# Generating and Comparing Models within an Ontology

*Trent Apted, Judy Kay*

School of Information Technologies
University of Sydney, Australia 2006

*{tapted, judy}@it.usyd.edu.au*

## Abstract

*An ontology is useful for providing a conceptually concise basis for developing and communicating knowledge. This paper discusses an application of an automatically constructed ontology of computer science and its use for comparison of models in the computer science domain. We present the architecture, algorithms and current results for MECUREO, a system which builds an extensive model of a computer science entity from limited information. The entity might be a document such as an email message, a course description document or a biography. It is intended to give a measure of the similarity of any two such models. We describe MECUREO's mechanism for constructing and representing models and its present performance in comparing models.*

**Keywords** information retrieval, ontologies, acquisition, sharing and reuse of conceptual structures

## 1 Introduction

A central use of ontologies is to facilitate the exchange of data. However, the rich semantic information contained in an ontology can be used for a variety of other roles in the learning domain. We will discuss the ability to automatically construct a model from limited, arbitrary textual input and then use this model with others to *learn* from the input in the context of a specific problem.

Given a personal biography, a list of publications, or a person's self-description of interests, we would like to identify people who have similar interests. There are several ways that this information might be used. For example, the documents that are interesting to one person are more likely to be interesting for a person with similar interests. If we have access to the rated documents from one person, we can then use this information to make recommendations to other, similar people.

**Proceedings of the 7th Australasian Document Computing Symposium,
Sydney, Australia, December 16, 2002.**

For instance, a computer science researcher describes their interests as *knowledge representation*, *genetic programming* and *data mining*. Another researcher describes their interests as *neural networks*, *data marts* and *STRATEGY*. We would like to be able to recognise that these two people are actually interested in quite similar things. Since they have chosen to express their interests with different terms, we can only recognise the similarity between the two self-descriptions if we can recognise the similarity of the terms used by each user. The term-level similarities can then be reconstructed to form a measure of similarity between the models of the two users themselves.

One way to tackle this problem is to make use of knowledge about the relationships between terms to grow the set of terms supplied by each user. With this knowledge we can describe the users' interests as the set of terms they explicitly supply, plus the set of terms that we infer are similar. There are two elements in MECUREO's approach to matching people's interests: growing the set of user-supplied terms and the matching process for the enlarged sets of interests.

In Section 2, we give a brief description of the way that we have constructed a computer science ontology. Then Section 3 outlines the process we use to grow ontology graphs from an initial small set of concepts. Section 4 describes the process used to match graphs and Section 5 describes its evaluation. We conclude by outlining related work in Section 6 and giving conclusions in Section 7.

## 2 An Ontology for Computer Science

The ontology used is the result of the automatic construction process discussed in [1] applied to the Free On-Line Dictionary of Computing [2]. The process results in an ontology of computer science that is backed by a weighted digraph. Relationship weights are reals distributed in the interval (0, 1] indicating the "cost" of the relationship (i.e. a smaller weight indicates a stronger relationship).

The relationships are also given a type that has conceptual meaning and associated direction. For example, *laziness* is modelled as a 'synonym' for *lazy*

