to be found within a reasonable running time, since the heuristics implemented in an HGA often are time-consuming [12]. A final strategy to reduce crowding is always selecting the fittest solution at least once (elitist selection) or by applying ranking-based selection [4], in which the probability of a solution s to be selected or used in a cross-over is determined by its rank R_s in relation to the sum of all n ranks (see equation 2). Here, the fittest solution is ranked best, whereas the least fit solution is associated with the worst rank.

$$\mathrm{Pr}\,(\texttt{s selected}) = \frac{R_s}{\sum_{c=1}^{n} R_c} \quad (2)$$

In order for a GA to be applicable in RDF query path determination, several parameters must be set. General settings, derived from literature, are discussed briefly in Section 4.1 before presenting suggestions for improving the performance of a GA in an RDF query execution environment. Since a GA is based on the principle of survival of the fittest, determining a solution's fitness is a crucial step in a GA. Section 4.2 discusses fitness determination and related issues. Finally, Section 4.3 provides a quick overview of the encoding scheme used for the current purpose to efficiently encode query paths.

4.1 Settings

Due to the time constraint associated with executing queries in an RDF environment, using an HGA is not an option, regardless of its potential of returning even better results than the algorithm used in [4]. This is because solutions of good quality are not guaranteed to be found within a reasonable amount of time, as discussed above. Therefore, it would be best to opt for a basic GA, adopting the settings best performing in [4].

The algorithm, BushyGenetic (BG), considers a solution space containing bushy query processing trees. A crowding prevention attempt is made by implementing ranking-based selection. Furthermore, the population consists of 128 chromosomes. The crossover rate is 65%, while the mutation rate equals 5%. The stopping condition is 50 generations without improvement. However, long executing times are not desirable for a GA in an RDF query execution environment. Therefore, the stopping condition is complemented with a time limit.

In literature, a GA has been proven to generate better results than a 2PO algorithm in many cases. However, in order to accomplish these results, a GA has turned out to be needing more execution time than 2PO. On the other hand, research did show that a GA is aware of good solutions faster than 2PO [4]. Hence, the algorithm spends a lot of time optimizing good results before it terminates. The latter property is an interesting property of GAs to exploit in RDFGA for the current purpose. Since in a real-time environment like the Semantic Web queries need to be resolved as quickly as possible, preliminary and/or quicker convergence of the model might not be such a bad idea after all, even though this increases the probability of outputting a sub-optimal result. If the model could somehow quickly converge in the final stage of optimization of good results, the execution time could be reduced remarkably and the sub-optimal result would not be too far from the global optimum. The challenge is to find a balance between execution time and solution quality.

The BG algorithm could be adapted in order to improve its performance in an RDF query execution environment. For instance, the algorithm could be forced to select the best solution for proliferation in the next generation at least once (elitist selection), hereby avoiding the loss of a good solution. Replacing ranking-based selection with fitness-based selection could be a subject of tests too in this case, since this increases the probability of relatively fit solutions to be selected, which could result in quicker convergence of the model due to increased crowding. Furthermore, evolution could be considered to have stopped after, e.g., 30 generations without improvement instead of 50; long enough in order for the algorithm to be able to state with sufficient certainty that the best known solution is either a very good local optimum or a global optimum, especially in solution spaces with a relatively small number of solutions (which is the case with smaller queries). Finally, the population size could be reduced to for example 64 solutions, which would noticeably reduce the time needed for computing the costs of all solutions in the population and would provide just enough room for diversity in the population (especially for smaller queries), hereby also enforcing quicker model convergence.

4.2 Determining a solution's fitness

In the context of RDF query path determination, let the fitness F_s of a solution s depend on its associated costs g_s. When ranking the solutions, the solution associated with the lowest costs should be associated with the highest rank and the solution associated with the highest costs should be associated with the lowest rank. In case of fitness-based selection, the probability of a solution to be selected (as defined in equation 1) must be inverse proportional to its associated costs [4]. This can be accomplished by defining the fitness F_s of solution s as shown in equation 3, hereby assuming that the population contains n solutions.

$$F_s = \frac{1 - \frac{g_s}{\sum_{c=1}^{n} g_c}}{n-1} \tag{3}$$

For the current goal, only nested-loop joins and hash joins are considered in the calculation of solution costs. No index or hash key exists for the source used here (making single-loop joins impossible) and the source data are unsorted (requiring the sort-merge join algorithm to sort the data first, hereby unnecessarily taking up precious running time).

When joining two operands, say c_1 and c_2, using a nested-loop join, the processing costs are $|c_1| \times |c_2| \times compC$, where $|c_1|$ and $|c_2|$ represent the cardinality of respectively operand c_1 and c_2 and $compC$ denotes the cost of comparing two elements. In case a hash join is used, the processing costs are $(insC \times |c_1|) + (retC \times |c_2| \times avgB)$, where $|c_1|$ and $|c_2|$ again represent the cardinality of respectively operand c_1 and c_2, $insC$ denotes the costs of inserting an element into the hash table, $retC$ represents the cost of retrieving a bucket (which contains elements) from the hash table and $avgB$ stands for the average bucket size [2]. In an RDF environment, cardinalities could be estimated, as actually performing the joins in order to retrieve the number of elements resulting from each join of sub paths would imply the execution time of the optimization process to be very likely to exceed the execution time of a random query path. Hence, we work with estimated cardinalities. These estimations could be updated after a query has been evaluated; computed join costs can be saved for possible re-use in order to reduce the time needed for evaluating joins.

4.3 Query path encoding

Encoding of query processing trees is done using an ordinal number encoding scheme for bushy trees, proposed in [4], which not only efficiently represents bushy trees (including the subset of right-deep trees), but enables relatively easy and efficient crossover operations as well. This encoding algorithm iteratively joins two concepts in an ordered list of concepts, the result of which is saved in the position of the first appearing concept. In each iteration, the positions of the selected concepts are saved into the encoded solution.

For example, consider the following ordered list of concepts: (c_1, c_2, c_3, c_4). An initial join between the third and fourth concept yields the list (c_1, c_2, c_3c_4). Another join between the first and second concept in this new list yields (c_1c_2, c_3c_4). A final join between the first and second concept in this list results in $(c_1c_2c_3c_4)$. A possible encoded notation of these joins is $((3,4),(1,2),(1,2))$. Additional information, such as the applied join method, can also be stored in this encoded notation. For details on the crossover and mutation methodology applied for the current goal, we refer to [4].

5 Experimental Setup & Results

5.1 Experimental Setup

All experiments performed for the current purpose are run in a Microsoft Windows XP environment, on a 2,400 MHz Intel Pentium 4 system with 1,534 MB physical memory (DDR SDRAM). Tests are conducted on a single source: an RDF version of the CIA World Factbook [6], generated using QMap [7]. The first algorithm to be tested is the 2PO algorithm as proposed in [2]. The performance of the BG algorithm [4] and its improved version (RDFGA) as proposed in Section 4.1 are benchmarked as well. Finally, the performance of time-constrained 2PO and RDFGA (respectively 2POT and RDFGAT, in which the T denotes the time-constrained nature of these algorithms) is evaluated.

Several experiments are conducted in order to determine the performance of the considered algorithms; each algorithm is tested on chain queries varying in length from 2 to 20 predicates (see Section 3 for a 6-predicate example). Each experiment is iterated 100 times, in order to increase the accuracy of the results. The parameters in cost determination, *compC*, *insC*, *retC* and *avgB*, are assigned random values of 0.02, 0.05, 0.05 and 5.0 respectively, since these exogenous variables are computer, programming language, and/or implementation dependent and hence would be hard to determine. Since these variables are exogenous, their values will not affect the way the algorithm works, so their exact values are not relevant for the goal pursued here.

The algorithms are configured according to the settings proposed in their sources and thus all consider the entire solution space containing bushy query trees. However, preliminary experimental results on the data set used in this research show that, ranking-based selection perform quicker and yield better results than fitness-based selection. Hence, we have decided to use the ranking based-selection method in this research for RDFGA. Furthermore, the time limit for 2POT and RDFGAT is set to 1000 milliseconds, since this allows the algorithms to perform at least a couple of iterations and since in practice, waiting 1 second in order to have your complex query executed quickly, would probably not be too long.

	2PO	2POT
maxSol	10	10
startTempFactor	0.1	0.1
tempRed	0.05	0.05
frozenTemp	1	1
maxConsRedNoImpr	4	4
neighbourExpFactor	16	16
timeLimit	-	1000

Table 1. Parameters of considered two-phase optimization algorithms

Table 1 presents an overview of the parameters of the 2PO algorithms considered. The *maxSol* parameter sets the maximum number of starting solutions analyzed in the II part of 2PO. The fraction of the optimal cost resulting from II to be used as starting temperature in SA is specified in *startTempFactor*, whereas *tempRed* is the factor with which the temperature of the system is to be reduced every iteration of SA. The *frozenTemp* parameter defines the temperature below which the system is considered to be frozen. The maximum number of consecutive temperature reductions not yielding improvement is defined in *maxConsRedNoImpr*. For each visited solution, SA tries to move to neighbouring solutions for a limited number of times, which equals the number of joins in the query, multiplied by *neighbourExpFactor*. Finally, the maximum running time in milliseconds is configured using the *timeLimit* parameter.

	BG	RDFGA	RDFGAT
popSize	128	64	64
crossoverRate	0.65	0.65	0.65
mutationRate	0.05	0.05	0.05
stableFitnessGens	50	30	30
rankingBased	true	true	true
elitist	false	true	true
timeLimit	-	-	1000

Table 2. Parameters of considered genetic algorithms

An overview of the parameters of the GAs is presented in Table 2. The number of chromosomes (solutions) to be subjected to a simulated biological evolution process is defined using the *popSize* parameter. The *crossoverRate* parameter represents which fraction of each new generation is to be filled with offspring resulting from crossover operations between pairs of randomly selected chromosomes. The rest of the new generation is filled with direct selections from the current generation. The fraction of the new population to be mutated is defined using the *mutationRate* parameter. Furthermore, *stableFitnessGens* is the number of consecutive generations not showing improvement in optimal fitness needed for the fitness of the population to be considered stable. The *ranking-Based* parameter is used to define whether ranking-based selection should be applied rather than fitness-based selection, whereas the *elitist* parameter states whether the best solution should always be selected for the next generation. The maximum running time in milliseconds is defined in *timeLimit*.

5.2 Results

For each algorithm tested, Figure 3 visualizes the average time needed for optimizing chain queries. The chain queries considered in the experiments vary in length from 2 to 20 predicates. The average execution times depicted in Figure 3 are based on 100 iterations of the query optimization process per experiment.

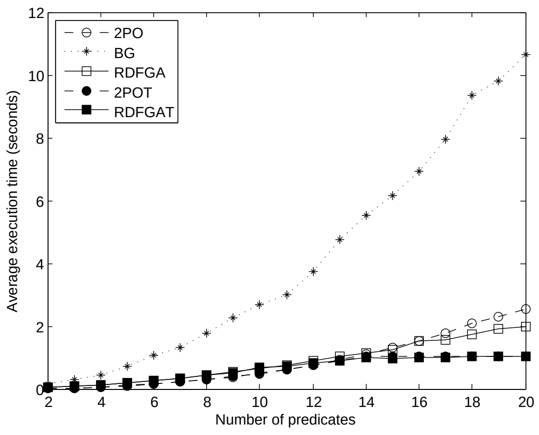


Fig. 3. Average execution times

For all considered query lengths, on average, BG needs the most execution time of all considered algorithms. Furthermore, 2PO turns out to be the fastest performing optimization algorithm for relatively small chain queries containing up to about 10 predicates. For the latter chain queries, on average, RDFGA performs slower than 2PO, but still needs less execution time than BG. For bigger chain queries, RDFGA is the fastest performing algorithm. However, the time-constrained variants of 2PO and RDFGA obviously take the lead for even bigger queries, where RDFGA's execution time exceeds the time limit.

For each algorithm that we consider, the average costs associated with the optimal solutions of chain queries varying in length from 2 to 20 predicates, based on 100 iterations of the query optimization process per experiment, do not appear to differ very much. However, a closer look to the relative deviations from the optimal solutions found by 2PO can reveal more clear indications of differences in performance. Without a time limit, both genetic BG and RDFGA tend to find lower cost solutions, especially for larger queries. When a time limit for query optimization is set, a GA tends to generate even better results compared to 2PO, as shown in Figure 4. The known behaviour of both algorithms supports this observation, since a GA tends to generate better results in less time, although it needs more time to converge than a 2PO algorithm (as discussed in Section 4.1). Therefore, the earlier in the optimization process both algorithms are forced to stop, the better the result of a GA will be compared to the solution generated by 2PO.

