XML-Based Offline Website Generation

Florence Armardeilh

Information Technology and Electrical Engineering
The University of Queensland
St. Lucia QLD 4072, Australia

florence@dstc.edu.au

Peter Becker

Distributed Systems Technology Centre
The University of Queensland
St. Lucia QLD 4072, Australia

pbecker@dstc.edu.au

Abstract

The approach and the tool XWeb[1]*, presented in this paper, shows one way to create websites from XML and other input files which can then be uploaded onto standard web-servers. The system uses an extra input file describing the structure of the content and the processes for creating the website. This information is also used to create the navigational elements in the output. Generating the content offline avoids having additional requirements on the server side such as CGI interfaces or Servlet engines.*

Keywords Document Management, XML, Hypermedia, Website Generation, Processing Model

1 Introduction

HTML was originally designed for writing documents with simple structures that are hyperlinked with other documents. This design was very successful and created what we know nowadays as the World Wide Web. Unfortunately HTML did not address two important issues when managing documents on a larger scale: there is no support for navigational structures outside the documents and content and layout information is highly intermixed.

The screenshot in Fig. 1 shows a typical webpage, which contains navigational structure on the left hand side, using a nested layout for two levels of navigation, a number of layout elements plus information content. The information content in this example comes from two different sources: a description of the person and a BibTeXML[2] file describing the publications of the whole workgroup, a single publication can appear on multiple pages if it has more than one author.

Separating these different aspects of a site has been addressed in a number of ways, the most common approaches being the use of frames as an HTML extension and the use of scripting languages to create websites dynamically from databases or files on the server. Frames introduce a number of technical problems on the client side and often introduce accessibility issues for the visually impaired. They also do not provide a linkage between the document structure given and the navigation structure implemented as an HTML document in one of the frames. It is left to the website creator to ensure consistency between the two.

Using scripting languages like PHP, ASP, JSP, Perl, Python or others on the server side introduces two issues. The first is that the server needs to be capable of running the scripting language used via some interface, usually the Common Gateway Interface (CGI). This raises the requirements on the server side, especially if the data itself is stored in a RDBMS instead of plain files – quite common with this approach. Another problem is the computational effort. Creating pages dynamically means running a script every time a page is requested, which increases the need for computational power on the server side significantly.

The most common way to separate content and layout in the more modern HTML versions is the use of Cascading Style Sheets (CSS). CSS introduces layouting information that can be used to change the rendering of different HTML elements, either by tag names, by their position in the document or by additional information attached to them. CSS is a quite powerful way to define how elements are to be rendered in a browser, although it does not allow structural changes.

[1] `http://xweb.sourceforge.net`
[2] `http://bibtexml.sourceforge.net`

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

Figure 1: A page of a website presenting an academic working group.

2 XML

The World Wide Web Consortium (W3C)[3] proposes XML as an approach to separate content and layout. XML itself offers a simple, tree-based data model called the Document Object Model (DOM), which is mapped into an Unicode-based text format.

The eXtensible Stylesheet Language for Transformations (XSLT) can be used to transform XML documents into any other format, including standard HTML files and other web-based formats. The XML input can have a structure that reflects the content semantics, while all layout information is moved into XSLT programs. These XSLT programs can then be applied either on the client side (i.e. in a browser) or on the server side.

3 XWeb

The approach and the tool presented here try to fulfil the following goals, addressing some of the issues discussed above:

- XML and XSLT shall be leveraged to separate content and layout;
- CSS shall be used to define the rendering details;
- the website navigation shall be generated from the content structure;
- the processing should be done offline, i.e. a result shall be created using a compiler-like approach, which can then be uploaded onto a standard webserver.

The software architecture implemented uses a three-layer model to fulfil these design goals in a flexible way. The lowest layer is a Java-based processing engine, which collects all information needed for the creation of the navigation and calls different processors to do the document processing, the most important processor being an XSLT-processor. The second layer consists of XSLT programs, XML formats and CSS files to create websites with specific structures and designs. The XSLT programs can use parameters to allow flexible layouts, e.g. for deciding the position of the navigation on the pages. As a third layer, specific graphical frontends for creating those sites can be implemented.

This layered approach allows for a high level of abstraction and therefore reuse in the lower layers and also accomodates different user types with different needs for control and comfort.

The approach taken for defining a website in XWeb is using a separate file defining the structure of the content in an XML-based format. The toplevel element of the file is called `website` and has some attributes defining input and output locations as well as the target webspace. Below this, the two elements `structure` and `layout` define the content structure and the layout resp., the latter can be given multiple times to create different versions of the same site, e.g. for different browsers.

