Towards Ontological Reconciliation for Agents

Leon Sterling & Kendall Lister
Intelligent Agent Laboratory:
Department of Computer Science and Software Engineering
The University of Melbourne, Victoria, 3010, Australia
leon,krl@cs.mu.oz.au

Abstract

This paper addresses issues faced by agents operating in large-scale multi-cultural environments. We argue for systems that are tolerant of heterogeneity. The discussion is illustrated with a running example of researching and comparing university web sites, which is a realistic scenario representative of many current knowledge management tasks that would benefit from agent assistance. We discuss efforts of the Intelligent Agent Laboratory toward designing such tolerant systems, giving a detailed presentation of the results of several implementations.

1 Introduction

Useful knowledge systems inevitably incorporate vast amounts of information. The evolution of the computer as a data processing device, and computer networks as communication media, has provided the technical means to aggregate enormous quantities of information. Our capacity for accumulation, storage and reproduction of data and information has out-paced our ability to perceive and manipulate knowledge. These three observations are not new – Vannevar Bush identified just such a glut of knowledge and information over fifty years ago. He proposed a technological solution in the form of the memex, an enlarged intimate supplement to memory that anticipated the hypertext systems of today [3].

The development of a (pseudo-)global communication infrastructure that provides means for the publication, comparison and aggregation of apparently limitless amounts of data, i.e. the WWW, has changed the way we can manipulate information. We have created a potential to ask questions as individuals conducting our daily lives that previously would have been dismissed as infeasible unless one had the resources of a dedicated organisation. For example, with the entry cost of publishing a web site effectively negligible, the university that does not do so is the exception rather than the rule. Consequently, thousands of descriptions of courses, programs and facilities are available for us to peruse. There immediately arises the need for comparison. It is natural to ask reasonable and seemingly simple questions such as “Which faculties offer courses in applied machine vision?” or “Which campuses provide accommodation facilities for post-graduate students?”.

To answer questions like these, we could fairly easily compile a list of university web sites. Each site could be visited in turn. Through browsing or searching we could uncover the information relevant to answer our question. We could then compare the results of our research from each site to formulate an answer. Many people perform this very task every day.

The question that interests this paper is why our computers can’t do this for us yet. The followup question is how to approach the issue of enabling our computers to search and filter information to ask specific questions. While search engines are becoming increasingly powerful, and answer some specific questions well, they do not have anywhere near the generality needed.

The example of university service descriptions is useful to study. Finding information from university sites is a real problem. Universities as institutions tend naturally to develop and often then actively promote their individuality. Their local culture flavours their presentation of information that must then be reconciled with information from other institutions that apply their own cultural characteristics to their publications. If we are to manage knowledge from a variety of sources effectively, we will need the assistance of software that is culturally aware and is capable of negotiating the conflicts that arise when such heterogeneous knowledge is juxtaposed.
2 Organisational Culture & Communication

Today’s reality is that knowledge from large numbers of heterogeneous sources must be integrated in such a way that any differences in representation and context can be effectively reconciled. The ability to work with knowledge from incongruous sources is becoming increasingly necessary [15] as the focus of information processing moves beyond intra-organisational interaction and begins to transgress borders, whether departmental, corporate, academic or ethnic. Organisations, whether companies, universities, industries, or nations, develop unique cultures as they grow. Organisational culture is considered to be both constructive and inhibitive as far as the day to day operations of the organisation are concerned. In the context of knowledge management organisational culture creates significant barriers to inter-organisational communications and transactions.

Organisational cultures arise as individual organisations develop mechanisms, procedures and representations for dealing with the issues that they face. Inevitably, because these cultures are generally developed in isolation, each organisation arrives at different solutions to what are often very similar problems. To work effectively in an organisation, individuals often must disregard their personal approach to a situation in lieu of an agreed common understanding shared by the other members of the group. We do this naturally when we work together on a problem. Teamwork and the ability to understand another’s point of view are recognised as desirable qualities. Such qualities are also becoming desirable in software as agents play an increasing role in our communication and collaboration.