The consistency in performance is shown in Figures 5 and 6, using coefficients of variation (standard deviation, expressed in relation to the mean) of the execution times and optimal solution costs, respectively, of chain queries of varying

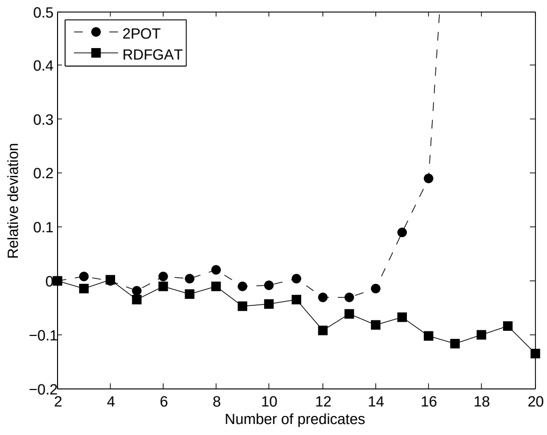


Fig. 4. Relative deviation of average optimal costs from 2PO average

lengths. These statistics are based on 100 iterations of the query optimization process per experiment. A coefficient of variation close to 0 indicates all observed values are closely clustered around the average. Hence, the higher the coefficient of variation, the less consistent the performance of the algorithm.

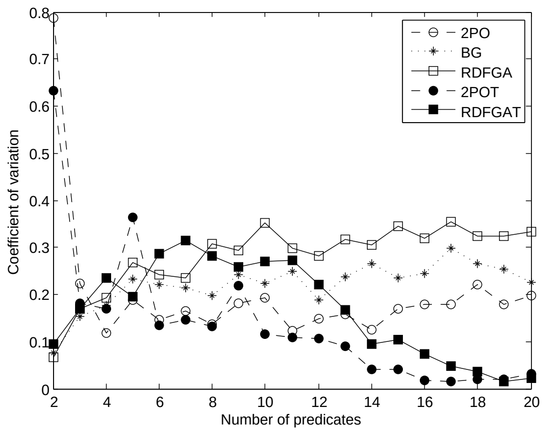


Fig. 5. Coefficients of variation of execution times

The coefficients of variation of the execution times for chain queries of different lengths indicate that time-constrained algorithms tend to perform more and more consistently for bigger chain queries. This observation can be explained by realizing bigger chain queries require longer execution times, which are increasingly likely to exceed the time limit. Hence, increasing parts of iterations of

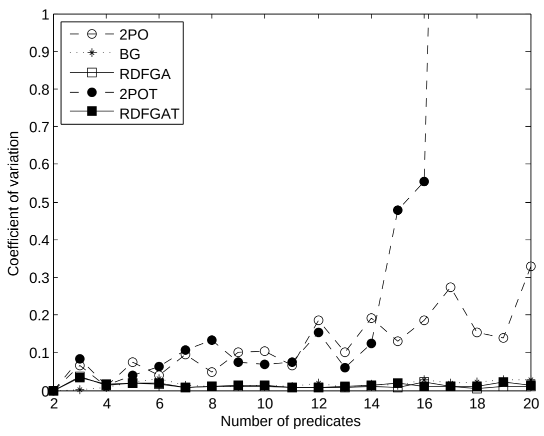


Fig. 6. Coefficients of variation of optimal costs

bigger queries execute exactly as long as allowed, hereby reducing the variance in execution times. As for the algorithms not constrained by a time limit, the GAs appear to be less consistent in execution time needed than 2PO, especially for more complex queries.

The 2PO algorithm shows a higher coefficient of variation of optimal costs than BG and RDFGA. Also, the more predicates a chain query consists of, the higher the coefficient of variation of optimal costs. When a time limit is set, the coefficient of the 2PO algorithm increases rapidly with the number of predicates chain queries consist of. GAs on the other hand show a constantly low coefficient of variation of optimal costs. The results of RDFGA are not clearly affected by a time limit.

6 Conclusions

The results detailed in this paper lead to the conclusion that in determining the (optimal) query path in a single-source RDF query execution environment, a correctly configured genetic algorithm can outperform the two-phase optimization algorithm in i) solution quality, ii) execution time needed, and iii) consistency in performance, especially for more complex solution spaces. The superiority of genetic algorithms relative to the two-phase optimization algorithm becomes more clear in positive correlation with the restrictiveness of the environment (e.g. a time limit) and the complexity of the solution space. However, it should be noted that in less complex solution spaces, a genetic algorithm performs worse compared to the two-phase optimization algorithm when it comes to execution time. Furthermore, in some cases, the optimization process could take longer than the actual execution of a query. This falls outside the scope of this paper, but the total query execution process deserves more detailed study and should be considered for further research.

Acknowledgement

The authors are partially supported by the EU funded IST STREP Project FP6 - 26896: Time-determined ontology-based information system for realtime stock market analysis. More information is available on http://www.towl.org.

References

1. Klyne, G., Carroll, J.: Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation (2004)
2. Stuckenschmidt, H., Vdovják, R., Broekstra, J., Houben, G.J.: Towards Distributed Processing of RDF Path Queries. International Journal of Web Engineering and Technology 2(2-3), 207–230 (2005)
3. Manikas, T.W., Cain, J.T.: Genetic Algorithms vs. Simulated Annealing: A Comparison of Approaches for Solving the Circuit Partitioning Problem. Technical report, Univerisy of Pittsburgh (1996)
4. Steinbrunn, M., Moerkotte, G., Kemper, A.: Heuristic and Randomized Optimization for the Join Ordering Problem. The VLDB Journal 6(3), 191–208 (1997)
5. Elmasri, R., Navathe, S.B.: Fundamentals of Database Systems. 4th edn. Addison-Wesley (2004)
6. Central Intelligence Agency: The CIA World Factbook (2007) See https://www.cia.gov/cia/publications/factbook/, last visited April 2007.
7. Hogenboom, F., Hogenboom, A., van Gelder, R., Milea, V., Frăsincar, F., Kaymak, U.: QMap: An RDF-Based Queryable World Map. In: Third International Conference on Knowledge Management in Organizations (KMO 2008), pp. 99–110 (2008)
8. Swami, A., Gupta, A.: Optimization of Large Join Queries. In: The 1988 ACM SIGMOD International Conference on Management of Data (SIGMOD 1988), pp. 8–17 ACM Press, New York, NY, USA (1988)
9. Ioannidis, Y.E., Kang, Y.C.: Randomized Algorithms for Optimizing Large Join Queries. In: The 1990 ACM SIGMOD International Conference on Management of Data (SIGMOD 1990), pp. 312–321 ACM Press, New York, NY, USA (1990)
10. Frăsincar, F., Houben, G.J., Vdovjak, R., Barna, P.: RAL: An Algebra for Querying RDF. World Wide Web Journal 7(1), 83–109 (2004)
11. Mitchell, T.M.: Machine Learning. McGraw-Hill Series in Computer Science. McGraw-Hill (1997)
12. Misevicius, A.: A Fast Hybrid Genetic Algorithm for the Quadratic Assignment Problem. In: The 8th Annual Conference on Genetic and Evolutionary Computation (GECCO 2006), pp. 1257–1264 ACM Press, New York, NY, USA (2006)

Optimizing Ontology Alignments by Using Genetic Algorithms

Jorge Martinez-Gil, Enrique Alba, and José F. Aldana-Montes

Universidad de Málaga, Departmento de Lenguajes y Ciencias de la Computación
Boulevard Louis Pasteur s/n 29071 Málaga (Spain)
{jorgemar,eat,jfam}@lcc.uma.es
http://www.lcc.uma.es

Abstract. In this work we present GOAL (Genetics for Ontology Alignments) a new approach to compute the optimal ontology alignment function for a given ontology input set. Although this problem could be solved by an exhaustive search when the number of similarity measures is low, our method is expected to scale better for a high number of measures. Our approach is a genetic algorithm which is able to work with several goals: maximizing the alignment precision, maximizing the alignment recall, maximizing the f-measure or reducing the number of false positives. Moreover, we test it here by combining some cutting-edge similarity measures over a standard benchmark, and the results obtained show several advantages in relation to other techniques.

Key words: ontology alignment; genetic algorithms; semantic integration

1 Introduction

The Semantic Web is a new paradigm for the Web in which the semantics of information is defined, making it possible for the web to understand and satisfy the requests of people and machines to use the web resources. Therefore, most authors consider it as a vision of the Web from the point of view of an universal medium for data, information, and knowledge exchange [1].

In relation to knowledge, it is very important the notion of ontology as a form of representation about a particular universe of discourse or some part of it. Ontology alignment is a key aspect in order to the knowledge exchange in this extension of the Web may be real; it allows organizations to model their own knowledge without having to stick to a specific standard. In fact, there are two good reasons why most organizations are not interested in working with a standard for modelling their own knowledge: (a) it is very difficult or expensive for many organizations to reach a agreement about a common standard, and (b) these standards do not often fit to the specific needs of the all participants in the standarization process.

Altought ontology alignment is perhaps the most valuable way to solve the problems of heterogeneity and, even there are a lot of techniques for aligning ontologies in a very accurate manner, experiences tells us that the complex nature of the problem to be solved makes difficult that these techniques operate in a satisfactory way for all kinds of data, in all domains, and as all users expect. This problem has been studied in [2].

As a result, techniques that combine existing methods have appeared. The goal of these techniques is to obtain more complex and accurate matching algorithms. The way to combine these matching algorithms is under an exhaustive research now. The most promising mechanisms are reviewed in the Section 6, but we can advance that the use of Genetic Algorithms (GAs) has been studied in little depth by researchers. Therefore, the main contributions of this work are:

– The proposal of an efficient mechanism, other than those that already exist, to compute the optimal function for aligning arbitrary sets of ontologies.
– The additional possibility to obtain goal-driven results, thus optimize some of the characteristics of an output alignment.
– We provide results following a standard benchmark to enable the comparison with other approaches.

The rest of this work is structured in the following way: Section 2 describes the problem statement. Section 3 presents the technical preliminaries which are neccesary to our approach. Section 4 discusses our aproach. Section 5 findings extracted from several experiments, including the use of a benchmark provided by the Ontology Alignment Evaluation Initiative [3]. Section 6 compares our results with other proposals. Finally, we remark the strengths and flaws of our proposal and discuss the future work in Section 7.

2 Problem Statement

The process of aligning ontologies can be expressed as a function f where given a pair of ontologies o and o', an partial (and optional) input alignment A, a set of parameters p and a set of resources r, returns a new alignment A':

$$A' = f(o, o', A, p, r)$$

A' is a set of mappings. A mapping is an expression that represents a semantic correspondence between two entities. A mapping is the atomic component of an alignment and is a formalism that allows to share knowledge models created separately.

However, experience tells us that getting f is far from trivial. As we commented earlier, the heterogeneity and ambiguity of data descriptions makes unrealistic the scenario in which that optimal mappings for many pairs of entities will be considered as "best mappings" by any of the existing matching algorithms. For instance, the Fig. 1 shows an alignment that is valid for users from some countries, but not for some others. The current trend is to diversify (and possibly weight) the matching algorithms. To do it, it is neccesary to use composite ontology matchers.

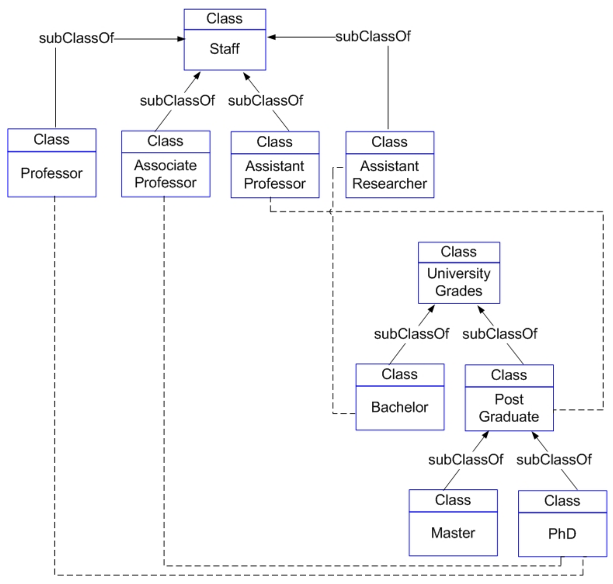


Fig. 1. Example of alignment between two ontologies. Most probably none of the two ontology owners will consider it optimal for them

Composite matchers are aggregation of simple matchers which exploit a wide range of information, in fact, we can classify the matching algorithms in the following types:

1. **String normalization.** This consists of methods such as removing unnecessary words or symbols from the entity names. Moreover, they can be used for detecting plural nouns or to take into account common prefixes or suffixes as well as other natural language features.
2. **String similarity.** Text similarity is a string based method for identifying similar entity names. For example, it may be used to identify identical concepts of two ontologies if they have a similar name. The reader can see [4] for more details about this algorithms.
3. **Data Type Comparison.** These methods compare the data type of the ontology elements. Similar concept attributes are logically expected to have the same data type.
4. **Linguistic methods.** This consists in the inclusion of linguistic resources such as lexicons and thesauri to identify possible similarities. The most popular linguistic method is to use WordNet [5] to identify some kinds of relationships between entities.
5. **Inheritance analysis.** Theses kinds of methods take into account the inheritance between concepts to identify relationships. The most popular method is the *is-a* analysis that tries to identify subsumptions between concepts.
6. **Data analysis.** These kinds of methods are based on the rule: If two concepts have the same instances, they will probably be similar. Sometimes, it

is possible to identify the meaning of an upper level entity by looking at a lower level entity. For example, if instances contain a string such as *years old*, it probably belongs to an attribute called *age*.