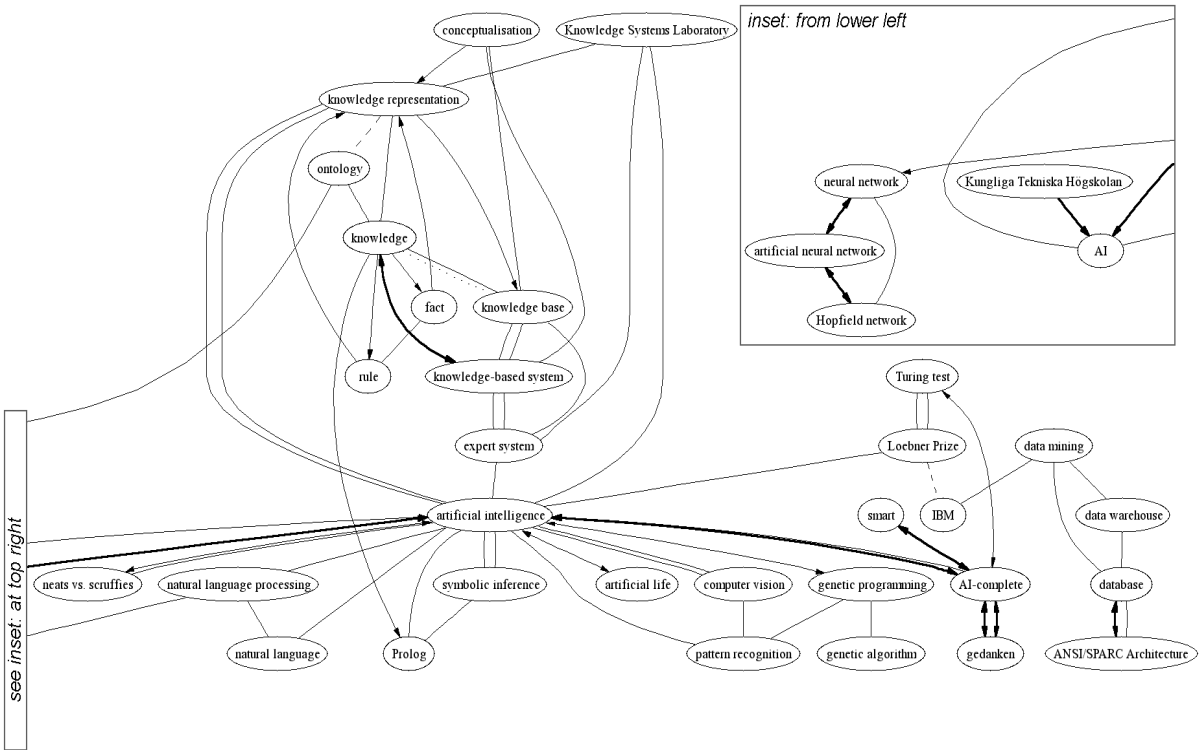


**Figure 1 – Model expanded from the concepts *knowledge representation*, *genetic programming* and *data mining* with a depth of 0.8 and minimum peerage of 3.**

*Bidirectional edges indicate synonym (or strong sibling) relationships, reversed arrowheads indicate antonym (or other opposing) relationships, directed edges indicate strict parent/child relationships and undirected edges indicate undetermined relationships (or weak siblings). Bold, normal, dashed and dotted line styles indicate progressively weaker relationships for all types.*

*The image was constructed by inputting the above query into the web form for the MECUREO demonstration cgi accessed via* http://www.ug.it.usyd.edu.au/~tapted/modelgrow.html.

*evaluation*, *declarative language* is modelled as an 'antonym' for *imperative language*, *TABLOG* is modelled as a 'child' of both *relational programming* and *functional programming*, *curried function* has *Haskell* modelled as a 'parent' and other relations have 'sibling' or 'unknown' relationship types. Synonym and antonym relationships are bi-directional. Parent and child relationships are directed from the parent to the child (whether it is called a *parent* or *child* depends on the context of the location where the relationship was discovered), and sibling and unknown relationships are undirected.

The particular ontology used contains 23,095 concepts and 57,550 directed relationships of which 55,038 are between distinct concepts. The queries used for generation and comparison traverse relationships in either direction, so there is an effective average of 4.77 distinct peers per node for the purposes of these queries. However, often a node has only a single relationship that is an explicit synonym with negligible weight such that the related nodes behave semantically as if they were a single node. More information about the ontology can be found in [1].

## 3 Model Expansion

Model generation proceeds in three stages. First, key concepts are extracted from a data source. These are then matched (where possible) to concepts in the ontology. These concepts have a corresponding subgraph, which is then *grown* within the ontology to produce a rich model. Our work has concentrated on the second and third stages of the process. We start with a simple list of terms needed from the first stage.

### 3.1 Subgraph Growing

By indicating simply a subset of nodes in the ontology, it is possible to *grow* the corresponding subgraph a specified distance. For example given a set of concepts in the ontology, it is possible to construct a subgraph consisting of the nodes corresponding to these concepts. This graph may then be grown to include nodes *nearby* any of the existing nodes. The procedure takes the form of a query on the ontology for a model, given a set of concepts and a distance (or alternatively, a maximum output size) as input.