[3] `http://www.w3c.org`

3.1 The Content Structure

The two core concepts in the structure definition are denoted by `section` and `entry` elements. A section defines a part of the website, which usually has a name attached to it used in the navigation rendering, and usually maps into subdirectories in the input and output. This information is attached as attributes on the `section` element. A `directory` is similar to a section, but will not create a navigation element.

Entries define single pages in the website. They are also named for the rendering of the navigation, they need an input and an output filename and an identifier for the type of process that has to be executed to transform the input file into the output file. This process has to be defined in the layout section later on. If a single file should be omitted in the navigation, the `file` element can be used instead.

The code in Fig. 2 defines a section of the website with three pages and two files which will not be shown in the navigation, in this case they are actually linked from the output file created by the entry above. Two different processes are used, attached to the files via their type definitions, the entries are all of type "XHTML" while the other two files are of type "copy". The processes for these file types are then defined in the layout sections.

Compared to using directory structures for defining the website navigation and file extensions for the processes, this XML structure gives more control on the way files are processed and offers ways to separate the input and output structure, as well as the input and output file names and the names rendered in the navigation structures. Additional information can be added in the same way, in its current state XWeb allows adding types to the section (for rendering graphical buttons) and an additional *title* attribute can be given to distinguish between a long and a short name for an entry, usually used with banners across the pages.

3.2 Layout information

The `layout` sections of the XML website file define how the website should be generated from the given content. In the most simple case one `layout` section defines a process for each of the file types used in the `structure` part, as shown in Fig. 3.

The layout definition given here tells XWeb that all files declared as having the type "copy" should just be copied from the input file name and location to the output file name and location. The files of type "XHTML" are supposed to be processed by an XSL process, using a specific stylesheet given as attribute of the `xsl` element. The second attribute *navigationElement* tells XWeb to insert the navigation information created from the structural information into the input document before calling the XSLT processor. This process will be discussed in the next section.

There are other options for processing documents and creating additional layout elements. For brevity they will be only listed here, please refer to the XWeb project site for details.

- *XSLT chains:* Multiple XSLT stylesheets can be concatenated to separate different aspects of a transformation and thereby increase reuse.
- *Other process types:* SVG images can be rendered into bitmap formats and external programs can be called.
- *Rendering buttons and banners:* Images can be rendered from SVG content or using a simple XML format. Replacing text with names from the navigation allows creating buttons and banners.

4 Website Generation Process

After the input file has been parsed into a DOM document, the layout definitions are used to create processing objects and its structural part is traversed twice. During the first traversal, XWeb collects all information necessary for creating the navigation and adds it into the DOM. Most importantly this is the URL of the target location for each file. In addition to this, images for buttons and banners will be created and the information about their location added into the DOM.

During the second traversal the documents are processed. For all `entry` and `file` elements the according processes are called. They can use the information in the DOM, including the parts calculated or generated during the first traversal.

The most interesting sub-process is the `xsl` process. The XSLT processing can be done by calling a single stylesheet on the input file, with the output written to the output file given in the structural part, but there are two further options: XSLT chains and the addition of the navigation information. If multiple `xsl` tags are nested, XWeb will parse the file, pass it as input into the XSLT processor using the innermost stylesheet and then pass the output stream into an XSLT processor again, this time with the next outer stylesheet being used. The result of the outermost XSLT process is then written into the output file. In this way multiple transformations are applied without the need for temporary files, using the Simple API for XML (SAX) as streaming model.

If one of these transformation steps needs access to the navigation information collected by XWeb before, the relevant section of the website DOM can be added into the XML stream. Whenever the attribute *navigationElement* is defined on the `xsl` element, XWeb will scan the input for this XSLT

```
<section name="Docu" sourceDir="documentation" targetDir="doc">
    <entry name="DocMain" sourceFile="index.xhtml" targetFile="index.html" type="XHTML"/>
    <entry name="Installation" sourceFile="install.xhtml" targetFile="install.html" type="XHTML"/>
    <entry name="Tutorial" sourceFile="tutorial.xhtml" targetFile="tutorial.html" type="XHTML"/>
    <file sourceFile="xweb_tutorial.zip" targetFile="xweb_tutorial.zip" type="copy"/>
    <file sourceFile="xweb_tutorial_output.zip" targetFile="xweb_tutorial_output.zip" type="copy"/>
</section>
```

Figure 2: A definition of a website section.

```
<layout>
    <documentStyle type="copy">
        <copy/>
    </documentStyle>
    <documentStyle type="XHTML">
        <xsl stylesheet="stylesheets/orangeLayout.xsl" navigationElement="html"/>
    </documentStyle>
</layout>
```

Figure 3: A definition of a website layout.

transformation for the first occurance of an element with the given name. If this is found, a copy of the `structure` part, including the collected additional information will be added as a child into the XML stream. In this way the XSLT stylesheets can get access to the information they need to create the navigation. How exactly this information is used, and how the look and feel of the website will be, is left up to the stylesheet author. In combination with the buttons and banners created by the SVG image renderer this approach is very flexible.