7. **Graph-Mapping.** This consists in identifying similar graph structures in two ontologies. These methods use known graph algorithms to do so. Most of times this involves computing and comparing paths, adjacent nodes and taxonomy leaves.
8. **Statistical analysis.** It consists of the extraction of keywords and textual descriptions for detecting the meaning of the entities in relation to other entities.
9. **Taxonomy analysis.** It tries to identify similar concepts by looking at their related concepts. The main idea is that two concepts belonging to different ontologies have a certain degree of probability of being similar if they have the same neighbours.

The main idea of composite matchers is to combine similarity values predicted by multiple simple algorithms to determine correspondences between entities belonging to different ontologies. The most popular proposals in this field are COMA [6], COMA++ [7], QuickMig [8], FOAM [9], iMAP [10] and OntoBuilder [11]. But these proposals use, in the best of the cases, weigths determined by an expert. Our work does not use weights from an expert, but compute those for obtaining the optimum alignment function so that the problem can be solved accuarately and without requiring human intervention.

3 Technical Preeliminaries

Definition 1 (Similarity measure). *A similarity measure sm is a function* $sm : \mu_1 \times \mu_2 \mapsto \Re$ *that associates the similarity of two input ontology entities* μ_1 *and* μ_2 *to a similarity score* $sc \in \Re$ *in the range [0, 1], where a similarity score of 0 stands for complete inequality and 1 for complete equality of the input ontology entities* μ_1 *and* μ_2.

Definition 2 (Weighted similarity measure). Let $\boldsymbol{A}$ be a set of well-known similarity measures and $\boldsymbol{w}$ a numeric weight vector, and let O_1, O_2 be two input ontologies, then we can define *wsm* as a weighted similarity measure in the following form:

$$wsm(O_1, O_2) = x \in [0,1] \in \Re \rightarrow \exists \langle \boldsymbol{A}, \boldsymbol{w} \rangle, x = max(\textstyle\sum_{i=1}^{i=n} A_i \cdot w_i)$$
$$subject\ to\ \textstyle\sum_{i=1}^{i=n} w_i \leq 1$$

From an engineering point of view, this function leads to an optimization problem for calculating the numeric weight vector, because the number of candidates from the solution space (in this case an arbitrary continous interval) is infinite. Hence, exact techniques are of low help here, and we are interested in methods such metaheuristics (e.g.g genetic algorithms) that find quasi optimum results in such solution spaces.

Definition 3 (Ontology alignment). *An ontology alignment oa is a set of tuples* $\{(id, e, e', n, R)\}$*. Where id is an unique identifier of the mapping, e and* e' *are entities belonging to two different ontologies, R is the relation of correspondence between these entities and n is a real number between 0 and 1 that represents the mathematical probability that R is true. The entities that are related are the concepts, roles, rules, and even axioms of the two ontologies.*

Definition 4 (Ontology matching function). *An ontology matching om is a function* $om : O_1 \times O_2 \overset{sm}{\rightarrow} A$ *that associates two input ontologies* O_1 *and* O_2 *to an alignment A using a similarity measure (or a weighted similarity measure).*

Definition 5 (Alignment evaluation). *An alignment evaluation ae is a function* $ae : A \times A_R \mapsto precision \times recall$ *that associates an alignment A and an reference alignment* A_R *to two real numbers in the interval* $[0, 1]$ *stating the precision and recall of A in relation to* A_R*.*

Code 1 shows an example of an output from an alignment evaluation process where two ontologies from a standard benchmark provided by the OAEI [3] have been aligned. Parameters will be discussed in more detail in Section 5.

Code 1 Example of Alignment Evaluation

```
<?xml version='1.0' encoding='utf-8' standalone='yes'?>
<rdf:RDF xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
 xmlns:map='http://.../projects/ontology/ResultsOntology.n3#'>
 <map:output rdf:about=''>
   <map:input1 rdf:resource="http://.../benchmarks/101/onto.rdf"/>
   <map:input2 rdf:resource="http://.../benchmarks/204/onto.rdf"/>
   <map:precision>1.0</map:precision>
   <map:recall>0.6288</map:recall>
   <fallout>0.0</fallout>
   <map:fMeasure>0.7721</map:fMeasure>
   <map:oMeasure>0.6288</map:oMeasure>
   <result>0.6288</result>
 </map:output>
</rdf:RDF>
```

4 Genetics for Ontology ALignments (GOAL)

We are beginning our research. First, we are going to consider GAs. Later, we may consider other approaches. GAs are often used to search along very high dimensional problems spaces. For example, if we want to find the maximum value of the function wsf with three independent variables w_0, w_1 and w_2:

$$wsf(O_1, O_2) =$$
$$w_0 \cdot \mathit{datatype}(O_1, O_2) + w_1 \cdot \mathit{normalization}(O_1, O_2) + w_2 \cdot \mathit{synonyms}(O_1, O_2)$$

where w_0, w_1 and w_2 are weights to determine the importance of the three respective similarity measures, which belong, for instance, to the continuous interval [0, 1]. The problem that we want to solve consists of finding a good value of w_0, w_1 and w_2 to find the largest possible value of wsf.

While this problem can be solved trivially by a brute force search over the range of the independent variables w_0, w_1 and w_2, the GA method scales very well to similar problems of a higher dimensionality; for example, we might have functions using a large number of independent variables w_0, w_1, w_2,..., w_n. In this case, an exhaustive search would be prohibitively expensive.

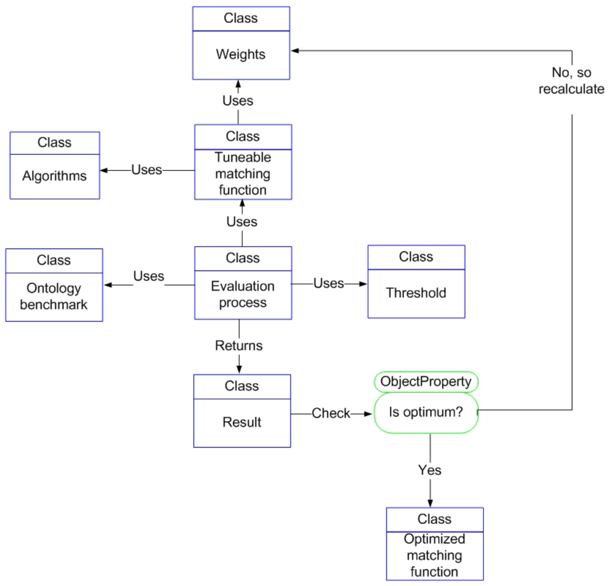


Fig. 2. General schema for our proposal

The methodology of the application of a GA requires defining the following strategies:

– Characterize the problem by encoding in a string of values the contents of a tentative solution.

– Provide a numeric fitness function that will allow to rate the relative quailty of each individual tentative solution in a population.

That is what we are going to do with GOAL. Our first task is to characterize the search space as some parameters. We need to encode several parameters in a single chromosome, so we have designed a method for converting a bit representation to a set of floating-point numbers in the real range [0, 1].

Later, we haved designed a fitness function to determine which chromosomes in the population are most likely to survive and reproduce using genetic crossover and mutation operations.

Related to the fitness function, we can choose any parameter provided for the alignment evalution process. In this way, we are providing the possibility to select one of these goals.

– Optimizing the precision ($fitness := precision$)
– Optimizing the recall ($fitness := recall$)
– Optimizing the f-measure ($fitness := f - measure$)
– Reducing the number of false positives ($fitness := fall - out$)

The fitness function consist of selecting one of the parameters retrieved by an Alignment Evaluation (see Definition 5). All of these parameters are concepts used in Information Retrieval [12] for measuring the quality of a retrieval task. Precision is the percentage of items returned that are relevant. Recall is the fraction of the items that are relevant to a query (in this case, to a matching task). F-measure is a harmonic mean from precision and recall. Finally, false positives are relationships which have been provided to the user although they are false. In some domains, (for instance in Medicine) false positives are absolutely unwanted.

Our algorithm works under the paradigm of a single goal programming strategy, but optimizing the F-Measure (a weighted sum of precision and recall) has an effect similar to a multi-objetive strategy. However, a brief discussion about using a multi-objetive algorithm will be presented as future work.

5 Empirical Evaluation

In this section, we provide an empirical evaluation of our approach. To do that, we have worked with the well-known benchmark provided by the OAEI [3]. Firstly, we have performed a preeliminary study to choose the parameters and then we have performed the main experiment.

5.1 Preeliminary Study

We are going to do a preeliminary study of the parameters for the algorithm.

– For the number of genes per chromosome we have selected such values as 5, 10 and 20. A study using a t-Test distribution has shown us that that the differences between samples are not statistically significant. Therefore, we have selected 20 genes per chromosome.

- For the number of individuals in the population, we have selected such values as 20, 50 and 100. Again, a t-Test statistical distribution has shown that the differences between these samples are not statistically significant. So we have selected a population of 100 individuals.
- Related to crossover and mutation fraction, we have choosen a high value for the crossover between genes and, a little percentage for mutations, because we wish a classical configuration for the algorithm.
- After ten independent executions, we noticed that the genetic algorithm does not improve the results beyond the fifth generation, so we have set a limit of five generations.

5.2 Main Experiment

Related to the conditions of the experiment, we have used:

- As similarity measure vector:
  {*Levhenstein*[13], *SIFO*[14], *Stolios*[15], *QGrams*[16]}
- The GA has been configured having into account the following parameters[1]:
  - 20 genes per chromosome
  - Each gene is encoded in a 10-bit representation
  - A population of 100 individuals
  - 0.98 for crossover probability
  - 0.05 for mutation probability
  - We allow 5 generations
- The platform characteristics: Intel Core 2 Duo, 2.33GHz and 4GB RAM.

The way that we have choosen for providing the dynamic evaluation of the alignment uses the following formulas:

$$Precision = \frac{\{relevant\ mappings\} \cap \{retrieved\ mappings\}}{\{relevant\ mappings\}}$$

$$Recall = \frac{\{relevant\ mappings\} \cap \{retrieved\ mappings\}}{\{retrieved\ mappings\}}$$

$$FMeasure = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

$$Fallout = \frac{\{non\ relevant\ mappings\} \cap \{retrieved\ mappings\}}{\{non\ relevant\ mappings\}}$$

Now, let us discuss the results we have obtained. Table 1 shows a brief description about the purpose of each test of the benchmark.

Table 2 shows the results from a Precision-Driven test, the Table 3 shows the results from a Recall-Driven test, the Table 4 shows results from a F-Measure-Driven test and, finally Table 5 shows the empirical data from a Fall-out-driven test.