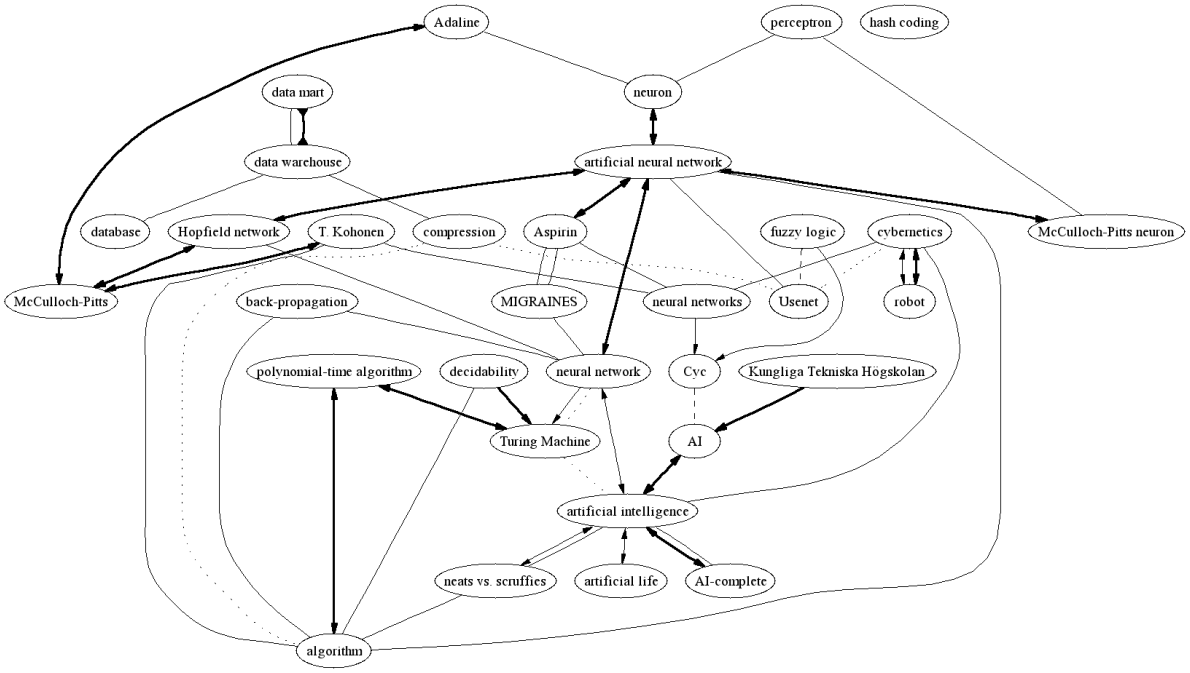


**Figure 2 – Model expanded from the concepts *STRATEGY, neural networks* and *data marts* with a depth of 0.8 and minimum peerage of 3.**

*Note that in* both *Figure 1 and Figure 2, the node whose label is 'ShowCase STRATEGY (www://www.showcasecorp.com)' has been culled because its only peer is 'data mining' and so the node does not meet the minimum peerage requirement for image output of 3, although it is still part of the model from which the image is derived. The label is such simply because the dictionary from which the ontology is derived references the relationship in this way.)*

Figure 1 and Figure 2 are graph representations of models constructed using the MECUREO system. Figure 1 is the result of expanding the terms *knowledge representation*, *genetic programming* and *data mining* a distance of 0.8 in the ontology. For a smaller and clearer image, nodes in the graph with fewer than 3 peers (and their incident edges) were removed. Figure 2 is the result of similarly expanding the terms *neural networks*, *data marts* and *STRATEGY* after going through an intermediate concept matching step (see §3.2). Both graphs are generated using knowledge from the unmodified, automatically constructed ontology.

Each node in the subgraph to be grown is given an initial weight to begin the query, which defaults zero. This can be specified along with the set of words from which the model is generated, or it can simply be the *node* weight assigned to a node in a previous query. The graph that results from the query will include all nodes visited along any path originating from any node in the input graph, whose path length is not more than the specified query distance, minus the weight of the originating node.

