

# MECUREO Ontology and Modelling Tools

Trent Apted, Judy Kay  
School of Information Technologies,  
University of Sydney, Australia, 2006.  
{taped, judy}@it.usyd.edu.au

## Abstract

This paper describes MECUREO, a system that automatically constructs an ontology from a dictionary. We describe its use in the context of a dictionary of computer science. MECUREO parses this to construct a graph representing a lightweight ontology of computer science. This paper describes MECUREO's approach to parsing the dictionary and the construction of the ontology graph. We also describe MECUREO's point query facility. This generates a small ontology for use in a particular teaching context, such as single lesson. We report two evaluations of the effectiveness of the ontology construction, one based on a qualitative concept mapping experiment and the other a quantitative comparison of the computer science ontology with another similar ontology.

**Keywords:** ontologies, student modelling, ontological inference, knowledge acquisition

## 1 Introduction

A software representation of an extensive computer science ontology may be used in a variety of roles in a teaching system. It enables a system to infer from a minimal knowledge base about a student to a more extensive model of their knowledge. Teaching goals can then be inferred by studying this model and expanding concepts the student is modelled as knowing. For example, if there is limited evidence about the student's knowledge, a system may use this to infer the student's likely knowledge of closely related concepts. It also has a potential role in supporting the planning of a teaching sequence, with the ontology used to identify aspects that need to be taught as a foundation for a learning goal; for example, if they are prerequisites.

If we have a large ontology, such as that for computer science, it is important to be able to extract relevant portions of it for a particular teaching purpose. For example, a particular teaching system may have the teaching goal, *data mining*. It may then need to model the learner's developing knowledge in terms of this concept and other closely related concepts. For example, the student may have learnt about *genetic programming* and *knowledge representation*. The teaching system may be able to exploit knowledge of the relationships between these concepts and the current teaching goal.

Rather than creating a subject-area ontology by hand, MECUREO (originally, Model Expansion and Comparison Using Reverse Engineered Ontologies) is able to automatically construct an ontology. It uses an existing, reliable resource to ensure appropriate coverage of terms. This saves a considerable amount of effort and helps minimise errors while maximising breadth of coverage. The generated ontology may later be refined by hand. For example, MECUREO may be able to identify that there is a relationship between two concepts but it may not be able to determine what that relationship is. In this type of situation, the refinement of the ontology would require a person to label this relationship. This task is generally less challenging than the step that MECUREO has automated.

Our approach to the ontology construction overcomes some of the limitations we perceived in existing ontology tools and representation formats such as OilEd [1] and DAML [2]. Specifically, MECUREO uses weighted relationships between concepts. This allows more information to be extracted from the ontology. In addition, the representation format is customised to improve the handling of such a large ontology efficiently and effectively. This is particularly important if the ontological reasoning needs to operate in real time, as the learner interacts with a teaching system. A major reason for the difference between MECUREO's approach and those of many ontological tools derives from the underlying motivation of the work. In the case of MECUREO, our primary motivation was to support reasoning about a learner's knowledge. By contrast, much work on ontologies has been motivated by the goal of facilitating the exchange of data [3].

MECUREO builds lightweight ontologies. It represents an ontology as a digraph. Each edge has a weight. A small weight indicates an edge that connects concepts that are closely related. This is important since it means we can reason about the closeness of any arbitrary pair of concepts in the ontology. We identify the shortest path between them and add the weight values on all the intervening edges. When MECUREO performs a process like this, it performs a breadth-first expansion from a single concept to all those concepts that have a direct link to that concept. We describe two concepts as *peers* if there is a relationship linking them in the ontology.

In Section 2, we give a description of the way that we have constructed a computer science ontology. Then Section 3 outlines the querying process we use to grow ontology graphs from an initial small set of concepts, as well as the process used to match and compare graphs. In Sections 4 and 5 we describes our evaluation process and the results, respectively, and in Section 6 we give an outline of a supplementary model visualisation used during evaluations. We conclude by outlining related work in Section 7 and giving conclusions in Section 8.

## **2 Ontology Construction**

### **2.1 Dictionary Source**

The basis for our ontology was the Free On-Line Dictionary of Computing (FOLDOC) [4]. An example is given in Figure 1 for the word *ontology*. FOLDOC was chosen because of its breadth and up-to-date coverage within the computer science domain, machine readability and licensing. In its entry under *Free On-Line Dictionary of Computing*, the dictionary describes itself as "a searchable dictionary of acronyms, jargon, programming languages, tools, architecture, operating systems, networking, theory, conventions, standards, mathematics, telecoms, electronics, institutions, companies, projects, products, history, in fact anything to do with computing".

Our ontology is generated from an off-line image of the FOLDOC. Individual definitions are parsed, looking for a categorisation for each word and to identify words that are related to it.

## 2.2 Concept Relationships

The consistent grammatical conventions mean that *synonym*, *antonym*, *child* and *parent* relationships can be extracted if certain keywords are present; otherwise an *unknown* relationship is created. For example, acronyms usually become modelled as synonyms of their expansions. Relationships such as that between *call-by-value* and *call-by-name* can be modelled as antonyms and patterns such as *developed at X* or *a type of X* represent child relationships. A list of cross-references after an 'e.g.' in the dictionary usually represent instances of the concept being defined and so that concept is modelled as the parent of the cross-references.