[1] Fitness and search space have been explained in the previous section

Ontology	Brief explanation
101	Strictly identical ontologies
102	A regular ontology and a null ontology
103	A regular ontology and other with a language generalization
104	A regular ontology and other with a language restriction
201	Ontologies without entity names
202	Ontologies without entity comments
203	Ontologies without entity names and comments
204	Ontologies with different naming conventions
205	Ontologies whose labels are synonymous
206	Ontologies whose labels are in different languages
221	A regular ontology and other with no specialisation
222	A regular ontology and other with a flatenned hierarchy
223	A regular ontology and other with a expanded hierarchy
224	Identical ontologies without instances
225	Identical ontologies without restrictions
301	A real ontology about bibliography made by MIT

Table 1. Explanation of the performed tests

Ontology	Comment	Best Precision	Generations
101	Reference alignment	**1.00**	1
102	Irrelevant ontology	N/A	1
103	Language generalization	**1.00**	1
104	Language restriction	**1.00**	1
201	No names	**1.00**	1
202	No names, no comments	**1.00**	1
203	No comments (was missspelling)	**1.00**	1
204	Naming conventions	**1.00**	1
205	Synonyms	**1.00**	2
206	Translation	**1.00**	2
221	No specialisation	**1.00**	2
222	Flatenned hierarchy	**1.00**	3
223	Expanded hierarchy	**1.00**	2
224	No instance	**1.00**	1
225	No restrictions	**1.00**	2
301	Real: BibTeX/MIT	0.90	5

Table 2. Precision-Driven test

Ontology	Comment	Best Recall	Generations
101	Reference alignment	**1.00**	1
102	Irrelevant ontology	N/A	1
103	Language generalization	**1.00**	1
104	Language restriction	**1.00**	1
201	No names	**1.00**	1
202	No names, no comments	**1.00**	1
203	No comments (was missspelling)	**1.00**	1
204	Naming conventions	**1.00**	1
205	Synonyms	0.71	5
206	Translation	**1.00**	2
221	No specialisation	**1.00**	1
222	Flatenned hierarchy	**1.00**	1
223	Expanded hierarchy	**1.00**	1
224	No instance	**1.00**	1
225	No restrictions	**1.00**	1
301	Real: BibTeX/MIT	0.69	5

Table 3. Recall-Driven test

Ontology	Comment	Best F-Measure (Pr, Rec)	Generat.
101	Reference alignment	**1.00** (1.00, 1.00)	1
102	Irrelevant ontology	N/A	1
103	Language generalization	**1.00** (1.00, 1.00)	1
104	Language restriction	**1.00** (1.00, 1.00)	1
201	No names	**1.00** (1.00, 1.00)	1
202	No names, no comments	**1.00** (1.00, 1.00)	1
203	Comments was missspelling	**1.00** (1.00, 1.00)	1
204	Naming conventions	**1.00** (1.00, 1.00)	1
205	Synonyms	0.44 (0.38, 0.53)	5
206	Translation	0.43 (0.38, 0.51)	5
221	No specialisation	**1.00** (1.00, 1.00)	1
222	Flatenned hierarchy	**1.00** (1.00, 1.00)	2
223	Expanded hierarchy	**1.00** (1.00, 1.00)	2
224	No instance	**1.00** (1.00, 1.00)	3
225	No restrictions	**1.00** (1.00, 1.00)	3
301	Real: BibTeX/MIT	0.57 (0.54, 0.62)	5

Table 4. F-Measure-Driven test

Ontology	Comment	Best Fallout	Generations
101	Reference alignment	**0.00**	1
102	Irrelevant ontology	N/A	1
103	Language generalization	**0.00**	1
104	Language restriction	**0.00**	1
201	No names	**0.00**	1
202	No names, no comments	**0.00**	1
203	No comments (was missspelling)	**0.00**	1
204	Naming conventions	**0.00**	1
205	Synonyms	0.06	5
206	Translation	0.06	5
221	No specialisation	**0.00**	1
222	Flatenned hierarchy	0.00	2
223	Expanded hierarchy	**0.00**	2
224	No instance	**0.00**	2
225	No restrictions	**0.00**	3
301	Real: BibTeX/MIT	0.07	5

Table 5. Fallout-Driven test

As it can be seen, we have found the optimal alignment function for the majority of tests. In this way, we could cover matching cases, and therefore increase the chances of success. Some of test cases are solved in the first generation, this is because our application is not very difficult, maybe the problem is, but these specific instances are not.

6 Related Work

If we look at literature, we can distinguish between individual algorithms (i.e. FCA-MERGE [17] or S-Match [18]) applying only a single method of matching items i.e. linguistic or taxonomical matchers and combinations of the former ones, which intend to overcome their limitations by proposing hybrid and composite solutions. A hybrid approach (i.e.Cupid [19]) follows a black box paradigm, in which various individual matchers are melt together in a new algorithm [20], while the so-called composite matchers allow an increased user interaction (i.e. COMA++ [7], Falcon [21], CtxMatch [22], RiMOM [23]). In Fig. 3, we can see a comparison between some of the most popular tools for matching ontologies. The figure represents the arithmetic means of the values obtained for the standard benchmark for the precision and recall, obtaining the F-Measure and Fall-Out is trivial.

The problem is that those kinds of proposals use weights defined by an expert for configuring the composite matchers, while using our approach involves to compute the weigths in an automatic way, so the process can be more flexible, at least, in real scenarios.

To avoid the expert intervention, there are two research lines; one line for evaluating the results of an alignment tool and maybe feedback the process

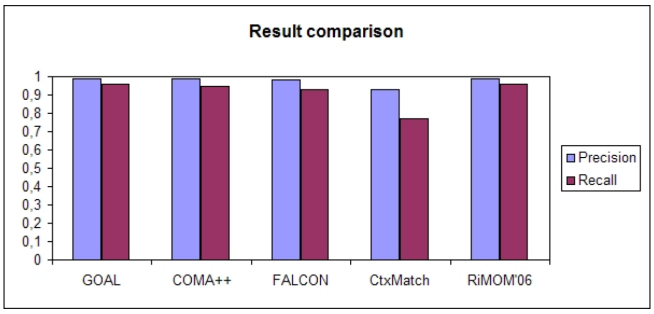


Fig. 3. Comparison between most outstanding tools

[24] [25] and another called ontology meta-matching [26] that tries to optimize automatically the parameters related to matching task. So, our approach could be considered a mechanism for meta-matching. Most outstanding examples for this paradigm are evaluated in the next sections: (i) Exhaustive Search solutions, (ii) Machine Learning solutions, and (iii) Genetic Algorithms solutions.

6.1 Exhaustive Search

Ontology meta-matching can be solved trivially by an exhaustive search when the number of similarity measures is low. The most popular approach in this sense is eTuner [27] that it is a system which, given a particular matching task, automatically tunes an ontology matching system (computing one-to-one alignments). For that purpose, it chooses the most effective basic matchers, and the best parameters to be used.

However, exhaustive searches are very expensive, and unworkable when combining a great number of measures, from a computational point of view. Unfortunately, the paper from eTuner [27] has not used an standard benchmark to offer the results, so we cannot show a comparison.

6.2 Machine Learning

Based on Machine Learning meta-matching techniques can be divided into two subtypes: Relevance feedback [28] and Neural Networks [29]:

- The idea behind relevance feedback [28] is to take the results that are initially returned from a given query and to use information about whether or not those results are relevant to perform a new query: APFEL (Alignment Process Feature Estimation and Learning) [29] is a machine learning approach that explores user validation of initial alignments for optimising automatically the configuration parameters of some of the matching strategies of the system, e.g., weights, and thresholds, for the given matching task.

– Neural Networks [30] are non-linear statistical data modeling or decision making tools. They can be used to model complex relationships between inputs and outputs or to find patterns in data. SFS [31] It is a tool for ontology meta-matching that tries to obtain automatically a vector of weights for different semantic aspects of a matching task, such as comparison of concept names, comparison of concept properties, and comparison of concept relationships. To do that, it uses neural networks.

However, these kind of solutions implies spending much time on training the systems in relation to our proposal.

6.3 Genetic Algorithms

In relation to other based-on-Genetic-Algorithm solutions, the most oustandig tool is GAOM [32] which is a genetic algorithm based approach for solving the ontology matching problem. For the purpose of a more precise representation of ontology features, it defines two aspects: intensional and extensional. On the other hand, ontology matching problem is modeled as a global optimization of a mapping between two ontologies. Then, a genetic algorithm is used to achieve a quasi optimal solution.

Table 7 shows a comparison of the results we have obtained for both GAOM and GOAL.

	Precision	**Recall**
GAOM	0.94	0.87
GOAL	**0.99**	**0.96**

Table 6. Comparison between GAOM and our proposal

Although we also follow a GA based paradigm, our GOAL is slightly better in terms of numbers to GAOM as our results shows. We think that the main diference in relation to the other tool is the fitness function. Therefore, as far as we know, our results constitute the new state of the art (S.O.T.A.) in this domain.

7 Conclusions and Future Work

We have here presented a mechanism for obtaining optimum ontology alignment functions using genetic algorithms which is part of a novel computational discipline, called meta-matching, which allows flexible and accurate automatic ontology matching and generalizes and extends previous proposals for exploiting an ensemble of ontology matchers.

We have shown that our proposal is able to find the optimal solutions for ontology alignment in most cases. According to the results, our approach seems to be an accurate and efficient tool for this task. And most importantly, it can be used in a single goal-driven way versus others using composite matching algorithms.

However, this mechanism is heavily dependent of the similarity measures to be weighted. By this reason, we highly recommend to use not only cutting-edge measures, but a big enough and representative set of them. We recommend to use, at least, one similarity measure for each kind of the matchers discussed in Section 2.

As future work, we want to study a multiobjetive strategy, thus, we plan to avoid unwanted deviations from precision and recall values. Moreover, we want to learn more about [33] for automatically selecting matching algorithms on the basis of their metadata. Our goal is, given the specifications of an ontology matching problem, to compute the optimum alignment function so that the problem can be solved accuarately and without requiring human intervention. In this way, the real interoperability in the Sematic Web might become true.

Acknowledgements

This work has been funded by Spanish Ministry of Education & Science through the research projects: *A basic infrastructure for development in the Semantic Web and its application to conceptual mediation in Bioinformatics.* (TIN2005-09098-C05-01) and *Net Centric Optimization.* (TIN2005-08818-C04-01).

References

1. Tim Berners-Lee, James Hendler and Ora Lassila. The Semantic Web. Scientific American, May 2001.
2. Christoph Kiefer, Abraham Bernstein, Markus Stocker: The Fundamentals of iSPARQL: A Virtual Triple Approach for Similarity-Based Semantic Web Tasks. ISWC/ASWC 2007: 295-309.
3. Ontology Alignment Evaluation Initiative (OAEI). `http://oaei.ontologymatching.org/2007`. Last visit: 29-jan-2008.
4. Gonzalo Navarro: A guided tour to approximate string matching. ACM Comput. Surv. 33(1): 31-88 (2001).
5. WordNet. `http://wordnet.princeton.edu`. Visit date: 11-march-2008.
6. Hong Hai Do, Erhard Rahm: COMA - A System for Flexible Combination of Schema Matching Approaches. VLDB 2002: 610-621.
7. David Aumueller, Hong Hai Do, Sabine Massmann, Erhard Rahm: Schema and ontology matching with COMA++. SIGMOD Conference 2005: 906-908.
8. Christian Drumm, Matthias Schmitt, Hong Hai Do, Erhard Rahm: Quickmig: automatic schema matching for data migration projects. CIKM 2007: 107-116.
9. Marc Ehrig, York Sure: FOAM - Framework for Ontology Alignment and Mapping - Results of the Ontology Alignment Evaluation Initiative. Integrating Ontologies 2005.
10. Robin Dhamankar, Yoonkyong Lee, AnHai Doan, Alon Y. Halevy, Pedro Domingos: iMAP: Discovering Complex Mappings between Database Schemas. SIGMOD Conference 2004: 383-394.