There are also disadvantages in requiring every member of an organisation to follow a centralised doctrine. Standardisation can result in inflexibility and a reduced ability to adapt and cope with a wide variety of situations effectively. Lack of flexibility is exacerbated when organisations attempt to interact with external groups. People inevitably find that even when they think that they are working in similar domains and facing common problems, they are unable to communicate effectively. Cultural differences between organisations, differences that arise as each organisation attempts to codify its individual approach to various situations, create impediments. The streamlining that appeared necessary for efficient operation within each organisation now stands as a barrier to interoperation and sharing of resources. Exactly such an incongruity also manifests in the information and knowledge generated by organisations. The problems faced by software agents negotiating such data are analogous.

When we suppress our own intuitive understanding of a situation and attempt to adopt a standardised, agreed upon approach, we increase our ability to interact with others who have similarly adapted their individual understanding to that of the group or community. But we also lose something in the process: context and generality. An efficient understanding of a situation is like a model. The more closely it describes a particular situation, the less effectively it describes a general class of situations. As we move from a general conceptualisation of a situation rich with semantic flexibility to a specific understanding, we tend to eschew context. The very generality that gives us the ability to deal with many varied and new situations is a barrier to communication. At the same time that ambiguity allows adaptation, it prohibits individuals from establishing the certainty of agreement that is necessary for confidence that each understands the other. Standardisation of practices and understandings does not create a panacea for the difficulties of communication and collaboration, as organisations discover. On a small scale, adoption of standardised approaches helps individuals to cooperate and achieve goals too large for a single person. On a larger scale, the effort required to establish and prescribe global standards and common approaches grows rapidly beyond feasibility as the number of participants and the amount of data being manipulated increases. As our ability to communicate and interact across cultural borders increases, so does our desire to do so. If our software tools are to scale, they must be provided with reconciliation capabilities.

3 Our Software Colleagues

Computers can be viewed as an extreme example of co-workers with poor teamwork and communication skills. When specifying a task for a software application or agent, we must specify every step in precise detail, detail that will generally remain constant throughout the life of the software. Humans are able to adjust the level of abstraction at which they conceptualise a particular situation. Computers by contrast have the capacity only for comparatively very low levels of abstraction. As machines that follow explicit instructions to the letter, their operation is analogous to the most procedural organisational standards, and unsurprisingly they adapt to new situations with great difficulty.

Traditional computational paradigms require that computer-mediated representations of information and knowledge be exact and literal; for a computer to process information requires simplistic structuring of data and homogeneous representations of concepts. In order to maintain consistency during processing, traditional approaches require that each participant in a system, whether human or software, subscribes to a common understanding of the concepts within the system. In other words, traditional knowledge systems require the adoption of an absolute ontological world-view; deviation from a priori agreed terms and understandings results in a breakdown in
communication and loss of consistency through the system.

Ontological homogeneity has worked well for systems with little direct human interaction, when the computers can be left to sort out technical details and humans can work at a level removed. Isolating the technical details of a system from those areas with which humans interact permits engineering of the technical aspects to create an optimised environment. The World Wide Web is an example of a large-scale system where the level at which humans interact with the system is quite separate from the level at which machines interact with each other. We write web pages and read them by navigating along hypertextual paths, while machines manage domain name resolution, protocol selection, transmission of data and rendering of text and images.

The gap between the activities of humans and machines is highlighted by the problems that occur when we try to make machines work closer to our level as we attempt to automate various functions that we currently perform manually. The example of this most recognisable to the ordinary web user is searching for information, an obviously difficult problem that has yet to be solved to our satisfaction. But a more far-reaching problem is that of integrating the vast quantities of information available in such a way that we can seamlessly assimilate whatever sources of data are most appropriate to the task at hand, whatever that task may be.