The keyword-relationship mapping is read from a text (configuration) file in the form:

```
<Keywords> ::= { <item> \n }
<item>      ::= <keyword>:  invalid | <defn>
<defn>      ::= <strength> <type>
<strength>  ::= weak | normal | strong | very
<type>      ::= child | parent | dissim | sibling
```

An example is shown in Figure 2. This strategy makes the mapping customisable for other dictionaries.

The final output of the parsing process is a weighted digraph. We explain the procedure in terms of the entry for *ontology* in Figure 1. A visual representation of the output corresponding to this context is shown in Figure 3. This context was chosen as a *typical* entry. In the discussion that follows, *concept* is used to describe an entity at the ontology (conceptual) level, *word* at the dictionary level and *node* at the representation level; similarly for *relationship*, *link* and *edge* with respect to entity associations.

## 2.3 Determining the Nodes

Parsing *ontology* in Figure 1 generates the node *ontology*. It is linked with the *category* nodes for *philosophy*, *artificial intelligence* and *information science*. All phrases in braces ({AI}, {knowledge}, {domain}, etc.) also become nodes with an edge between each and *ontology*.

It should also be noted that, the edges are traversed in either direction for queries. This is true, despite the inherent direction within a definition; the term being defined in the dictionary is the source of all relationships identified from its dictionary entry and it links to concepts in the body of

its definition. So, any dictionary entry that referred to ontology will be considered in a query involving the ontology node.

In addition, it is sometimes the case that a link in an entry is not itself defined. Such references undergo the same matching process. The sensitivity of this process is adjustable. For example, this could cause inclusion, or not, of *Knowledge-Level*. Note that our use of the Porters stemming algorithm [5] gives different stems for *index* and *indices*. Hence, the node *subject indices* remains unmatched (see Figure 3).

## 2.4 Determining the Relationship

We decided that each relationship should be given a weight to reflect the amount of work required to 'travel' from one concept to the next. The type for each edge is determined from keywords around the phrase and the current relationship mapping. Where a link type is identified, it becomes a property of the edge for subsequent processing and output, such as the graphical display in Figure 3. It also determines an associated direction of the edge.

It is often possible to identify a link as being an antonym, or strictly a child or parent, and this information is preserved in the graph. For our experiments, these mappings were empirically determined and refined (see Section 4.1). For example, keywords identified in Figure 1 include *formal*, *see*, *e.g.* and *about*. *See* gives a type to the edge with *subject index* and *about* with (all three of) *domain*, *declarative language* and *universe of discourse*. Here *axiom(s)* is an unfortunate anomalous classification; *e.g.* and *formal* both appear near *axioms* and because *e.g.* was configured with a stronger base weight, it is used for this edge rather than *formal*, giving a 'child' edge rather than the 'parent' type associated with *formal*. This is partly reflected in Figure 3: *subject index* has a bidirectional (sibling) edge and *axiom* has a directed 'child' edge. *About* was not configured with a strong enough base weight to have a distinctive edge in this output configuration.

Examples from other parts of the dictionary include: *laziness* modelled as a 'synonym' for *lazy evaluation*, *declarative language* modelled as an 'antonym' for *imperative language*, *TABLOG* 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. As already described, whether it is called a parent or child depends on the context of the location where the relationship was discovered. Sibling and unknown relationships are undirected.

## 2.5 Determining the Weight

We have defined a base weight for each link type as a constant between 0 and 1. The actual weights are specified in a configuration set. We developed these values experimentally. This type-link value is 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 currently being used as a default until other parameters have been fine-tuned. Small  $C$  values penalise later links in highly branching nodes while a large  $C$  distributes weights for successive links more gradually. Rearranging with the default of 10 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.

Ideally, in choosing a value for  $C$ , the average weight of all relationships (taken individually across all concepts) should equal the average of all *concepts'* average peer weight. This would reflect a consistent distribution of link weights for both highly branching nodes and nodes with few peers. Knowing that this is the case helps choose a *distance* when formulating queries so that growth of models with varying distance is roughly consistent (see Section 3). Experiments showed that the discrepancy between these values was only significant for very small values of  $C$  (less than four). However, if  $C$  is large, there is less spread in the weights chosen, making it more difficult to distinguish nodes that should be filtered when choosing the distance in a query.