11. Haggai Roitman, Avigdor Gal: OntoBuilder: Fully Automatic Extraction and Consolidation of Ontologies from Web Sources Using Sequence Semantics. EDBT Workshops 2006: 573-576.
12. Ricardo A. Baeza-Yates, Berthier A. Ribeiro-Neto: Modern Information Retrieval ACM Press / Addison-Wesley 1999.
13. Vladimir Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics-Doklady*, Vol. 10, pages 707-710, August 1966.
14. Jorge Martinez-Gil, Ismael Navas-Delgado, Jose F. Aldana Montes. SIFO. An efficient taxonomical matcher for ontology alignment. Technical Report ITI-08-3. Department of Languages and Computing Sciences, University of Malaga. February 2008.
15. Giorgos Stoilos, Giorgos B. Stamou, Stefanos D. Kollias: A String Metric for Ontology Alignment. International Semantic Web Conference 2005: 624-637
16. Esko Ukkonen: Approximate String Matching with q-grams and Maximal Matches. Theor. Comput. Sci. 92(1): 191-211 (1992).
17. Gerd Stumme, Alexander Maedche: FCA-MERGE: Bottom-Up Merging of Ontologies. IJCAI 2001: 225-234.
18. Fausto Giunchiglia, Pavel Shvaiko, Mikalai Yatskevich: S-Match: an Algorithm and an Implementation of Semantic Matching. ESWS 2004: 61-75.
19. Jayant Madhavan, Philip A. Bernstein, Erhard Rahm: Generic Schema Matching with Cupid. VLDB 2001: 49-58.
20. Carmel Domshlak, Avigdor Gal, Haggai Roitman: Rank Aggregation for Automatic Schema Matching. IEEE Trans. Knowl. Data Eng. 19(4): 538-553 (2007).
21. Wei Hu, Gong Cheng, Dongdong Zheng, Xinyu Zhong, Yuzhong Qu: The Results of Falcon-AO in the OAEI 2006 Campaign. Ontology Matching 2006.
22. Slawomir Niedbala: OWL-CtxMatch in the OAEI 2006 Alignment Contest. Ontology Matching 2006.
23. Yi Li, Juan-Zi Li, Duo Zhang, Jie Tang: Result of Ontology Alignment with RiMOM at OAEI'06. Ontology Matching 2006.
24. Avigdor Gal, Ateret Anaby-Tavor, Alberto Trombetta, Danilo Montesi: A framework for modeling and evaluating automatic semantic reconciliation. VLDB J. 14(1): 50-67 (2005).
25. Patrick Lambrix, He Tan: A Tool for Evaluating Ontology Alignment Strategies. J. Data Semantics 8: 182-202 (2007)
26. Jerome Euzenat, Pavel Shvaiko. Ontology Matching. Springer-Verlag, 2007.
27. Yoonkyong Lee, Mayssam Sayyadian, AnHai Doan, Arnon Rosenthal: eTuner: tuning schema matching software using synthetic scenarios. VLDB J. 16(1): 97-122 (2007).
28. Gerard Salton, Chris Buckley: Improving retrieval performance by relevance feedback. JASIS 41(4):288-297 (1990).
29. Marc Ehrig, Steffen Staab, York Sure: Bootstrapping Ontology Alignment Methods with APFEL. International Semantic Web Conference 2005: 186-200.
30. Michael I. Jordan, Christopher M. Bishop: Neural Networks. The Computer Science and Engineering Handbook 1997: 536-556.
31. Jingshan Huang, Jiangbo Dang, Jos M. Vidal, and Michael N. Huhns. Ontology Matching Using an Artificial Neural Network to Learn Weights. IJCAI Workshop on Semantic Web for Collaborative Knowledge Acquisition 2007.
32. J. Wang, Z. Ding, C. Jiang: GAOM: Genetic Algorithm based Ontology Matching. In Proceedings of IEEE Asia-Pacific Conference on Services Computing, 2006.
33. Malgorzata Mochol, Elena Paslaru Bontas Simperl: A High-Level Architecture of a Metadata-based Ontology Matching Framework. DEXA Workshops 2006: 354-358

Anatomy of a Semantic Virus

Peyman Nasirifard

Digital Enterprise Research Institute
National University of Ireland, Galway
IDA Business Park, Lower Dangan, Galway, Ireland
`peyman.nasirifard@deri.org`

Abstract. In this position paper, I discuss a piece of malicious automated software that can be used by an individual or a group of users for submitting valid random noisy RDF-based data based on predefined schemas/ontologies to Semantic search engines. The result will undermine the utility of semantic searches. I did not implement the whole virus, but checked its feasibility. The open question is whether nature inspired reasoning can address such problems which are more related to information quality aspects.

1 Introduction and Overview

Semantic-Web-Oriented fellows encourage other communities to generate/use/share RDF statements based on predefined schemas/ontologies etc. to ease the interoperability among applications by making the knowledge machine-processable. The emergence of semantic-based applications (e.g. Semantic digital libraries[1], SIOC-enabled shared workspaces[2], Semantic URL shorten tools [3]) and also APIs (e.g. Open Calais[4]) etc. are good evidences to prove the cooperation among application developers to talk using the famous subject-predicate-object notion. However talking with the same alphabets but various dialects brings ambiguity-related problems which have been addressed by some researchers and are out of scope of this paper.

Searching, indexing, querying and reasoning over (publicly) available RDF data bring motivating use cases for Semantic search engine fellows. The crawlers of Semantic search engines crawl the Web and index RDF statements (triples) they discover on the net for further reasoning and querying. Some of them are also open to crawl the deep Web by enabling users to submit the links to their RDF data.

Since the birth of computer software, especially operating systems, clever developers and engineers benefited from software security leaks and developed

[1] `http://www.jeromedl.org/`
[2] `http://www.bscw.de/`
[3] `http://bit.ly/`
[4] `http://www.opencalais.com/`

software viruses which in some cases brought lots of disasters to governments, businesses and individuals[5].

In this paper, I describe a potential piece of software which can be used by a malicious user or a group of synergic malicious users in order to undermine the utility the Semantic search engines. In brief, what the virus does, is generating automatically random noisy knowledge which will be indexed by Semantic search engines. My main motivation of presenting this idea here is identifying some research challenges in trust layer of the well-known Semantic Web tower.

It is worthwhile mentioning that the title of this paper *Anatomy of a Semantic Virus* is perhaps misleading. Actually, I am not going to describe the anatomy of a virus that is based on the Semantic Web[6], but rather I focus on a distributed virus that targets Semantic Web data.

The structure of this position paper proceeds like the following: In the next part, I describe the problem and a scenario that demonstrates the method that the potential virus may operate upon. In section 3, I have a discussion on potential directions of finding solutions. Finally, I conclude this short position paper.

2 Problem

Semantic Web search engines (e.g. SWSE[7], Swoogle[8]) crawl and index new Semantic Web documents containing RDF statements. There are some services available on the net (e.g. Ping The Semantic Web[9] (PTSW)) that enable end users to publicly submit and announce the availability of their Semantic Web data. These submissions can be later fetched by Semantic search engines for indexing and further reasoning.

The main module of the potential virus is a piece of code that receives as input several triples and generates as output several triples based on the inputs and also predefined schemas, so that the generated RDF triples are syntactically correct, but semantically wrong (fake). Figure 1 shows a simple example. As illustrated in the figure, the input is two RDF triples: "Galway is part of Ireland" and "London is part of England". The RDF schema has already defined that Galway and London are instances of the concept City, whereas Ireland and England are Countries. In this example, the virus exchanges the object (or subject) parts of triples, taking to account the fact that both objects (or subjects) are instances of the same class (Country or City). The generated result will be "Galway is part of England" and "London is part of Ireland"; which both are correct RDF statements, but wrong (fake) knowledge. Note that the whole process is done by a malicious software and it is not kind of supervised editing and/or does not have human in the loop.

[5] `http://www.landfield.com/isn/mail-archive/2000/May/0067.html`

[6] I see this a bit strange, as common computer viruses do not communicate to each other and interoperability among viruses is not well-defined.

[7] `http://swse.org/`

[8] `http://swoogle.umbc.edu/`

[9] `http://pingthesemanticweb.com/`

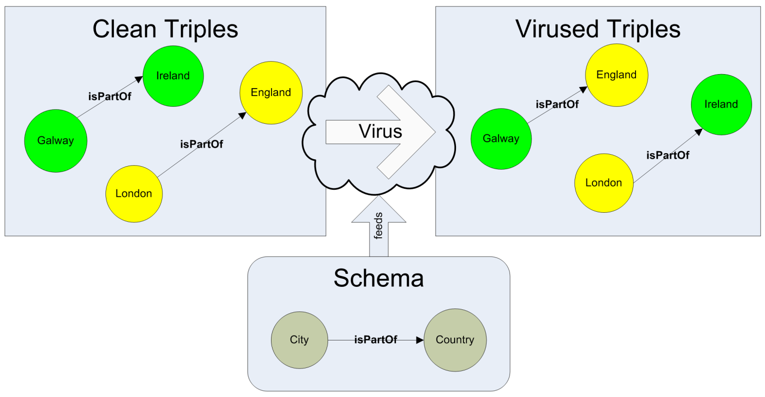


Fig. 1. Main module of the virus

The number of fake clones that can be generated is all possible instances of various concepts within RDF document.

Note that the same problem may exist on the Web and somebody may put fake knowledge on common Web pages. Moreover there exist lots of tricks to get high ranking in search engines, but as we know, the growth of the Semantic Web is not as fast as the Web[10] [1] and such malicious activities are feasible and can be *performed* using available RDF documents. Meanwhile, Semantic Web's main promise is to make the knowledge machine-processable, whereas the unstructured data on the Web is more suited for humans and obviously current machines do not have the wisdom and sense of humans.

On the other hand, someone may claim that due to the success of collaborative information gathering platforms like Wikipedia [11], the motivation for producing wrong knowledge in RDF is weak. However, we all benefit from platforms like Wikipedia, but we rarely use its articles to cite in scientific papers. The reason is perhaps the fact that the authors of such articles are unknown and we can not really trust on the content. The same applies to the RDF data. If we gather a large amount of Semantic Web data in RDF, can we really trust them? How to exclude potential fake triples from the knowledge base?

2.1 Scenario

Here I present a simple scenario to describe the possible attack that a virus can affect RDF data. As we know, publicly available services like PTSW, provide processable feeds that include the recently-added/updated RDF documents. These feeds are used by malicious software. However, the virus may even use Semantic search engines to find RDF data from the net.

[10] `http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html` and `http://sw.deri.org/2007/06/ontologymap/`

[11] `http://www.wikipedia.org/`

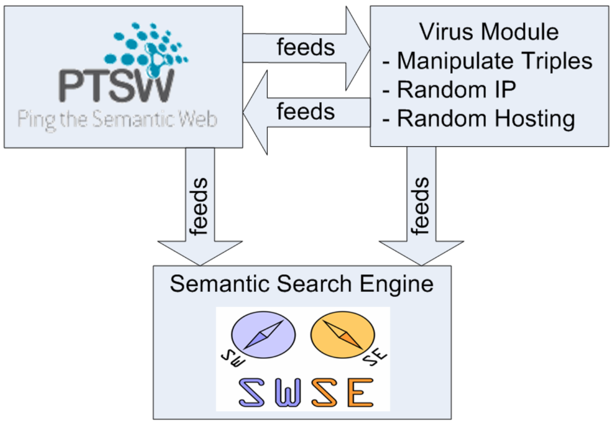


Fig. 2. Possible attack

In our scenario, the malicious software will parse the output feed of the PTSW and get an index of the published RDF files. Then it fetches the RDF statements from the net and changes them so that the generated RDF will be still valid. The result will be then submitted as an updated (or new) RDF after a random time interval with a random IP address (TCP/IP level) using a random hosting to PTSW which will be indexed by Semantic search engines. The malicious software may even submit the content directly to semantic search engines, if they provide such functionality. Figure 2 demonstrates the overall view of the possible attack which can be performed using PTSW service.

The main problem arises, when a group of people or even an individual in large scale employs several instances of the malicious software and generates fake RDF triples which will be submitted/indexed to/by the Semantic search crawlers. If search engines are not capable to cope with this situation, the result will undermine the utility of semantic searches.

3 Discussion

Digital signature is a vertical layer in the Semantic Web tower. There exist some third-parties that issue certificates for authorized users. However we may use digital signatures, certificates or any other means to cope with authentication and authorization aspects of RDF data, but we can not cope with the Quality aspects of the information (accuracy, validity, etc.). Moreover, we can not really bound the usage of Semantic Web to only authenticated, authorized and/or certified parties. Otherwise, we are highly eliminating its usage.

It is important to consider that the source of a piece of data is an important factor in validity and accuracy which are two important concepts of information quality. However, the virus is not able to change the origin of RDF document, but it is able to edit the RDF with fake statements. As *virused* RDF is still valid based on a schema, it can not be simply tracked for possible manipulation. One research problem that arises with this issue is investigation on the cloned graphs to find out the original one and perhaps log the cloned versions as illegal graphs. Probably one naive approach is using a trusted knowledge body (universal common sense facts) that verify the material generated by others. But maybe this also brings some limitations and we do not have a really comprehensive knowledge base for the whole universe facts. On the other hand, nature inspired reasoning tries to benefit from other domains to address mainly the complex reasoning challenges within Semantic Web. The open question is whether nature inspired reasoning can be useful in this area to validate the quality aspects of data.

To my view, this problem and its potential solutions can bring also some commercial interests. As an example, building a trusted knowledge party that can validate RDF-based knowledge generated by people or giving authorities to people to evaluate (semi-automatically) the generated RDF-based knowledge by others.

4 Conclusion

In this position paper, I presented briefly a method that can be used by a piece of automated software to maliciously target Semantic Web data, in order to put lots of noisy elements into the knowledge base. I mentioned that the search results of Semantic search engines may not be really trustable, as they may contain fake noisy knowledge and machines can not really benefit from them, unless we are certain that the existing knowledge in their repositories is true reliable one.

The fact that I presented this idea here is exploring some research problems that I am not aware of their solutions, after reviewing literature and having some discussions with senior Semantic Web researchers. Generating meaningful clones of a given graph based on a schema (virus) and identifying the original one from a bunch of cloned graphs (anti-virus) are possible research directions that can be further explored. I personally did not implement the whole virus, but I checked its feasibility using PTSW and a set of fake triples.