4 Reconciling Conceptualisations

The ability to manipulate concepts at varying levels of detail and to match the level of detail to the needs of the situation at hand is an effective tool for processing knowledge and communicating. Being able to subsume detail within conceptual units of knowledge allows us to overcome the natural limits of our processing capacity. Although there appear to be cognitive limits on the number of concepts we can articulate at any given time, we have the critical ability to ‘chunk’ collections of knowledge into single units [11, 5], effectively providing a capacity to search through information webs both widely and deeply as necessary.

When the scope of an information or data handling task becomes too great for us to process in a reasonable amount of time, we conscript computers to assist us with storage, recall and simple processing. By handing low-level information processing tasks to machines, humans are freed to consider issues at higher levels of abstraction. To continue to increase the assistance provided by computers as we work, our tools must be elevated to higher levels of abstraction.

As knowledge travels through progressively lower levels of abstraction, its context degrades as generality is replaced by specificity and logical operability. Humans require some specificity in order to communicate successfully. The desired degree of consistency of conceptualisations determines the extent of specificity that is necessary. It has been suggested that consensus between participants is not always necessary for successful collaboration [1, 12].

Humans are capable of identifying mismatches of understanding in our communications and negotiating shared perspectives as we interact with others [2]. Human natural language is neither precise nor predictable, and this seems to reflect the way that we understand the world through our internal representations and conceptualisations. When we express ourselves in natural language, we often encounter confusion and difficulty as others attempt to understand us. This requires us to explore alternative expressions, searching for representations that others understand. We do this naturally, and our attention is drawn to the process only when it fails. However we are generally capable of finding enough common ground for communication of knowledge to proceed. We are often even able to convey basic information without a common language, as any tourist who has managed to gain directions to a restaurant or train station with much waving of hands can attest.

Computer mediated communication removes many of the mechanisms that we use to assist our process of reconciling conceptual differences during interpersonal communication, and generally leaves us at best with spoken or written language. We use the term ontological reconciliation for the process of matching conceptual differences. Anecdotal evidence documents the detrimental affects on effective communication of using a ‘low bandwidth’ medium such as a telephone or a ‘high latency’ medium such as the post or e-mail. The effects of limited representation of concepts are exacerbated when computers are no longer just the communication medium but also themselves participants in the communication and knowledge manipulation. In order for the processing power of computers to be utilised, knowledge must be reduced to a representation suitable for logical operations. Fitting knowledge to logical representations is largely a subjective process. Decisions must be made about how to express complex concepts in relatively constrained languages; these decisions are made by people whose choices of representation and expression are influenced by their own cultural background. Consequently, as context is lost problems then arise as other organisations with different cultures, or even just individuals with different conceptualisations, attempt to understand the logical representation and rebuild the original knowledge.

Let’s return to our running example of university web sites. We accept that universities must deal with teaching and research. Most universities offer undergraduate degrees in the areas of engineering, arts, science and commerce. But when it comes to describing their activities, where one university may use the word course to refer to a particular degree program, another will use course to mean an individual subject within a degree; a third institution may use
course to describe a particular stream or program within a degree. Some institutions will say unit where others say subject or class. Due to their own individual organisational cultures, different institutions use different vocabularies to describe their activities. The researcher wishing to compare the services provided by different universities will generally quickly identify the differences. Through an understanding of the knowledge domain of university activities and services, a researcher will be able to translate between terms, usually assimilating them into their own personal ontological understanding, which itself will be shaped by personal experiences. If they are from a university that uses course to mean a unit of teaching and program to describe an undergraduate degree, they will probably translate the descriptions from other institutions into this ontology. If they are not from a particular university, they will probably draw on whatever experience they have of academic institutions, and if they have none, they may build their own ontology from the collection of university representations.