However, in order to give greater emphasis to output nodes that are close to more than one node in the input subgraph, the weight assigned to any node adjacent to more than one input node is reduced. The reduction is calculated by treating the edges as if they were *parallel resistors*. This adjustment propagates outwards, such that any path from this node may be slightly longer. In this way, inputs (especially word lists) that tend to focus on groups of related concepts in the ontology will have these groups emphasised, while isolated concepts will have less growth. It also makes it possible to specify relatively small distances to grow (perhaps smaller than any edge weight) and yet still yield some growth if the input graph has one or more conceptual focal points.

This type of query allows any computer science entity to be modelled, provided it can be translated into a set of key concepts. The ontology determines the relationships between those concepts and allows the model to be extended to include related concepts. As a side-effect, the nodes are each given a weight that reflects their *closeness* to the query input. This model may then be output, stored or used in further queries.

These calculations rely upon the weights associated with the links within the ontology generated from the online dictionary of computer terms. We now describe the way we determine these.

We have defined a base weight for each link type. This is a constant between 0 and 1. The actual weights are specified in a configuration set. We have chosen values which are both intuitive and which appear to produce sensible results on our training and evaluation sets. This approach seemed to provide better results than the more common practice of a common weight for all link types in such ontological structures. The

need for our differentiated link weight values may be due to the fact that our ontology was constructed automatically from an online dictionary, rather than being hand-crafted.

This type-link is then adjusted according to the position of the phrase in the definition: later terms have a higher weight, indicating that the term is less closely related to the term being defined. The formula for the weight adjustment factor, $p_i$, is calculated as:

$$p_0 = 0$$

$$p_{i+1} = p_i + \frac{\frac{1}{2} - p_i}{C}$$

Here $C$ is an adjustable positive constant, greater than one with 10 as the default. Small $C$ values penalise later links in highly branching nodes while a large $C$ distributes weights for successive links more gradually. As $p_i$ approaches 0.5, rearranging with the default of 10, this yields $p_{i+1} = 0.05 + 0.9p_i$. Otherwise, the general closed form is

$$p_i = \frac{1 - r^i}{2} \quad \text{with} \quad r = 1 - \frac{1}{C}$$

Essentially it gives a gradually increasing penalty *smoothly* tapered at 0.5.

This formula is a compromise designed to give 'fair' weights to all links, be they in definitions with many links or not. Informal evaluations indicated that this strategy was more effective. Earlier experiments with more discrete weights gave poorer results for highly branching nodes. The formula spread the distribution of weights so that links mentioned early in a definition are treated as more closely related to the word being defined. This may seem a rather bold generalisation. However, it is reflected in the nature of many definitions in the dictionary.

The final weight assigned to an edge is the sum of the base weight and the adjustment. It is possible for the weights of links whose type is explicitly *weak* to exceed 1.0, so this value is capped at 1.0. This allows a query to deliberately visit (at least) every node at a particular *depth* (as opposed to distance) without additional computation being required.

## 3.2 Concept Matching

Although our current work has assumed that we have been provided with a list of terms which describe each entity being compared, our longer term goal is to integrate the system with tools which can extract the terms from documents such as email, an on-line biography or a call for papers. The aim will be to find near matches between terms from the entity with nodes in the ontology and to calculate their relevance. This can then be used as a basis to model (and grow) the entity. To accomplish this, we have been evaluating the potential value of techniques such as stemming [3] and edit distance [4].

There are significant obstacles in developing a good matching procedure for concepts in the computer science domain. There is detail that would be lost if a simple stemmer, edit distance or even case-insensitivity is used to match terms. For example, there is *COM* (Component Object Model or Computer On Microfilm) and there is *com* (www...com) for which two, distinct and distant nodes exist. Furthermore, the vast number of acronyms and proper nouns in our ontology are not suited to a stemmer for the English language. There is also an obstacle in the form of the size of our ontology, which makes a full scan of concepts time consuming.