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 spreads 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. For example, from the *ontology* extract in Figure 1, *AI*, *knowledge* and *domain* occur before *universe of discourse*, *axioms*, *agents* and *Knowledge-Level* and so the edges to the former are given less weight. In these cases, it is intuitively the right thing to do and while there are many exceptions (eg *logical theory* in Figure 1), the generalisation was found to work well in our initial precision evaluations. The final weight is capped at 1.0 so that a query visits (at least) every node at a particular *depth* (as opposed to *distance*) without additional computation being required. Thus every weight is in the interval (0, 1]. On top of this, there are some special cases. In the extract for *ontology* in Figure 1, *AI*, for example, was not adjusted because it is the first relationship identified ( $p_i = 0$ ). So this edge retains the default base weight, but if we then look to the definition of *AI* in Figure 4 then another pattern is identified – a synonym. So an edge with negligible weight is formed between *AI* and *artificial intelligence* to reflect this property. Links occurring in the body of a definition whose type is determined to be *very strong* are similarly modelled (i.e. their base weight is near-zero). However, these are adjusted depending on their position in the same fashion as normal links. The final result of this example is shown in Figure 3, as a point query, and is further discussed in Section 3.1.

### 3 Ontology Queries

The ontology generated is in excess of 21,000 nodes (after matching). Tools have been written to query the ontology from a single concept, or a set of concepts (for example, a user model). This results in a subgraph of the original ontology graph. As a useful side-effect, nodes as well as edges are assigned a weight that reflects how closely each matches the original query.

### 3.1 Point Queries

A query results in a concept map style representation of all things related to the query node within a certain *distance*. This is useful for browsing the ontology and for general information retrieval, and was the primary method used to evaluate the ontology construction. The distance is specified as part of the query and its reach is determined by the edge weights on nodes. All nodes whose shortest path to the query node is less than or equal to the specified distance are included in the output graph. This is determined in increasing order of path cost. Once the nodes are determined, any edges between them (not only those along the shortest paths) are duplicated. This preserves the structure of the ontology to facilitate further querying on the output graph, to perform structure analyses such as comparisons, and in order to represent all the information available when the graph is displayed.

An example of a point query is shown in Figure 3 from the node *ontology*. This also shows the result of parsing the dictionary extract given in Section 2. Bold, bidirectional edges indicate relationships identified as being synonyms (or other immediate cross references in the dictionary). For example, the relationships between *KR* and *knowledge representation*, and between *neural network* and *artificial neural network*, are identified as synonyms. Bold edges with reversed arrows, such as that between *declarative language* and *imperative language*, represent antonym relationships. Other bold edges indicate very strong relationships, as determined by the keyword-relationship mapping.

Directed edges attempt to identify strict parent-child relationships. However, they are also used for strong links where an explicit type could not be determined (the definition that contained the original link becomes the child). Edges with no arrows represent relationships for which a type could not be determined, with dashed and then dotted line styles representing progressively weaker relationships (see Figure 5 for an example with dotted line styles), as determined by the assigned edge weight. When there is more than one edge between two nodes, this indicates more strength, because the definitions for each of the nodes will have referred to each other.

In Figure 3, the depth limit of 0.85 is restricting the range of the nodes included. All the nodes related to *ontology* are included because there are no weak relationships with ontology and 1.0 is the weight limit for a single edge. However, for the nodes related to *ontology*, only synonyms or other strongly related peers are included. We also require that there is a path from any included node to *ontology*, with a cost less than or equal to 0.85.

A deeper query, also from *ontology*, is shown in Figure 5. As one would expect, the number of nodes is exponentially proportional to the depth of a query. In this figure, we restricted the output for effective visualisation using a minimum peerage of twelve. Highly connected nodes generally represent less obscure concepts and so, are more interesting. This is what makes the peer-based filtering method effective. Viewing the ontology in this way gives the 'bigger picture' to be seen. These queries can also show interesting chains of relationships, along the lines of the 'degrees of separation'. In fact, over 90% of the ontology is within a distance of 3.0 from some highly branching nodes such as *SQL*. This means that at least 90% of the nodes in the ontology are within a maximum distance of 6.0 from *each other*. See also Section 5 and Figure 8.

### **3.2 Multipoint Queries (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, for example, the keywords of a document.

#### **3.2.1 Subgraph Growing**

A subset of nodes from the ontology can be used to *grow* a graph to a specified distance. This can then be grown to include nodes *nearby* any of the existing nodes. The procedure is a form of a query on the ontology. Figure 6 and Figure 7 are graph representations of such queries constructed by MECUREO. Figure 6 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 7 is the

result of similarly expanding the terms *neural networks*, *data marts* and *STRATEGY* after going through an intermediate concept-matching step (see Section 3.2.2). 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 to zero (as in the examples). This can be specified in the query, or it can simply be the *node* weight assigned to a node from a previous query. The graph resulting from the query includes 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. This is essentially the generalisation of a point query to handle multiple input nodes.

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*. That is, each node will be given a final weight equal to the reciprocal of the sum of the reciprocals of weights assigned to this node from each of the input nodes. 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, matched in the dictionary. 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.

### **3.2.2 Concept Matching**

Our current work has assumed that we have been provided with a list of terms that describe each entity being compared. Our longer-term goal is to integrate into the system tools that 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 [5] and edit distance [6].

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 easily 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 if a match is to be searched for.

The procedure used to find the nodes in Figure 7 matches concepts 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 entire ontology) for each concept to be matched. A cut off edit distance [6] is then used to help mitigate incorrect matches unless human interaction is involved (whereby the user can select terms ranked by edit distance). The stemmer used is a version of the Porters stemmer [5], slightly modified to perform on the UK-English used in the FOLDOC resource [4].