To create software agents that can handle this level of ontological complexity would seem to be very difficult. Why then is it preferable to simply agreeing upon a global ontology to which all agents subscribe, a centralised language of understanding and representation, or even a global directory of multiple re-usable ontologies from which agents select as necessary? Ontology creation itself is very difficult. It requires the ability to define many concepts precisely and consistently. It requires the ability to predict appropriate assumptions and generalisations that will be acceptable to most, if not all, people. It also requires universal access and distribution infrastructure, and a well-established and accepted knowledge representation format. It requires some way to address the desire for agents and humans to interact at variable levels of abstraction as particular situations demand. It requires constant maintenance to ensure freshness and currency, yet also must provide backward compatibility for old agents. It requires that agent developers familiarise themselves with the prescribed knowledge representation formats, ontologies and protocols and adapt their own development efforts to suit them. These issues make a global ontology infrastructure unsuitable as the sole approach, and it is our belief that effort spent adding tolerance of heterogeneity to systems will provide greater benefit as we begin to introduce agents to our multi-cultural world.

In addition to the practical benefits, one of our strongest desires for tolerance of heterogeneity for software systems is rooted unashamedly in idealism: humans manage to resolve ontological differences successfully, in real time and ‘on the fly’. This ability gives us much flexibility and adaptability and allows us to specialise and optimise where possible and yet generalise and compromise when necessary. Therefore, it seems both feasible and desirable to have as a goal a similar capability for software agents.

If we are to make effective use of multi-cultural data from heterogeneous sources, we need ways and means to reconcile the differences in representation. If we are to work efficiently to solve large information problems, we need the assistance of automated mechanisms. To achieve both, we need systems that are tolerant of heterogeneity.

Reconciling ontological differences requires understanding the difference between concepts and their representations; in semiotic terms, appreciating the difference between the signified and the signer. Reconciling ontological differences means reading multiple texts that represent identical, similar or related concepts and being able to work with them at the concept level rather than at the level of representation.

For databases or XML documents, ontological reconciliation might be as simple as realising that two fields in different data sources actually contain the same class of data. On the other hand, it might be as complex as deciding that articles from an economics magazine and an automotive magazine are discussing different topics even though they both have ‘Ford’ and ‘analysis’ in their titles, something that current search technologies would be unlikely to realise.

As the number of data sources available to us and our ability to access them on demand and in real time is increasing, the overhead of pre-constructing a complete ontology for a given interaction becomes less and less viable. Large scale interconnectedness and increased frequency of data transactions across organisational and cultural borders leads to a reduction in the useful life of any context constructed for a particular transaction. Just as we are able to establish contexts and construct suitable local ontologies as needed for particular interactions, if we want to be able to include software agents in our higher level communication and knowledge management they will need to be capable of similar conceptualisation.

5 IAL Developments

The Intelligent Agent Laboratory at the University of Melbourne has been working for a number of years on knowledge representation and manipulation for information agents [13, 14]. When considering how best to structure knowledge for information agents, two questions arise: what types of knowledge should be pre-defined and what should be left to be learned dynamically? Research at the Intelligent Agent Laboratory addresses these questions in both theory and practice; the remainder of this paper describes three recent projects.
5.1 Finding sports scores

At first thought, finding sports scores may seem a straightforward task. However, the complexity of building a general program to recognise scores can easily be appreciated by looking at the sports results in a daily newspaper. Score formats differ, the significance of numbers are different, the order of two teams sometimes reflects winners and losers, and sometimes where the game was played. Using capitals for names can reflect home teams, in U.S. Football for example, or can reflect Australian nationality in tennis as reported in Australian newspapers. A lot of terminology and style of reporting is cultural as anyone who has lived in a different country can attest to.

There is an extra dimension to consider for an information agent. The desired information must be actually located on the web page. Finding the score of a team means locating the team name, which is relatively straightforward, then locating the score and opponent from the surrounding context. This requires special knowledge. Note there can be more than one occurrence of the team name and other sources of confusion.