Acknowledgments. I thank Vassilios Peristeras and Stefan Decker for their supports. This work is supported by Ecospace (Integrated Project on eProfessional Collaboration Space) project: FP6-IST-5-35208

References

1. Ding, L., Finin, T.: Characterizing the Semantic Web on the Web. Proceedings of the 5th International Semantic Web Conference, 2006.

Human Similarity theories for the semantic web

Jose Quesada

Max Planck Institute, Human development
quesada@gmail.com

Abstract. The human mind has been designed to evaluate similarity fast and efficiently. When building/using a data format to make the web content more machine-friendly, can we learn something useful from how the mind represents data? We present four theories psychological theories that tried to solve the problem and how they relate to semantic web practices. Metric models (such as the vector space model and LSA) were the first-comers and still have important advantages. Advances in Bayesian methods pushed Feature models(e.g., Topic model). Structural mapping models propose that for similarity, shared structure matters more, although the formalisms that express these ideas are still developing. Transformational distance models (e.g., syntagmatic-paradigmatic -SP- model) reduce similarity to information transmission. Topic and SP models do not require preexisting classes but still have a long way to go; the need of automatically generating structure is less pressing when one of the driving forces of the semantic web is the creation of ontologies.

Keywords: similarity, cognition, semantics, information extraction, representation, psychology, cognitive science.

1. Introduction

The human mind has been "designed" to evaluate similarity fast and efficiently. When building/using a data format to make web content more machine-friendly, can we learn something useful from how the mind represents data? Are there any domain-independent findings on human representation that can inform ontology building and other semantic web activities? Can knowing humans be useful to design better for machines? I would say it might, considering that the end user of what machines using the semantic web produce is human, after all. Nature may have produced algorithms and representations that are reusable. And humans and machines dealing with lots of information may face similar problems.

There are different areas in which psychology may inform semantic web practitioners; For example, agents in the semantic web will do both inductive and deductive reasoning [1], follow causal chains [2], solve problems and make decisions [3]. All these activities depend crucially on how we represent information, and this is what similarity theories aim to explain. So in this paper we will review the major approaches to similarity in psychology and how they relate to the semantic web.

In the last 50 years, psychology has made good progress on the topic of similarity; the basic conclusion is that similarity is a hard topic, but approachable. But why is it so difficult? For a start, it is a very labile phenomenon. Murphy and Medin [4] noted that "the relative weighting of a feature (as well as the relative importance of common and distinctive features) varies with the stimulus context and task, so that there is no unique answer to the question of how similar is one object to another" (p. 296). Goodman [5] also criticized the central role of similarity as an explanatory concept. What does it mean to say that two objects *a* and *b* are similar? One intuitive answer is to say that they have many properties in common. But this intuition does not take us very far, because all objects have infinite sets of properties in common. For example, a plum and a lawnmower both share the properties of weighing less than 100 pounds (and less than 101 pounds, etc). That would imply that all objects are similar to all others (and vice versa, if we consider that they are different in an infinite set of features too). Goodman proposed that similarity is thus a meaningful concept when defined with a certain "*respect*". Instead of considering similarity as a binary relation *s*(a, b), we should think of it as a ternary relation *s*(a, b, r). But once we introduce "*respects*", then similarity itself has no explanatory value: the respects have. Thus, if similarity is useless when not defined "with respect to", then it is not an explanatory concept on which theories can be built: theories should be about "the respects" and similarity can leave the scenario without being missed.

Although this criticism could have been lethal for any psychological theories of similarity, it has not been. The abstract concept of similarity used by philosophers like Goodman and the psychological concept of similarity are different, the latter being more constrained: (1) There are psychological restrictions on what a respect can be. Although they can be very flexible and changeable with goals, purpose, and context, there are constraints in what form they take: they do not change arbitrarily, but systematically. These systematic variations prevent the set of common respects from being infinite, and enable their scientific study [6]. (2) Since people do not normally compare objects one "respect" at a time, but along multiple dimensions (e.g., size, color, function, etc.), the psychologically central issue is to explain the mechanism by which all these factors are combined into a single judgment of similarity. Then, respects do some, but not all of the work in explaining similarity judgments [7] (3) Goodman assumes that the set of features in which two objects can be compared is infinite (then, they have an infinite number of properties in which they are similar *and* dissimilar). However, in psychology we are interested in the similarity between two mental representations of the objects in the mind. Mental representations must be finite. Then computation of similarity can be thought to take place without the need of constraining respects. Theories of mental representation based on similarity should explain what is represented and how this is selected. The features represented cannot be arbitrary, otherwise they cannot be studied scientifically [8].

As a conclusion, what most similarity and categorization psychological theories have in common is the problem of choosing respects [8]: The feature selection and weighting process is outside of the scope of the models, that is, is set up a-priori by the researcher, not dictated by the theory. This is a very important flaw in a model of similarity, as Goodman pointed out. Semantic web practitioners face this problem too.

The semantic web 'standard' data structure language is RDF. In RDF, the fundamental concepts are resources, properties and statements. Resources are objects, like books, people or events. Resources have properties like chapters, proper names, or physical locations. Properties are a special type or resources that describe the relation between two resources. And a statement just asserts the properties of resources. In a sense, psychologists and semantic web practitioners are playing the same game: trying to model the world with a formalism. Psychologists want this formalism to be as close as possible to humans; Semantic web practitioners want it to 'just work'. For psychologists, a better formalism is one that models even human flaws and inconsistencies. For Semantic web practitioners, a better formalism is more expressive, while being as simple as possible; if a machine using it reaches conclusions that a human won't, so much more impressive.

The concept of similarity is very different in psychology and in machine learning too. Machine learning (and in particular, computational linguistics) use structured representations, while most of the psychologists use mainly 'flat' representations. But the main difference is that the machine leaning group often uses representations that are not psychologically plausible. For example, some parsers use human-coded representations of syntactic dependencies from corpora like TREEBANK [9], WordNet [10] or even Google queries. Semantic similarity according to Resnik [11] refers to similarity between two concepts in a taxonomy such as WordNet [10] or CYC upper ontology . These are of course not available to the mind; even though models may perform very well on interesting tasks, they have no psychological plausibility. Still, there seems to be some level of convergence between machine-learning and psychological approaches. This paper will try to make connections particularly where they are relevant for the semantic web paradigm.

2. What is Similarity, anyway?

The question "What is similarity" has inspired considerable research in the past, because it affects several cognitive processes like memory retrieval, categorization, inference, analogy, and generalization, to mention a few. We have divided current efforts to answer this question into four main branches: continuous features (spatial) models, set theoretic models, hierarchical models, and transformational distance. Similar classification can be found in Goldstone [12] and in Markman [13].

3. Continuous features (spatial) models

Shepard can be considered the father of metric models (models that use a multidimensional metric space to represent knowledge) in psychology. Shepard's [14] *Science* paper, '*Toward a universal law of generalization for psychological science*' is his most ambitious and definitive attempt to propose multidimensional spaces as an universal law in psychology. Shepard's [14] main proposal is that psychologists can utilize metric spaces to model internal representations for almost any stimulus (i.e., shapes, hues, vowel phonemes, Morse-code signals, musical intervals, concepts, etc.).

We rarely encounter the exact same situation twice. There is always some change in the environment. Usually, this new environment has some physical resemblance to an environment with which we have some history. This incremental change is the crucial element--the more similar the new environment is to something we already know, the more we will respond in a similar way.

A metric space is defined by a metric distance function D, that assigns to every pair of points a nonnegative number, called their distance, following three axioms: minimality [D(A,B) $\geq$ (A,A) = 0], symmetry [D(A,B) = D(B,A)], and the triangle inequality [D(A,B) + D(B,C) $\geq$ D(A,C)]. The methodological tool Shepard proposed is multidimensional Scaling [MDS, 15], a now-classic approach to representing proximity data. In MDS, objects are represented as points in a multidimensional space, and proximity is assumed to be a function of the distance in the space, p(*i,j*) = g [D(*i,j*)], where *g* is a decreasing function (a negative exponential). The distance in the *n*-dimensional metric space that the MDS generates represents similarity, and is calculated using the Minkowski power metric formula:

$$\mathrm{D(i,j)} = \left(\sum_{k=1}^{n} | X_{ik} - X_{jk} |^{r} \right)^{(1/r)} \tag{1}$$

Where *n* is the number of dimensions, X_{ik} is the value of the dimension *k* for entity *i*, and *r* is a parameter that defines the spatial metric to be used.

The vector space model from classical information retrieval capitalizes on this finding. It maps words to a space with as many dimensions as contexts exist in a corpus. However, the basic vector space model fails when the texts to be compared share few words, for instance, when the texts use synonyms to convey similar messages. Latent Semantic Analysis (LSA) [16, 17] solves this problem by running a singular value decomposition (SVD) and then dimension reduction on the term by document matrix. LSA can model human similarity judgments for words and text, but it faces problems. Some of these problems are conceptual: negation just doesn't work on any spatial models (NOT is a ubiquitous word and it forms a vector that adds nothing to the overall meaning). LSA uses a bag of words approach where word order does not matter; the semantic web approach requires machine learning algorithms that can produce structured representations from plain text. There are also problems with the implementation (scalability): the SVD is a one-off operation that assumes a static corpus. Updating the space with new additions to the corpus is possible, but not trivial.

LSA spawned a plethora of models for extracting semantics from text corpora. Some of them partially address structured representations. For example the Topic model [18] could potentially use a generative model with several layers of topics (hierarchical models). Beagle [19] proposes methods to capture both syntax and semantics simultaneously in a single representation using convolution. Beagle uses a moving window, so only close sequential dependencies make an impact in its understanding of syntax; it is still far from delivering a fully automatic propositional analysis of text.

Another approach is to use a large corpus of labeled articles as dimensions. For example, any text can be a weighted vector of similarities to Wikipedia articles [20]. This currently produces the highest correlation to human judgments of similarity (.72 vs .60 for LSA).

Although recent developments have addressed some implementation issues (e.g., the SVD can now be run in parallel) the direct application of LSA or any other statistical methods to semantic web problems is still not obvious. RDF operations are logical; in LSA vectors are obtained using statistical inference. Combining the logic and statistical approaches seems to be a worthwhile goal and some groups are pursuing it [21, 22].

4. Discrete set theoretic models

Tversky's set-theoretic approach and Shepard's metric space approach are often considered the two classic – and classically opposed – theories of similarity and generalization (although Shepard has some research on the set-theoretic approach`, e.g., [15, 23]).

Metric spaces have problems as a model for how humans represent similarities. Amos Tversky [24] pointed out that violations of the three assumptions of metric models (minimality, symmetry, and the triangle inequality) are empirically observed.

Minimality is violated because not all identical objects seem equally similar; complex objects that are identical (e.g., twins) can be more similar to each other than simpler identical objects (e.g., two squares).

Tversky [24] argued that similarity is an asymmetric relation. This is an important criticism for models that assume that similarity can be represented in a metric space, since metric distance in an Euclidean space is, of course, symmetric. He provided empirical evidence, for example, when participants were asked a direct rating, the judged similarity of North Korea to China exceeded the judged similarity China to North Korea[1]. A second criticism relates to the fact that similarity judgments are subjected to task and context-dependent influences, and this is not reflected in pure metric models.

[1] However, results from Aguilar and Medin 25. Aguilar, C.M., Medin, D.L.: Asymmetries of comparison. Psychon. Bull. Rev. **6** (1999) 328-337 suggest that similarity rating asymmetries are only observed under quite circumscribed conditions.

Another important criticism focuses on the triangle inequality axiom, which says that distances in a metric space between any two points must be smaller than the distances between each of the two points and any third point. In terms of similarities, this means that if an object is similar to each of the two other objects, the two objects must be at least fairly similar to each other [26]. However, James [27] gives an example in which this does not hold true: the moon is similar to a gas jet (with respect to luminosity) and also similar to a football (with respect to roundness) , but a gas jet and a football are not at all similar.

Tversky proposed that similarity is a function of both common and distinctive features, as described in the formula:

$$S(A,B) = f(\Theta(A \cap B) - \alpha(A - B) - \beta(B - A)) \qquad (2)$$

Where A and B are feature sets. The similarity of A to B is expressed as a linear combination of the measure of the common $(A \cap B)$ and distinctive $(A - B, B - A)$ features. The parameters Θ, α, and β are weighing parameters given to the common and distinctive components, and the function *f* is often simply assumed to be additive (i.e., all features are independent and their effects combine linearly).