For the query "*STRATEGY*" "*neural networks*" "*data marts*", the nodes *ShowCase STRATEGY* ([www://www.showcasecorp.com](http://www.showcasecorp.com)), *neural network* and *data mart* are matched. These nodes were then grown with a distance of 0.8 to produce Figure 7, with a minimum of 3 peers.

### **3.3 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 that model the two entities we want to compare. In the case of Figure 6 and

Figure 7, 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:

- 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;
- numbers of nodes present and common;
- relative sizes of the graphs, so that, for example, we can bias a subset comparison for model classification;
- *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 [7, 8, 9] may also show promise if they can be applied to our ontology without losing detail.

## **4 Evaluation of the Ontology**

### **4.1 Tuning the Construction**

In order to tune the construction procedure, a collection of randomly and deliberately chosen definitions were selected to intuitively evaluate the output and adjust the parameters. This began with selection of representative definitions, which were manually processed to determine the keywords and relationships to derive from them. The base weights and weight function were also adjusted to achieve an intuitively correct result.

Once the initial set of parameters was established, random definitions were queried and the results were compared with the definitions. Adjustments and additions were made to improve the correspondence between the logical structure of the definition and the resulting graph. The ontology

was then rebuilt and changes were checked against previously examined definitions. An error rate was established based on the number of incorrectly classified relationships.

A compromise had to be reached between the error rate and the number of classified relationships. Each addition decreased the number of *unknown* relationships. However, when a change was made that resulted in the number of incorrectly classified relationships increasing more than the number correctly classified relationships, it was reversed. The test set was gradually increased in this way, with randomly selected definitions, until no more changes could be made that satisfied this requirement.

The final size of the test set, not including synonyms, was 193 definitions. This is roughly one percent of the entire dictionary. Further examination may improve the success slightly by adapting to patterns specific to small sets of definitions. However, this is essentially refinement by hand and should be performed on the ontology directly (rather than the automatic construction parameters) so that fewer errors are introduced.

Once the construction parameters were thus tuned, empirical evaluation was performed on the resulting ontology. These were conducted using only concepts not in the final test set.

A more sophisticated keyword identification technique using NLP is currently being trialled. This makes it possible to identify a more detailed correspondence between a relationship and the keyword(s) classifying it. In this way, inappropriate keywords can be ignored and suspicious ones can be weighted accordingly (i.e. penalised). This decreases the error rate dramatically. It also allows more relationships to be determined by examining parts of the definition that are not explicitly cross-referenced for possible links and looking for *matches* for concepts that are known to be definitions in certain parts of each sentence. These too are penalised appropriately to reflect the use of an implicit rule.

## **4.2 Concept Mapping Experiments**

The behaviour of the point queries lends itself nicely to comparison with manually constructed concept maps. A small number of volunteers from undergraduate computer science were given a set of topics (concepts in the ontology) closely related to subjects they had previously

studied. They were then asked to draw a concept map around this concept, after being shown a mundane example of a concept map based around *farm*. There was no intervention with the volunteer until the exercise was complete.

The starting concepts were chosen to reflect the students' area of expertise and encourage uninhibited expansion, as well as represent some key concepts in the ontology. The concepts used were *declarative language*, *SQL* and *device*. This resulted in a collection of hand-drawn concept maps for each concept, which were compared against point queries in the ontology from the same starting concepts. The volunteers were then asked to comment qualitatively on the concept maps generated by the ontology.

### **4.3 Comparison with Trusted Sources**

To perform a quantitative analysis, trusted knowledge bases were consulted. Most often these were significantly smaller, or had a different focus that was not a direct for comparison with a computer science ontology. The main resource used to perform this analysis was the ASIS Thesaurus of Information Science [10]. This contains 1615 concepts from the information science domain arranged, hierarchically, in a tree. However, it touches upon many corporate concepts that the FOLDOC ontology does not cover (e.g. one of the top-level nodes is *business and management operations*).

The flexibility of the construction tools was also put to the test in the evaluation method. Some simple pre-processing on the thesaurus was performed to convert it to a format similar to the dictionary. It was then parsed, using the same tool as that employed to construct the ontology. In this way all the existing querying and visualisation tools could be used to aid the evaluation procedure.

However, the structural elements of the thesaurus and dictionary are not the same. The thesaurus is a tree, and there is only one edge type (although the ordering of children in the thesaurus could be used to give weight to the edges). Evaluation was conducted by examining the distance in the ontology between a concept and its parents in the thesaurus. Stemming was used to assist in matching of concept names.

## 4.4 Matching Evaluation

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 early refinement of the algorithms to be performed.

We note that accuracy is hard to define for this problem. Being able to definitively state that two models are, say, seventy 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.

## 5 Results

### 5.1 Ontology Properties

We note that all of the examples in this paper, and those used to evaluate the ontology, were the direct result of the automatic construction. Furthermore, the definitions corresponding to the nodes in these graphs were not used to tune the construction process (e.g. in determining the keywords to use that decide relationship types).