The first information agent built in the Intelligent Agent Laboratory, by Andrew Cassin, was called IndiansWatcher and handled baseball scores. It sent a daily e-mail message, to the first author in Australia, with the result of the Cleveland Indians baseball team for most of the 1996 American League baseball season. IndiansWatcher visited the WWW site of the Cleveland Indians, checked if there was a new Web page corresponding to a new game result, and if so, extracted the score and sent a mail message. The development highlighted issues of knowing what a baseball score was, what rules were for washed out games, and other baseball miscellany. Both game specific and site specific knowledge were essential.

The next, more elaborate, information agent was built by Alex Wyatt and retrieved soccer scores from a variety of international leagues. Some useful heuristics that emerged were:

1. Exploit table structures where possible. Free text versions of scores are harder in general to process.
2. Exploit typography, for example semi-colons instead of commas can delimit games, and HTML typography is very useful.
3. Have expert handlers of date formats.
4. Have dictionary support to identify words as opposed to team names, though words like united can be confusing.
5. Use common sense knowledge for checking sensibility of scores. One version of the heuristic produced a score of 69 to 23 which turned out to be the minutes in which the goals were scored.

The heuristics for soccer were readily adaptable to other team games, including rugby, American football, basketball, and Australian Rules football. This resulted in the system SportsFinder [9].

There were several types of knowledge in SportsFinder.

- General Internet knowledge, such as which tags end HTML blocks, and which HTML tags are line-breaking tags.
- General Sport Knowledge, such as that scores are usually in the format [integer]-[integer] or [team_name] [integer] format.
- Sport-specific Knowledge, such as maximum and minimum conceivable scores in a game, that baseball usually has nine innings, while Australian rules football has four quarters, etc.

It was readily apparent that naive approaches had difficulty. Here are some lessons learned.

- Don’t rely on a fixed format, as it doesn’t work and breaks easily. This had already been discovered in building IndiansWatcher.
- A fixed heuristic for scores and team names is likely to give mistakes. One amusing error was the following. From the request for Manchester’s score from the following line, Oct 3 - Manchester 2 - Liverpool 1 Match Report the message returned was “Bad Luck, Manchester lost to Oct 3-2”. This led to the development of a date expert.
- Ignore information in brackets, such as in Manchester 2 (Smith 47, Jones 81) Liverpool 1 (Brown 51).
- Don’t rely on single numbers. For American football results, the last number, which is the total of the four quarters, needed to be returned. From Buffalo Bills 0 3 11 2 16 vs Miami 9 2 4 6 21, the message returned should be “Bad Luck, Buffalo lost to Miami 16-21”.

A pleasing feature of SportsFinder was the ability to add new sports on the fly. A CGI script could prompt the user for the sport name; URL for results; list of teams; format for display of scores; maximum and minimum conceivable scores; and whether information in brackets should be considered. A variety of sports were added. A particularly pleasing example was a Dutch draughts competition where results were immediately retrieved with no tweaking at all despite the page being in Dutch, and no prior knowledge of the format having been known.
SportsFinder was extended by Hongen Lu for ladder-based sports, such as golf and cycling. This allowed a further extension to cooperative information gathering. An interesting question that we have considered is finding the best sporting city. We have simple demonstrations for Australia and Italy [16, 9].

5.2 CASA

Classified Advertisement Search Agent (CASA) is an information agent built by Sharon Gao. It was developed to assist people to search on-line advertisements about specific domains including rental properties and used cars [4]. It was built as a prototype to demonstrate that the effectiveness and flexibility of information agents could be increased while reducing their development cost by separating their knowledge from their architecture. CASA discriminates between different classes of knowledge in order to maximise the reusability of constructed knowledge bases. CASA can learn how to interpret new HTML documents, by recognising and understanding both the content of the documents and their structure. It also represents a framework for building knowledge-based information agents that are able to assimilate new knowledge easily, without requiring reimplementation or redundant development of the core agent infrastructure. In a manner that draws on similar principles to object-oriented analysis and design methodologies and component-based development models, an agent shell developed from CASA [8] allows simple construction of agents that are able to quickly incorporate new knowledge bases, both learnt by the agent itself and incorporated from external sources.