To respond to these criticisms, some researchers have proposed different solutions that basically extend the assumptions of metric models and enable them to explain the violation in the three assumptions. Nosofsky [28] defended the metric space approach arguing that asymmetries in judgments are not necessarily due to asymmetries in the underlying similarity relationships. For example, in word similarity judgments, if the relationship A → B is stronger than B → A, a simple explanation could be that word B has higher word frequency, is more salient, or its representation is more available than word A.

Krumhansl [26] has proposed that some objections to geometric models may be overcome by supplementing the metric distance with a measure of the density of the area where the objects that figure in the comparison are placed. Krumhansl argued that if A→ B is stronger than B → A, an explanation is that A is placed in a sparser region of the space. For example, in LSA the nearest 20 neighbors of "China" range between .98 and .80. However, the 20 nearest neighbors of "Korea" range between .98 and .66, which means "China" is in a denser part of the space than "Korea". One could argue that although Krumhansl’s explanation does propose a solution for the problem, the resulting modified distance function need not satisfy the metric axioms anymore.

Kintsch [29] offered yet another way of modeling asymmetric judgments using a metric model. In his predication model, Kintsch substitutes the productivity rule in LSA (addition) with more sophisticated mechanisms that related the neighborhood of the predicate and argument to create a composed vector. His model is another source of evidence of theories that, using metric underlying models, can explain phenomena that conflict with the metric assumptions. As well, there seems to be controversy about how much the stimulus density can affect psychological similarity [30-32].

In summary, it seems that *supplemented* metric models can explain most of the criticisms attributed to them, and that some of the traditional effects such as context effects and asymmetry of similarities can be due to additional factors not considered in the classical explanations.

There used to be no feature models able to work with plain text corpora and generate, but recently the Bayesian camp has proposed a few. The most successful of these is the Topic model. Griffiths, Steyvers, and Tenenbaum [18] propose that representation might be a language of discrete features and generative Bayesian models instead of continuous spaces. This bottom-up approach has the advantage of generating 'topics' instead of unlabelled dimensions, so the resulting representations are 'explainable'. The Topic model can also explain asymmetries in similarities, because conditional probabilities are indeed asymmetrical (P(A|B) != P(B|A) necessarily).

The Topic model is indeed a feature model because 'the association between two words is increased by each topic that assigns high probability to both and is decreased by topics that assign high probability to one but not the other, in the same way that Tverksy claimed common and distinctive features should affect similarity' [18 p. 223].

At the implementation level, the Topic model is not memory-intensive; since it is a Markov chain Montecarlo model, it simply allocates words to topics in an iterative way.

The combination of explainable dimensions and possibility to handle structured representations makes the Topic model an interesting choice for the representation problems the semantic web encounters. Still, the level of structural complexity that current topic models can derive from text is very basic. Future implementations may be able to accommodate more realistic structures because the overall probabilistic framework is more flexible than previous vector space models. For promising new ways of combining ontologies with bottom-up topics, see [33, 34].

5. Hierarchical models and alignment-based models

Some researchers [e.g., 7, 12, 35] argued that neither spatial models nor discrete set theoretic models are well suited to model human representation. In several experiments humans show evidence of using structured representations rather than a collection of coordinates or features.

The structural matching theory assumes that mental representations consist of hierarchical systems that encode objects, attributes of objects, relations between objects, and relations between relations [13]. Structure mapping models are then the closest to the data structures that the semantic web uses (RDF).

The two sets of objects (A) and (B) in Figure 1 would be represented by the hierarchical structures (a) and (b). What are represented as a hierarchical system are the features of one objects, and the comparison between two mental representations consists on aligning the two structures so the matching is maximal. The best structural matching possible determines the similarity between the two objects. In Figure 1, page 8, the best interpretation involves matching the "above" relations, since they are a higher-level connected relational structure than, e.g., "circle".

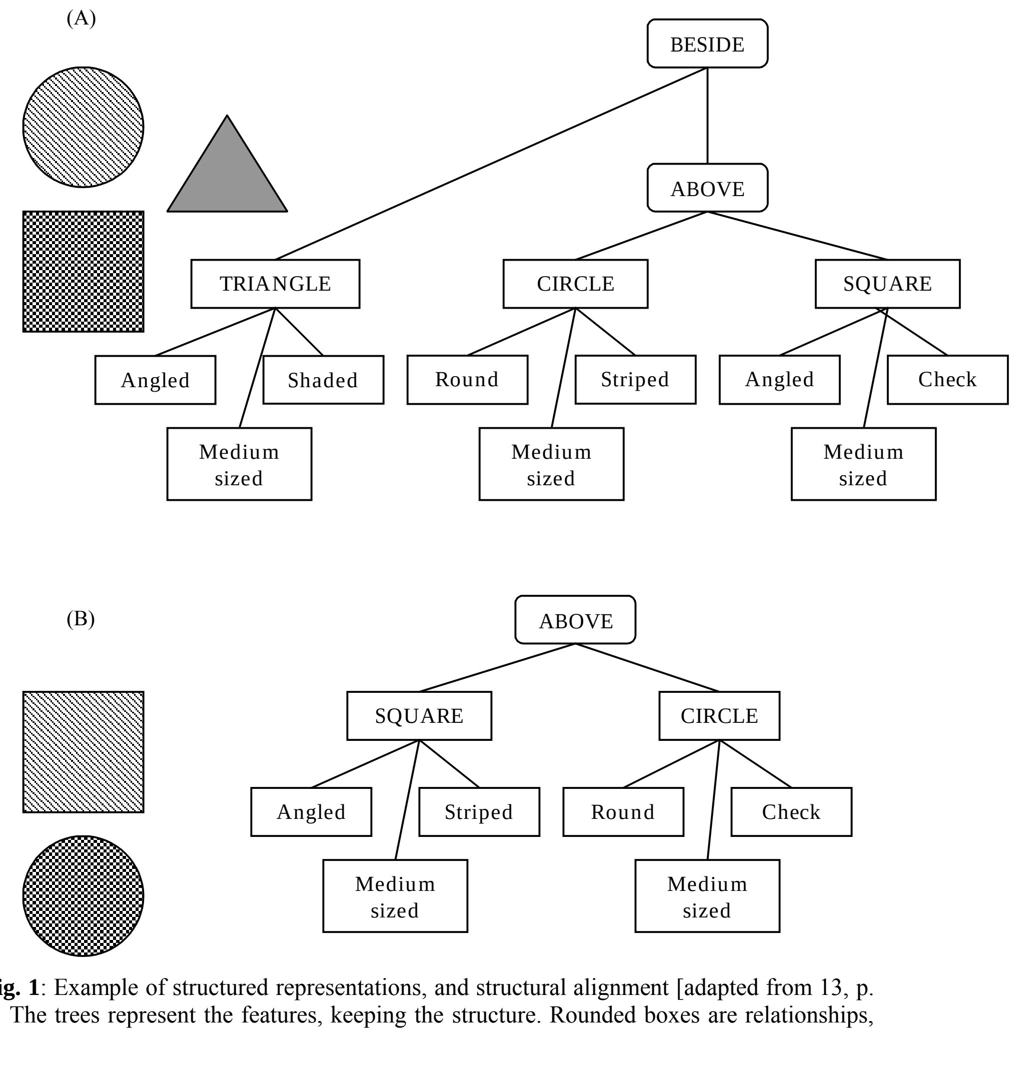


Fig. 1: Example of structured representations, and structural alignment [adapted from 13, p. 122]. The trees represent the features, keeping the structure. Rounded boxes are relationships,

uppercase square boxes are objects, and lowercase boxes are features. The "above" relation is directional; "Above" (square, circle) is different than "above" (circle, square).

The details on how the matching is done vary with the different models; The structure mapping engine SME [36] was the original; it works by forcing one-to-one mappings. That is, it limits any element in one representation to corresponding to at most one element in the other representation. SIAM [37] is an spreading activation model; it consists of a network of nodes that represent all possible feature-to-feature, object-to-object, and role-to-role correspondences between compared stimuli. The activation of a particular node indicates the strength of the correspondence it represents. SIAM treats one-to-one mapping as a soft constraint.

Structured representations gain some of their power form the ability to create increasingly complex representations of a situation by embedding relations in other relations and creating higher-order relational structures. These higher-order structures can encode important psychological elements like causal relations and implications [13]. In fact, RDF as a data structure has this property (reification, also called compositionality [38]). Currently compositionality is hard to implement for metric models and feature models.

So how are current structure-matching models in psychology different from the similarity models used in semantic web applications? The psychological models use very simple and artificial materials, like those in Figure 1. Most published papers contain a few examples where the model works (i.e., the solar system mapped to Rutherford's model of the atom) but not about where it fails. There is no published study on how general a model is (i.e., using a large selection of objects) nor what the boundary conditions are. More thorough testing and model comparison is needed. The overall impression is that fine-tuning the model to the examples in the paper took a good amount of time for the experimenter, so doing this for a large representative sample of structures may be time consuming. Second, psychological similarity models stress the importance of working memory capacity limitations, which have no relevance for machine learning and general usage in applications. Working memory limitations may help the model explain human patterns such as common errors, but do not contribute to better applications. Third, scaling may be an issue. The Rutherford example requires 42 and 33 nodes to represent the solar system and atom, respectively, and it is one of the largest mappings published. Semantic web applications can easily deal with knowledge bases several orders of magnitude larger (Although see [39, 40] for some examples of SME applications with larger knowledge bases). Last, all these theories use hand-built representations. Information extraction is a type of information retrieval whose goal is to automatically extract structured information, i.e. categorized and contextually and semantically well-defined data from a certain domain, from unstructured machine-readable documents. To date, no psychological theories of the structured kind do information extraction or propose an alternative solution to avoid hand-built representations.

So, is there no way to derive structured representation automatically from text to avoid all the above problems? The next section includes the latest, and most promising line of work: transformational distance.

6. Models based on Transformational distance

For transformational distance theories similarity of two entities is inversely proportional to the number of operations required to transform an entity so as to be identical to another [e.g., 41, 42-45]. The idea of similarity as transformation is promising in that it is very general and seems able to solve some of the previous theories problems.

We will review the representational distortion theory [8, 46], and the SP model [45, 47]. The representational distortion theory of Hahn and Chater [8, 46] uses a measure of transformation called Kolmogorov complexity, K(x|y) of one object, x, given another object, y. This is the length of the shortest program which produces x as output using y as input. The main assertion of the theory is that representations that can be generated by a short program are simple, and the ones that require longer programs are more complex. For example, a representation consisting in a million zeroes, although long, is very simple, whereas the sentence “Mary loves roses” is shorter but more complex. With this Kolmogorov measure of complexity, a similarity measure can be defined as the length of the shortest program that takes representation x and produces y. That is, the degree to which two representations are similar is determined by how many instructions must be followed to transform one into another. This approach to similarity implements the minimality and triangle assumptions (like metric theories), but enables the relationships between items to be asymmetrical, escaping one of the most pervasive criticisms of metric theories, namely the asymmetry in human similarity judgments. Note that the representational distortion theory needs to propose a vocabulary of basic representational units and basic possible transformations; but this vocabulary is currently not specified. However feature theories do not explain where features come from, so the transformational view is not at a disadvantage.

Another approach to measure transformational distance is string edit theory. The string edit theory centers on the idea that a string (composed by words, actions, states, amino acids, or any other element) can be transformed into a second string using a series of "edit" operations. String edit theory uses basic transformations like (insert, delete, match, and substitute), although this basic set varies in different implementations. Each "edit" operation for each particular item has a probability of occurrence associated. For example, in a perceptual word recognition task, the probability of substituting M for N could be higher than the probability of substituting M for B. These probabilities are defined a-priori and reflect the “cost” of the operation, but can also be learned for each problem. There is always more than one sequence of operations that can transform a string into a second string. Each sequence of operations has a probability too, which is the average of the probabilities of the transformations that form part of it.

The most well-developed model of cognition based on string edit is the syntagmatic paradigmatic (SP) model [45]. SP proposes that people use large amounts of verbal knowledge in the form of constraints derived from the occurrences of words in different slots. The constraints are categorized in two types: (1) syntagmatic associations that are thought to exist between words that often occur together, as in "run" and "fast" and (2) paradigmatic associations that exist between words that may not appear together but can appear in the same sentence context, such as "run" and "walk". The SP model proposed that verbal cognition is the retrieval of sets of syntagmatic and paradigmatic constraints from sequential and relational long-term memory and the resolution of these constraints in working memory. When trying to interpret a new sentence, people retrieve similar sentences from memory and align these with the new sentence. The set of alignments is an interpretation of the sentence. For instance, to build an interpretation of the sentence “Mary is loved by John” they might retrieve from memory “Ellen is adored by George”, “Sue who wears army fatigues is loved by Michael”, and “Pat was cherished by Big Joe”, leading to the following interpretation:

Mary					is	loved	by	John
Ellen					is	adored	by	George
Sue	who	wears	army	fatigues	is	loved	by	Michael
Pat					was	cherished	by	Big Joe

The set of words that aligns with each word from the target sentence represents the role that the word plays in the sentence. So, in the example [Ellen, Sue, Pat] represents the lovee role and [George, Michael, Joe] the lover role. The model assumes that any two sentences convey similar factual content to the extent that they contain similar words aligned with similar sets of words. Note that SP does not assume any previous knowledge (i.e., syntax). The model can solve basic question-answering tasks such as which tennis player won a match when trained on a specific plain text corpus of such news [47].