The procedure used to find the nodes in Figure 2 matches concepts simply on the case-insensitive stems of their component words. This takes constant time for each concept to be matched as string hashing is performed. However, if no stem match was found, a substring search is performed that takes linear time (on the size of the ontology ~$10^4$) for each concept to be matched. The stemmer used is a version of the Porters stemmer [3], slightly modified to perform on the international-English used in the FOLDOC [2] resource. For the query *'"STRATEGY" "neural networks" "data marts"'*, the nodes *ShowCase STRATEGY (www://www.showcasecorp.com)*, *neural network*, *neural networks* and *data mart* are matched. These nodes were then grown as described in §3.1 with a distance of 0.8 to produce Figure 2.

# 4 Matching Models

The purpose of a matching query is to return a value, or set of values, reflecting the similarity of the two input graphs which models two entities we want to compare. In the case of Figures 1 and 2, there is considerable overlap: for example, *artificial intelligence* and its immediate peers. Ideally, we would like to quantify this in terms such as: these models are, say, approximately 70% similar.

We have explored approaches to this task. We currently calculate the following measures:

- the node weight, as assigned by a previous query, and we do so both for nodes that are common to both graphs and those appearing in only one graph;
- the numbers of nodes present and common;
- the relative sizes of the graphs, so that, for example, we can bias a subset comparison for model classification;
- the *distance* in the ontology between a node in only one graph and any node of the graph in which it is not;
- minimum weights of spanning trees; and
- characteristics of the graph that would result from a *merging* of the input models.

We envisage that the heuristic for combining these measures may need to depend upon the application. Some applications may weight some of these elements more highly. Other uses may simply preserve them as separate measures. Techniques in related work [9, 10]

may also show promise if they can be applied to our ontology without losing detail.

## 5 Evaluation

It is difficult to evaluate an ontology and matches against them. Our first systematic step involved matching the ontology against concept maps created by senior undergraduate computer science students. A detailed account of the experimental procedure and the results can be found in [1].

To perform a comparison, we first automatically generated a concept map directly from the ontology. This was created by making a *model expansion* query using a single concept. This meant that in evaluating the ontology part of the model expansion procedure was evaluated as well. Overall, 105 of the 122 nodes drawn across all volunteers' concept maps matched exactly (~86%[1]) and only 4 of the 108 edges between matched concepts required more than two edges from the ontology to form the same path [1].

For the model matching procedure, informal evaluation has been conducted by comparing graphs derived by strategic modifications to a collection of mock models. This allowed algorithms to be analysed and for early refinement of the algorithms.

We note that accuracy is hard to define for this problem. Being able to state that two models are, say, eighty percent similar is not feasible – neither for a computer nor a human. The problem is not to construct an isomorphism, but to determine how disparate two models are within the context of the ontology. Furthermore, we wish to make use of the additional meta-data in our ontology in the form of relationship *weights* to give an accurate comparison using all the available information.

We have performed a comparison of our ontology against a trusted source, the ASIS Thesaurus of Information Science [11]. After pre-processing and parsing the thesaurus it was possible to use the same tools written for the ontology to aid our evaluation procedure.

In the thesaurus, only 270 of the 1345 concepts were exact matches. With stemming, this was increased to 519 *near* matches. We then attempted to estimate where ontology was failing to match the thesaurus. We knew that part of the difference was due to the different orientation of the Thesaurus and the on-line dictionary which is the foundation for our ontology. The former has a strong business orientation where the latter is for computer science terms. Manual analysis of some portions of the thesaurus that are directly related to computer science, rather than business, suggests that approximately 60% of nodes could be directly mapped to concepts in the ontology, for the purpose of comparing relationships.

We then explored the role of near matches of terms across the two sources. For each parent node in the thesaurus that matched a node in the ontology, a point query was performed. The cost assigned to each matched child (as a result of the query) was used to evaluate the precision. This cost is simply the shortest path, so this value was recorded for each node that matched. The analysis of these values was then conducted by hand. Ideally the nodes would be adjacent and the path cost between them would be between 0.5 and 1.0. For synonyms and other strong relationships, 90% of costs ranged between 0.2 and 1.9 units. The mean cost was approximately 1.4 units, translating to an average of around two edges on each path. The remaining 10% of costs were too large to have the nodes considered as being directly related due to branching effects.