The most recent ontology generated contains 21,029 concepts and 57,407 directed relationships of which 55,181 are between distinct concepts. The queries used for generation and comparison traverse relationships in either direction, so there is an effective average of 5.25 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. When this occurs the related nodes behave semantically as a single node when encountered in a query. This effectively increases the perceivable peerage.

The performance of the queries with respect to the breadth achieved is indicated in Figure 3 and Figure 8. The exponential increase in query size with respect to distance is seen clearly in Figure 8. As the size of the query approaches the total number of nodes available, this relationship is tapered, as expected, due to the limited scope of the ontology. Note that in this example, a

particularly dense region of the ontology (*SQL*) was chosen as a starting point to achieve consistency at small distances. Most queries yield a distribution that starts more slowly and they do not achieve exponential growth until a distance of approximately 0.7, showing the same decay at a greater distance.

Figure 5 and Figure 9 show the depth and far-reaching relationships that can be extracted from the ontology. These were constructed by filtering out nodes that did not have a minimum number of peers when the graph definition was output.

## 5.2 Concept Mapping Results

The results of the concept mapping experiment are qualitative. Findings are summarised in Figure 10. Of the 122 total nodes drawn, 105 were exact or near matches to concepts in the ontology – around 86%. The remaining nodes were most often very specific instances of a concept, or were parts of the syntax used by a particular concept. Specifically, the following hand-drawn nodes were unable to be matched: screwdriver, spanner, mallet, flat head, Phillips head, rule, constraint satisfaction problem, iProlog, FROM, WHERE, SELECT, UPDATE, INSERT, 1:1, 1:M, M:1, M:M.

There was some discrepancy in the placement of edges. All but one participant's concept maps had a tree structure, despite encouragement beforehand to draw a graph. This contributed a little to the observation that often concepts drawn as being connected via an intervening node in the concept map (perhaps intended as a common parent) corresponded to explicit *sibling* relationships in the ontology (for example, *Prolog* and *logic programming* being connected to *declarative language* in Figure 9). Otherwise, of the 108 edges between matched concepts, only 4 needed more than two edges from the ontology to form a path between them. Specifically, these were the ones (drawn) between *fact* and *Prolog*; *think* and *lazy evaluation*; *cluster* and *index*; and *tools* and *device*.

## 5.3 ASIS Thesaurus Comparison

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 the 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 that 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 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 that 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 the computer science dictionary, and so did not perform as well.

## 6 Supplementary Visualisation

For small models, with less than about one hundred concepts, a traditional graph output can give useful information. MECUREO supports this form of output. It automatically generates an ontological graph in the dot language [11]. This, with the dot graph visualisation tool [12], was used to generate all the examples in this paper. We use such dot displays of details for the relationship strengths and types. The output specification includes additional data to position nodes vertically so that the query nodes that were used to generate the graph are near the top. The vertical placement of other concepts reflects their ontological closeness to these query terms.

## 7 Related Work

The automatic construction of ontologies and extraction of semantic information from machine-readable dictionaries and text is of broad interest. Some important work by Moldovan [13] and Kietz [14] derived an ontology based on a document base within a corporation. Similarly,

Richardson [15] and Miller [16] have built extensive ontological representations for more generic concepts.

We began this research with the goal of building an ontology for parts of our teaching within computer science. We hoped to automate this potentially arduous task by trying to extract an ontology from some of the existing dictionaries. This seemed very attractive since these documents were explicitly designed to help people build ontological knowledge structures in computer science. They seemed to hold precisely the knowledge that we wanted. In the case of dictionaries like FOLDOC and the ASIS Thesaurus, they are also extensive. We wanted to be able to make point queries on such sources as a basis for a range of the forms of ontological reasoning that could be useful in a teaching system. For example, if we are able to determine that the learner knows or is interested in a concept, we would like to be able to use this to reason about other concepts they are likely to know or be interested in. We also wanted to be able to support a variant of the classic knowledge interchange problem. In our case, if we build a lesson about a concept and the learner wants to learn about a related concept, we would like to be able to determine how closely related these concepts are.

The task of matching terms with a similar meaning has been tackled by a quite diverse set of approaches. For example, Becker [17] used formal concept analysis for this task, in the context of document retrieval. In similar work, Suryanto [18] used lattices for inferring user interests. Also similar is the work of Moldovan [13] who addresses the problem of building richer models of people's interests by relying on knowledge elicitation from the user and taxonomic representation of computer terms.

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 follows the spirit of the broad range of work on semantic networks, reviewed in Sutcliffe [19] and Resnik [20].

## **8 Conclusion**

This paper has described our approach to the automatic construction of an accurate, detailed and comprehensive computer science ontology from a dictionary. The dictionary is parsed, giving a

graph with nodes representing concepts and weighted, directed edges representing the relationships in the ontology. MECUREO provides tools for making queries on this large graph. These queries consist of either a single concept or a small collection of concepts. MECUREO grows these query concepts into a limited ontological graph that is intended to represent the collection of closely related concepts and the relationships between them. We have described our evaluations of MECUREO in terms of a qualitative study. This compared individual people's conceptualisations of a sub-domain in computer science against that generated by MECUREO. We have also reported a quantitative evaluation which compared the results of point queries on the ontology that MECUREO constructed from FOLDOC against those from the ASIS Thesaurus. The results from these evaluations suggest that MECUREO shows considerable promise. The next form of testing for MECUREO will be in conjunction with applications that need its ontological reasoning support.