Both XML and RDF are data languages of labeled trees, and of course tree edit distance is a subclass of string edit theory [48]. There are several algorithms proposed to match such structures efficiently. For example Bertino et al [49] propose a way to match an XML tree to a set of trees (DTDs) in polynomial time. Thus, once the starting knowledge base is in a structured form, there are algorithms to do similarity operations either efficiently or in a cognitively plausible way, but not both. The remaining step is to get from a flat form to a structure that satisfies the requirements of the algorithms, which has proven not to be easy. This step is not necessary for models such as SP, since they work from plain text. In this sense this is a promising venue. Contrary to the semantic web idea to create domain-specific data languages by agreement and force that structure onto existing text in the wild, SP proposes no structure a priori. In fact, SP captures meaning as sentence exemplars. The difficult task of either defining or inducing semantic categories is avoided.

Both theories (string edit theory and on Kolmogorov complexity) deal with structured representations, feature representations and continuous representations if needed. Of course, feature theories can argue that each of the transformations proposed can be added as a feature without leaving the feature approach. However, adding higher order relationships as features makes evident one of the weak points of feature theories: anything can be a feature. Which transformations are allowed? What do people actually use? Is there a general transformation vocabulary that works for any domain? Such vocabulary, if it exists, should be independent of the transformations' characteristics (for example, their salience); otherwise, the description in feature terms becomes redundant, and could be eliminated without losing explanatory power. Because of this, the representational distortion theory proposes transformations as explanatorily prior. Feature models constitute a subset of the family of representational distortion theories, where similarity between objects is defined using a very limited set of transformations: feature insertion, feature deletion, or feature substitution. These are exactly the same transformation sets that the SP model proposes for sentence processing. However, the SP model escapes the former criticism because the "features" (in this case, words) are not generated ad-hoc, but learned empirically by experience with real-world text corpora. But the question of whether there is a viable universal transformation language still stands.

Transformational distance models could be more general than Tversky's contrast model. This view is shared by Hahn and Chater [8, pp. 71-72]: "indeed, the [Kolmogorov complexity] model can be viewed as a generalization of the feature and spatial models of similarity, to the extent that similar sets of features (nearby points in space) correspond to short programs". Chater and Vitanyi [50, 51] have mathematical proof that any similarity measure reduces to information distance.

7. Summary and Conclusion

We have presented why similarity is a hard problem and four major psychological theories that tried to solve it. We started the discussion presenting metric models and their flaws; which were partially addressed by feature theories. Then we presented structural alignment models, explaining how they relate to current work on structured data such as RDF. We concluded with transformational distance models as the closest to an ideal solution.

One recurring theme is that once the starting knowledge base is in a structured form, there are algorithms to do similarity operations either efficiently [49] or in a cognitively plausible way [52] (but not both). The remaining step is to get from a flat form to a structure that satisfies the requirements of the algorithms, which has proven not to be easy. Currently the SP model and the Topic model show promise as bottom-up models that start with plain text and generate structured representations. The immediate advantage when compared with traditional machine learning information extraction tools is that they do not require preexisting classes (as they are inferred). Admittedly, both SP and Topic models still have a long way to go, and up to now they have focused in extraction of syntactic categories (and in an imperfect way). The semantic web of course needs an entire universe of different categories (not only syntactic).

The semantic web practitioners however are perfectly happy manually creating domain-specific languages to describe their domains (i.e., RDF-schema). This is good news because it increases the number of similarity models one can choose from. SP and the Topic model have the head start of making no *a priori* commitment to particular grammars, heuristics, or ontologies. But this may not be a tremendous advantage in a world that seems to be eager to produce ontologies and fit all existing knowledge into those structures. Time will tell if bottom-up approaches will proliferate or fade away.

References

1. Heit, E., Rotello, C.: Are There Two Kinds of Reasoning? Proceedings of the Twenty-Seventh Annual Conference of the Cognitive Science Society (2005)
2. Glymour, C.: The Mind's Arrows: Bayes Nets and Graphical Causal Models in Psychology. MIT Press, Boston (2001)
3. Newell, A., Simon, H.A.: Human Problem Solving. Prentice-Hall, Inc., Englewood Cliffs, New Jersey (1972)
4. Murphy, G.L., Medin, D.L.: The Role of Theories in Conceptual Coherence. Psychol Rev **92** (1985) 289-316
5. Goodman, N.: Seven strictures on similarity. In: Goodman, N. (ed.): problems and projects:. Bobbs Merrill, Indianapolis (1972) 437-450
6. Medin, D.L., Goldstone, R.L., Gentner, D.: Respects for Similarity. Psychol Rev **100** (1993) 254-278
7. Goldstone, R.L.: The Role of Similarity in Categorization - Providing a Groundwork. Cognition **52** (1994) 125-157
8. Hahn, U., Chater, N.: Concepts and similarity. In: Lamberts, K., Shanks, D. (eds.): Knowledge, concepts, and categories. MIT Press, Cambridge, MA (1997)
9. Marcus, M., Marcinkiewicz, M., Santorini, B.: Building a large annotated corpus of English: the penn treebank. Computational Linguistics **19** (1993) 313-330
10. Miller, G., Beckwith, R., Fellbaum, C., Gross, D., Miller, K.: Introduction to WordNet: An On-line Lexical Database*. International Journal of Lexicography **3** (1990) 235-244
11. Resnik, P.: Using information content to evaluate semantic similarity in a taxonomy. Proceedings of the 14th International Joint Conference on Artificial Intelligence **1** (1995) 448-453
12. Goldstone, R.L.: Similarity. In: Wilson, R.A., Keil, F.C. (eds.): MIT encyclopedia of the cognitive sciences. MIT Press, Cambridge, MA (1999) 763-765
13. Markman, A.B.: Knowledge representation. Lawrence Erlbaum Associtates, Mahwah, NJ (1999)
14. Shepard, R.N.: Toward a universal law of generalization for psychological science. Science **237** (1987) 1317-1323
15. Shepard, R.N.: Multidimensional scaling, three-fitting, and clustering. Science **214** (1980) 390-398
16. Landauer, T., McNamara, D., Dennis, S., Kintsch, W.: LSA: A road to meaning. Mahwah, NJ: Lawrence Erlbaum Associates, Inc (2007)

17. Landauer, T.K., Dumais, S.T.: A solution to Plato's problem: The Latent Semantic Analysis theory of the acquisition, induction, and representation of knowledge. Psychol Rev **104** (1997) 211-240
18. Griffiths, T.L., Steyvers, M., Tenenbaum, J.: Topics in semantic representation. Psychol Rev **in press** (2007)
19. Jones, M.N., Mewhort, D.J.K.: Representing Word Meaning and Order Information in a Composite Holographic Lexicon. Psychol Rev **114** (2007) 1-37
20. Gabrilovich, E., Markovitch, S.: Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis. Proceedings of the 20th International Joint Conference on Artificial Intelligence (2007) 1606–1611
21. Bernstein, A., Kiefer, C.: Imprecise RDQL: towards generic retrieval in ontologies using similarity joins. Proceedings of the 2006 ACM symposium on Applied computing (2006) 1684-1689
22. Kiefer, C., Bernstein, A., Stocker, M.: The Fundamentals of iSPARQL: A Virtual Triple Approach for Similarity-Based Semantic Web Tasks. LECTURE NOTES IN COMPUTER SCIENCE **4825** (2007) 295
23. Shepard, R.N., Arabie, P.: Additive Clustering - Representation of Similarities as Combinations of Discrete Overlapping Properties. Psychol Rev **86** (1979) 87-123
24. Tversky, A.: Features of similarity. Psychol Rev **84** (1977) 327-352
25. Aguilar, C.M., Medin, D.L.: Asymmetries of comparison. Psychon. Bull. Rev. **6** (1999) 328-337
26. Krumhansl, C.: Concerning the applicability of geometric models to similarity data: The interrelationship between similarity and spatial density. Psychol Rev **85** (1978) 445-463
27. James, W.: principles of psychology. Holt, New York (1890)
28. Nosofsky, R.: Stimulus Bias, Asymmetric similarity, and classification. Cognitive Psychol **23** (1991) 94-140
29. Kintsch, W.: Predication. Cognitive Science **25** (2001) 173-202
30. Krumhansl, C.L.: Testing the Density Hypothesis - Comment. J Exp Psychol Gen **117** (1988) 101-104
31. Corter, J.E.: Testing the Density Hypothesis - Reply. J Exp Psychol Gen **117** (1988) 105-106
32. Corter, J.E.: Similarity, Confusability, and the Density Hypothesis. J Exp Psychol Gen **116** (1987) 238-249
33. Chemudugunta, C., Holloway, A., Smyth, P., Steyvers, M.: Modeling Documents by Combining Semantic Concepts with Unsupervised Statistical Learning. 7th International Semantic Web Conference, Karlsruhe (2008)
34. Chemudugunta, C., Smyth, P., Steyvers, M.: Combining Concept Hierarchies and Statistical Topic Models. ACM 17th conference on Information and Knowledge Management (2008)
35. Markman, A.B., Gentner, D.: Structural Alignment During Similarity Comparisons. Cognitive Psychol **25** (1993) 431-467
36. Falkenhainer, B., Forbus, K., Gentner, D.: The Structure-Mapping Engine: Algorithm and Examples. Artif. Intell. **41** (1989) 1-63
37. Goldstone, R.L.: Similarity, Interactive Activation, and Mapping. J. Exp. Psychol.-Learn. Mem. Cogn. **20** (1994) 3-27

38. Fodor, J.A., Pylyshyn, Z.W.: Connectionism and Cognitive Architecture - a Critical Analysis. Cognition **28** (1988) 3-71
39. Klenk, M., Forbus, K., IL, N.U.E.: Cognitive Modeling of Analogy Events in Physics Problem Solving From Examples. Proceedings of the29th Annual Meeting of the Cognitive Science Society meeting. NORTHWESTERN UNIV EVANSTON IL (2007)
40. Hinrichs, T., Forbus, K.: Analogical Learning in a Turn-Based Strategy Game. IJCAI - International Joint Conference on Artificial Intelligence, Hyderabad (2007)
41. Chater, N.: Cognitive science - The logic of human learning. Nature **407** (2000) 572-573
42. Chater, N.: The search for simplicity: A fundamental cognitive principle? Q. J. Exp. Psychol. Sect A-Hum. Exp. Psychol. **52** (1999) 273-302
43. Pothos, E.M., Chater, N.: A simplicity principle in unsupervised human categorization. Cognitive Science **26** (2002) 303-343
44. Pothos, E., Chater, N.: Categorization by simplicity:a minimum description length approach to unsupervised clustering. In: Hahn, U., Ramscar, M. (eds.): Similarity and categorization. Oxford University Press, Oxford (2001)
45. Dennis, S.: A memory-based theory of verbal cognition. Cognitive Science **29** (2005) 145-193
46. Hahn, U., Chater, N., Richardson, L.B.: Similarity as transformation. Cognition **87** (2003) 1-32
47. Dennis, S.: An unsupervised method for the extraction of propositional information from text. Proceedings of the National Academy of Sciences **101** (2004) 5206-5213
48. Rice, S., Bunke, H., Nartker, T.: Classes of Cost Functions for String Edit Distance. Algorithmica **18** (1997) 271-280
49. Bertino, E., Guerrini, G., Mesiti, M.: Measuring the structural similarity among XML documents and DTDs. Journal of Intelligent Information Systems (2008) 1-38
50. Chater, N., Vitanyi, P.: Simplicity: a unifying principle in cognitive science? Trends Cogn Sci **7** (2003) 19-22
51. Chater, N., Vitanyi, P.: The generalized universal law of generalization. J Math Psychol **47** (2003) 346-369
52. Larkey, L.B., Love, B.C.: CAB: Connectionist analogy builder. cognitive Science **27** (2003) 781-794

ISWC 2008

The 7th International Semantic Web Conference

October 26 – 30, 2008

Congress Center, Karlsruhe, Germany