CASA classifies knowledge into three categories: general knowledge, domain specific knowledge and site or source specific knowledge. Each category is independent from the others, and multiple instances of each category can exist. General knowledge gives a software agent enough information to understand and operate in its environment. General knowledge is knowledge that is true for all information sources, and is independent of specific domains and sites. The set of general knowledge developed for CASA describes on-line web documents, and includes knowledge of the components that make up an HTML document such as what are tables, paragraphs and lines, as well as knowledge of what a web page is and how one can be accessed.

Domain specific knowledge provides an information agent with a basic understanding of the area in which it is required to work. This knowledge is true for a particular field and is independent of site or source specifics. For the case of university services, domain knowledge would generally include the concepts of students, lectures, theatres, semesters, professors and subjects, as well as ontological relationships such as the idea that students take classes, classes cover particular topics and occur at certain times during the week at certain locations, and that particular subjects make up a course. Because domain knowledge is independent of site specific knowledge, it can be re-used across numerous sites and should remain useful into the future.

Site specific knowledge is true for a particular information source only. Site knowledge is specific and unique, but necessary for negotiating the contents of a particular information source; it provides a means of understanding the basic data that comprise an information source, for a particular representation. Continuing the university web site example, site specific knowledge might encode the particular pattern or format in which a certain institution presents a description of a unit of teaching, or of a degree, including information such as table structures, knowledge unit sequences and marker text that locates certain classes of information.

The three categories of knowledge that CASA manages provide different levels of operational assistance for the information agent. General knowledge enables an agent to act and interact in a particular environment, providing the basis for navigation and perception and giving the agent a means by which to internalise its input. Site specific knowledge permits an agent to assimilate and process information from a particular source, which is a necessary ability if the agent is to perform useful tasks. Domain specific knowledge sits between general and site specific knowledge, giving a conceptual framework through which an agent can reconcile information from different sources. Domain specific knowledge can also assist an agent to negotiate unfamiliar information sources for which it has no site specific knowledge. Domain knowledge can be used in conjunction with general knowledge to analyse a site’s conventions and representations and to attempt to synthesise the site knowledge necessary to utilise the new information source. Because domain knowledge is not tied to a particular representation, it can be adapted and applied to a variety of different sites or data sources, significantly reducing development time for information agents.

5.3 AReXS

Automatic Reconciliation of XML Structures (AReXS) is a software engine built by Dominic Hou that attempts to reconcile differences between XML structures that encode equivalent concepts. It is able to identify differences of expression and representation across XML documents from heterogeneous sources without any predefined knowledge or human intervention [6]. It requires no knowledge or experience of the domain in which it works, and indeed is completely domain independent. It uses Example-Based Frame Matching (EBFM) [7].

The success of AReXS relies on its ability to identify and resolve the differences in representation that re-
sult from sourcing data from a multi-cultural environment. For example, a pair of XML documents from different sources, both describing services offered by universities, might contain attributes named SUBJECT and UNIT respectively. If the two attributes happen to both signify self-contained units of course work, an agent with no prior domain experience or knowledge will have little hope of realising this. AReXS resolves this discontinuity by considering the values of instances of the attributes as well as the attribute names, deriving confidence in a match from similarities in either comparison. If one document contains the statement <SUBJECT>Introductory Programming</SUBJECT> and another contains a similar statement <UNIT>Introduction to Programming</UNIT>, AReXS is able to consider the possibility that the two attributes SUBJECT and UNIT are in this context signifying the same concept. If further correspondences could be found between other instances of these same attributes, the confidence of a conceptual match would increase.