The work we have described is distinctive at several levels. MECUREO exploits the nature of a dictionary, both in terms of its inherent ontological content and its rather consistent form. MECUREO appears to be able to parse the dictionary effectively enough to produce ontological structures that are similar to those that people constructed in our experiments and it appears to produce fairly consistent ontological structures for two similar but distinctive sources. This achievement is novel. Another distinctive aspect of the MECUREO representation is the use of weights on the links between concepts, with more closely related concepts having smaller weights. This means that even though MECUREO builds light-weight ontologies, it appears to be able to reason about the closeness of concepts that are not directly connected. Yet another very unusual aspect of MECUREO is its algorithm for varying the link weights between a dictionary term and concepts in its definition. Another important and unusual aspect of the work is the character of our evaluations. These were both qualitative, based on human judgements, and quantitative. Although these assess just a miniscule part of the whole FOLDOC-based ontology, this part was selected in an essentially random manner and we have every reason to consider it to give good indications of the effectiveness of the elements within FOLDOC.

## 9 References and Notes

---

- [1] S. Bechhofer, I. Horrocks, C. Goble and R. Stevens. *OilEd: a Reason-able Ontology Editor for the Semantic Web*. Lecture Notes in Computer Science Vol.2174, 2001. pp.396. Or OilEd – Ontology Interface Layer Editor, at <http://oiled.man.ac.uk/>
- [2] James Hendler and Deborah L. McGuinness. *The DARPA Agent Markup Language*. IEEE Intelligent Systems, Vol. 15, No. 6, November/December 2000, pp 67-73, or at <http://www.daml.org>
- [3] T.R. Gruber. *Towards Principles for the Design of Ontologies Used for Knowledge Sharing. Formal Conceptual Analysis and Knowledge Representation*. Kluwer Academic Publishers, 1993.
- [4] FOLDOC – the Free On-Line Dictionary Of Computing. © 1993 by Denis Howe, updated regularly, at <http://www.foldoc.org/>
- [5] M. F. Porter. *An algorithm for suffix stripping*. Program July 1980, 14(3):130-137.
- [6] I. V. Levenshtein. *Binary Codes capable of correcting deletions, insertions, and reversals*. Soviet Physics Doklady, Feb. 1966, 10(8):707-710.
- [7] A. Maedche, S. Staab. *Measuring Similarity between Ontologies*. EKAW, 2002.
- [8] A. Maedche, S. Staab. *Making Results Comparable – Cross-Evaluating Ontologies*.
- [9] 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.
- [10] ASIS [American Society for Information Science] Thesaurus of Information Science; (published 2002-02-07), <http://www.asis.org/Publications/Thesaurus/isframe.htm>
- [11] *The DOT Language*, by AT&T Labs; at <http://www.research.att.com/~erg/graphviz/info/lang.html>
- [12] E.R. Gasner, E. Koutsifios, S.C. North and K.P. Vo. *A Technique for Drawing Directed Graphs*. IEEE Trans. Software Eng., 1993, 19:214-230. Software at <http://www.research.att.com/sw/tools/graphviz/>.
- [13] D. Moldovan, R. Girju, V. Rus. *Domain-Specific Knowledge Acquisition from Text*. ANLP, 2000.
- [14] J. Kietz, R. Volz. *Extracting a Domain-Specific Ontology from a Corporate Intranet*. Proceedings of CoNLL-2000 and LLL-2000, Lisbon, Portugal, 2000, pp 167-175.
- [15] S. D. Richardson, W. B. Dolan, L. Vanderwende. *MindNet: acquiring and structuring semantic information from text*. ACL, 1998, Vol. 2, pp 1098-1102.
- [16] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, K. Miller. *Introduction to WordNet: An On-line Lexical Database*. International Journal of Lexicography (special issue), 1990, 3(4):235-312. (Revised 1993).
- [17] P. Becker, P.Eklund. *Using formal concept analysis for document retrieval*. ADCS, 2001.

- 
- [18] H. Suryanto, P. Compton. *Discovery of Ontologies from Knowledge Bases*. First International Conference on Knowledge Capture, 2001.
- [19] 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, pp 33-42.
- [20] 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, pp 95-130.

## ontology

1. <**philosophy**> A systematic account of Existence.

2. <**artificial intelligence**> (From philosophy) An explicit formal specification of how to represent the objects, concepts and other entities that are assumed to exist in some area of interest and the relationships that hold among them.