Again, these results reflect how the focus of the thesaurus does not correspond directly to that of the dictionary. For example, parent concepts in the thesaurus such as *artificial intelligence* generally perform well due to their correspondence to a specific concept in the ontology. On the other hand concepts such as *data processing* are somewhat ambiguous in a computer science domain, and so did not perform as well.

## 6 Related Work

The automatic construction of ontologies and extraction of semantic information from machine readable dictionaries and text is broadly interesting. Some important work in this area can be found in [5, 6] where the focus is deriving knowledge from a specific domain (such as within a corporation), and in [7, 8] where the focus is to determine semantics from knowledge in general, with little emphasis on proper nouns. Our work on automatic ontology construction has focussed on generating an ontology that can be used to assist in modelling computer science entities. For this reason, we took a different approach, building a detailed ontology of computer science backed by a weighted digraph to facilitate automatic querying.

The task of matching terms with a similar meaning has been tackled by a quite diverse set of approaches. For example, [12] explores the use of formal concept analysis in the context of document retrieval rather than modelling user interests as a first stage in matching users. Another similar project, [13], also uses lattices for inferring user interests. However, it uses the lattices as a basis for interacting with the user to elicit an expanded self-description. Similarly, [14] addresses the problem of building richer models of people's interests by relying on knowledge elicitation from the user, but it uses taxonomic representation of computer terms.

[1] the unmatched nodes were often specific instances of a concept or parts of the syntax used for a particular concept

MECUREO's term growing algorithms are based on the intuitive notion that a pair of terms that are more similar should be closer in the ontological graph. This idea is not novel and was recognised in the earliest work on semantic networks reviewed in [14, 15].

## 7 Conclusion

This paper has described our approaches to constructing a detailed computer science ontology as a basis for generating and comparing models of computer science entities from limited information. We accomplish this by using combinations of ontology *queries*. We have reported our initial evaluations which suggest that the approach is promising.

## References


[1] T.Apted and J.Kay. *Automatic Construction of Learning Ontologies*. ICCE, 2002 Workshop on Ontologies for Learning, to appear.

[2] FOLDOC - the Free On-Line Dictionary Of Computing [© 1993 by Denis Howe, updated regularly], at http://www.foldoc.org/

[3] M. F. Porter. *An algorithm for suffix stripping*. Program, 1980.

[4] I. V. Levenshtein. *Binary Codes capable of correcting deletions, insertions, and reversals*. Cybernetics and Control Theory, 1966.

[5] D. Moldovan, R. Girju, V. Rus. *Domain-Specific Knowledge Acquisition from Text*. ANLP, 2000.

[6] J. Kietz, R. Volz. *Extracting a Domain-Specific Ontology from a Corporate Intranet*. Proceedings of CoNLL-2000 and LLL-2000, Lisbon, Portugal, 2000.

[7] S. D. Richardson, W. B. Dolan, L. Vanderwende. *MindNet: acquiring and structuring semantic information from text*. ACL, 1998.

[8] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, K. Miller. *Introduction to WordNet: An On-line Lexical Database*. 1990 (revised 1993).

[9] A. Maedche, S. Staab. *Measuring Similarity between Ontologies*. EKAW, 2002.

[10] D. B. Leake, A. Maguitman, A. Cañas. *Assessing Conceptual Similarity to Support Concept Mapping*. Proceedings of the Fifteenth International Florida Artificial Intelligence Research Society Conference, 2001.

[11] ASIS [American Society for Information Science] Thesaurus of Information Science; (*visited 2002-08-08, published 2002-02-07*), at http://www.asis.org/Publications/Thesaurus/isframe.htm

[12] P. Becker, P.Eklund. *Using formal concept analysis for document retrieval*. ADCS, 2001.

[13] H. Suryanto, P. Compton. *Discovery of Ontologies from Knowledge Bases*. First International Conference on Knowledge Capture, 2001.

[14] R. F. E. Sutcliffe, D. O'Sullivan, A. McElligot, L. Sheahan. *Creation of a Semantic Lexicon by Traversal of a Machine Tractable Concept Taxonomy*. Journal of Quantitative Linguistics, 1995.

[15] P. Resnik. *Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language*. Journal of Artificial Intelligence Research, 1999.