AReXS analyses XML structures and identifies matching attributes, generating a map of equivalence between concepts represented in the two documents. Conceptual equivalence is identified based on lexicographical similarity between both the names and the values of attribute XML tags in each document. Matches are assessed to deduce structural similarities between documents from different sources. By repeating this search for semantic correspondence across other pairs of attributes generated from the contents of the XML documents under consideration, AReXS is able to build a local context for data and then use this context to reconcile the ontological differences between XML documents.

To establish the extent of the context shared by pairs of documents, the AReXS engine uses the Character-Based Best Match algorithm [10] to evaluate textual similarity between the names and values of attributes. Such a string based comparison works well to filter out simple manifestations of local cultures; for example, one university web site may choose to include the identification number of a subject in the name of the subject while another may not, opting instead to have a second attribute containing a numeric identification code for each unit. While AReXS will not be able to realise that the number in the name of a subject from one university corresponds to the numeric unit code from another, it will generally conclude from the similarity of the names that units and subjects are conceptually compatible in this context.

Applying a textual similarity analysis on real data is likely to generate a large number of candidate concepts that may or may not contribute to the local context of the data. AReXS increases its confidence in a candidate for equivalence depending on the uniqueness of the matches between attribute pairs. The uniqueness function described by [7] is used to establish the likelihood of a textual match between attributes actually revealing a shared, unique concept, based on the principle that the more common a concept is across significantly different attributes, the less rich the concept is and thus the less there is to be gained from considering it as part of the data context.

The results of tests based on sample real world data from web sites including amazon.com, barnesandnoble.com, angusandrobertson.com.au and borders.com show that AReXS is capable of accurately identifying conceptually equivalent attributes based on both the attribute names and sample instances of the attributes. These web sites were chosen as useful examples for two reasons. Firstly, they are live, international representatives of the types of data source with which people desire to interact (and in fact already do interact) on a regular but casual basis, and secondly they provide data that by its nature is open to subjective decisions during the process of choosing a logical representation. The casual nature of the interaction that people generally have with sites such as these is important, as discussed earlier in this paper.

6 Further Thoughts

AReXS is just a prototype that demonstrates the potential for automated ontological reconciliation of data sources from a culturally heterogeneous environment. Because the effectiveness of the concept matching algorithm is improved by examining more instances of the data, and each data attribute must be examined to increase the confidence of the conceptual matches, AReXS currently suffers from poor scalability as the complexity of data objects increases. The CBBM algorithm used for comparing attribute names and values is heavily biased toward text strings and struggles with variations of numerical data. Due to the modular design of AReXS, this component of the engine could be significantly improved with a combination of simple heuristics, alternative matching algorithms and possibly even the capacity to pre-populate the data context with concepts previously observed or learned. AReXS currently can only work with flat or un-nested XML structures, although it is quite reasonable to imagine extending the principles it demonstrates to more complex data structures, or even incorporating the AReXS concept matching engine as a component in a more sophisticated data analysis system.

Drawing on the example described earlier of university service descriptions, if one institution chose to present teaching units with an attribute of the form <UNIT>Machine Vision (Semester 1)</UNIT> and a second institution opts for two attributes <SUBJECT>Machine Vision</SUBJECT> and <SEMMESTER>1</SEMMESTER>, it is possible to
see that a software agent could use analysis techniques similar to those implemented in AReXS to realise that both attributes from the second source are encoded within a single attribute of the first source.

A significant benefit of categorising knowledge into categories is that knowledge can be more readily reused and incorporated into other agents. Compartmentalising knowledge also allows agents to teach each other about new information sources or even new knowledge domains.

Acknowledgements

This paper was supported by a University of Melbourne Research Development Grant. Thanks to the members of the Intelligent Agent Laboratory at the University of Melbourne for many useful discussions. An enlarged version of this paper has been submitted to the Australian Artificial Intelligence Conference 2001.

References