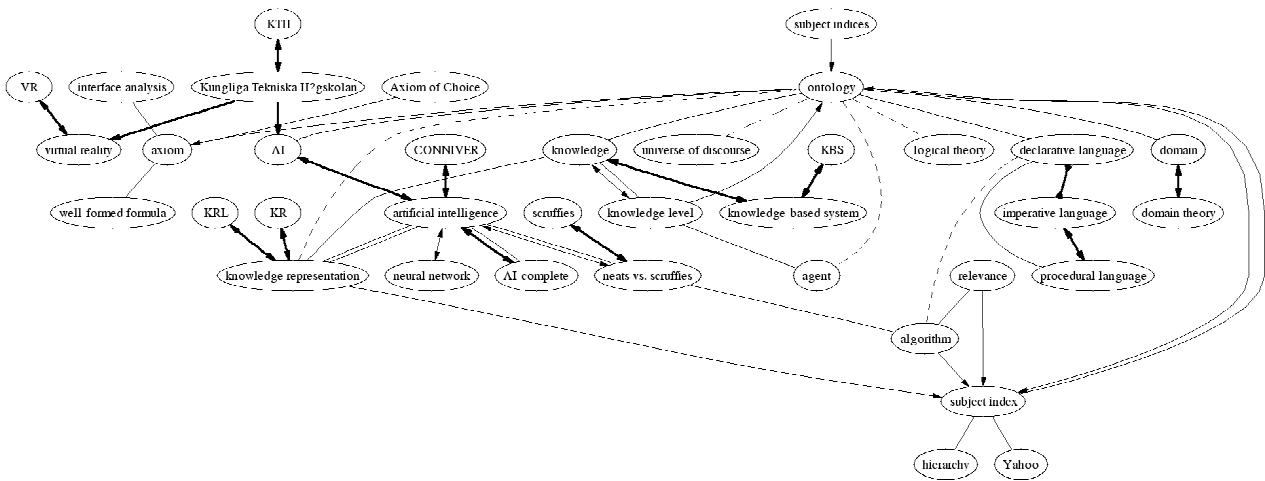
For {**AI**} systems, what "exists" is that which can be represented. When the {**knowledge**} about a {**domain**} is represented in a {**declarative language**}, the set of objects that can be represented is called the {**universe of discourse**}. We can describe the ontology of a program by defining a set of representational terms. Definitions associate the names of entities in the {**universe of discourse**} (e.g. classes, relations, functions or other objects) with human-readable text describing what the names mean, and **formal {axioms}** that constrain the interpretation and well-formed use of these terms. Formally, an ontology is the statement of a {**logical theory**}.

A set of {**agents**} that share the same ontology will be able to communicate about a domain of discourse without necessarily operating on a globally shared theory. We say that an agent commits to an ontology if its observable actions are consistent with the definitions in the ontology. The idea of ontological commitment is based on the {**Knowledge-Level**} perspective.

**Figure 1 – An extract from FOLDOC showing the entry for *ontology*. Key portions discussed are highlighted with bold**

```
analogous:    very sibling
any of:      strong child
archetypal:  very child
as in:       normal parent
compare:     very dissim
compatible:  strong sibling
developed:   strong parent
e.g.:       very parent
eg:         very parent
extension:   strong child
manufacturer: strong parent
opposed:    very dissim
opposite:   strong dissim
replaced:   very sibling
see:       very sibling
variant:    strong child
version:    strong child
```

**Figure 2 – a portion of the (much larger) keywords file used to process FOLDOC**

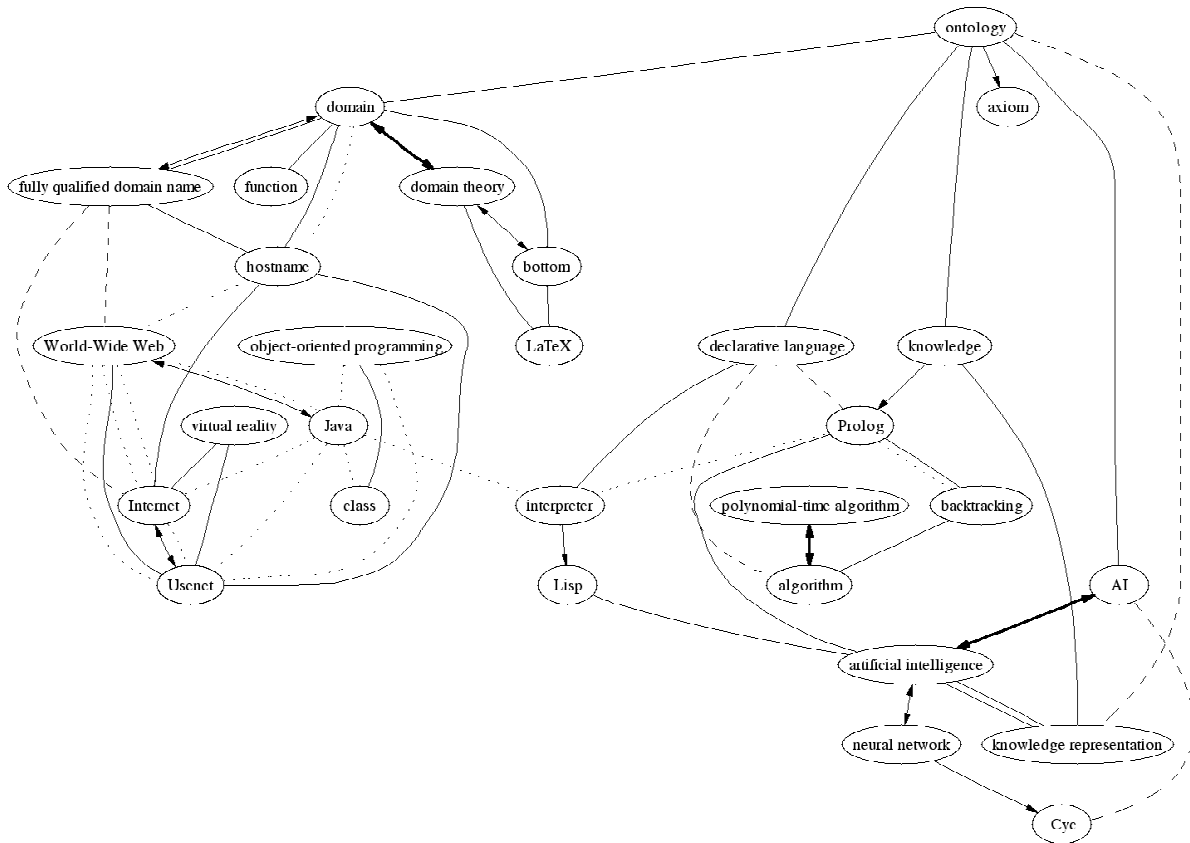


**Figure 3 - Point query from *ontology* with a depth limit of 0.85**

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. These are discussed further in Section 3.1



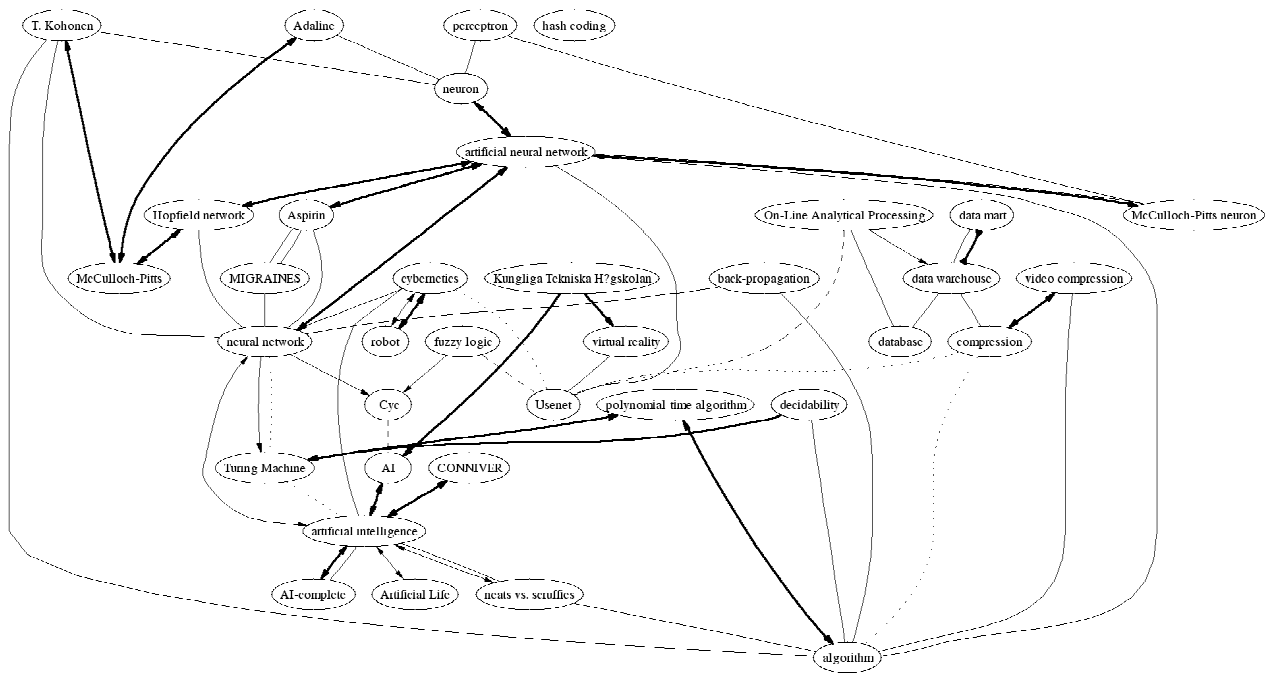
**Figure 4 - The definition for AI**



**Figure 5 - Point query from *ontology* with a depth limit of 1.35 and a minimum perage of 12**



**Figure 6 - Model expanded from the concepts *knowledge representation, genetic programming and data mining* with a depth of 0.73 and minimum peerage of 3**



**Figure 7 - Model expanded from the concepts *STRATEGY, neural networks and data marts* with a depth of 0.9 and minimum peerage of 3.**

Note that in both Figure 6 and Figure 2, the node whose label is 'ShowCase STRATEGY ([www://www.showcasecorp.com](http://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 (that is, with the ShowCase URL).