5 Further Research and Development

The processing model used in XWeb is in many ways limited. The streaming approach taken for the XSLT processing does not extend into other areas, for a number of purposes one would like an extended processing model, based on XML and binary streams. This should include forking (e.g. splitting XHTML with SVG and MathML islands into separate streams), merging (combining the information from different content sources, or adding the navigation information) and multiplexing (multiple streams of the same type at once, where the number depends on the input, not the process setup).

Some Open Source projects and research groups have a stream based processing model, e.g. Lagoon[4][3] and Transmorpher[5][1]. Their models are more advanced than XWebs document-centric model, but they are not sophisticated enough to support the whole range of XML-based processing and the integration of the POSIX stream model. The processing model for this will be complex, but it should map into a reasonably easy programming system for processing modules, implemented by mapping named output streams of one process onto named input streams of one or more others.

Another quite promising approach grounds web sites on a knowledge base structure using RDF[6] as done in the Haystack project[2]. RDF offers the ability to interlink elements with a large number of different relations, which then can be used to create specific pages. This gives the option to create user-centric portal sites, although the classic tree-based navigation approach gives a more common structure and might therefore be easier to use.

6 Acknowledgements

XWeb itself is not much more than some glue between a set of Open Source libraries. Thanks to all people who worked on the main libraries used: Saxon (XSLT processing), Xerces (XML parsing), JDOM (Java-based DOM access) and Batik (SVG rendering). Thanks also to all people on similar projects who spend time discussing different approaches with the authors and all users giving feedback.

References

[1] Jérôme Euzenat and Laurent Tardif. XML transformation flow processing. In *Proceedings of the Extreme Markup Languages*, 2001.

[2] David Huynh, David Karger and Dennis Quan. Haystack: A platform for creating, organizing and visualizing information using rdf. Presented at the Semantic Web Workshop 2002.

[3] Mikael Ståldal. Presenting XML documents on different media with stylesheets. Technical report, KTH, Stockholm, Sweden, 2000.

[4] `http://lagoon.sourceforge.net`

[5] `http://transmorpher.inrialpes.fr/`

[6] `http://www.w3.org/`

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*

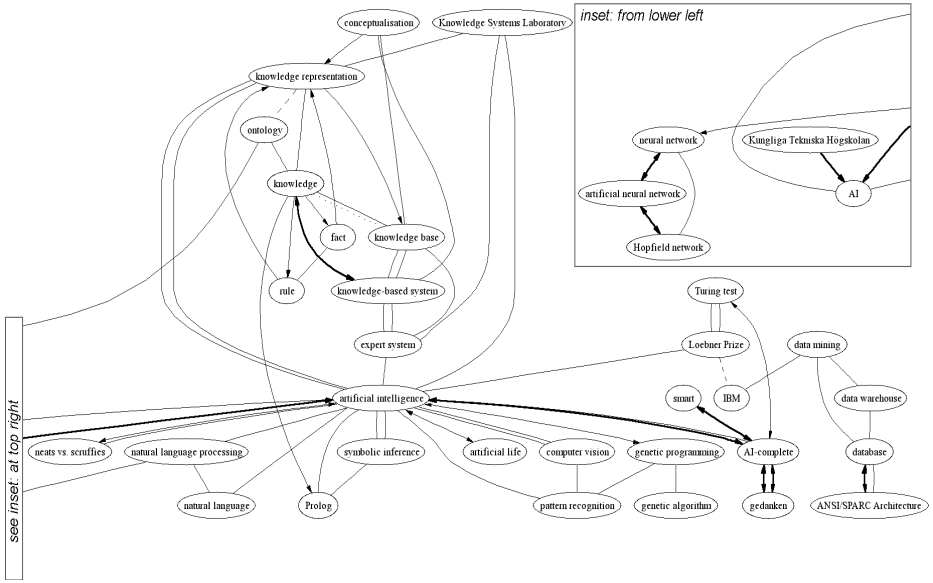


Figure 1 – Model expanded from the concepts *knowledge representation*, *genetic programming* and *data mining* with a depth of 0.8 and minimum peerage of 3.

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.

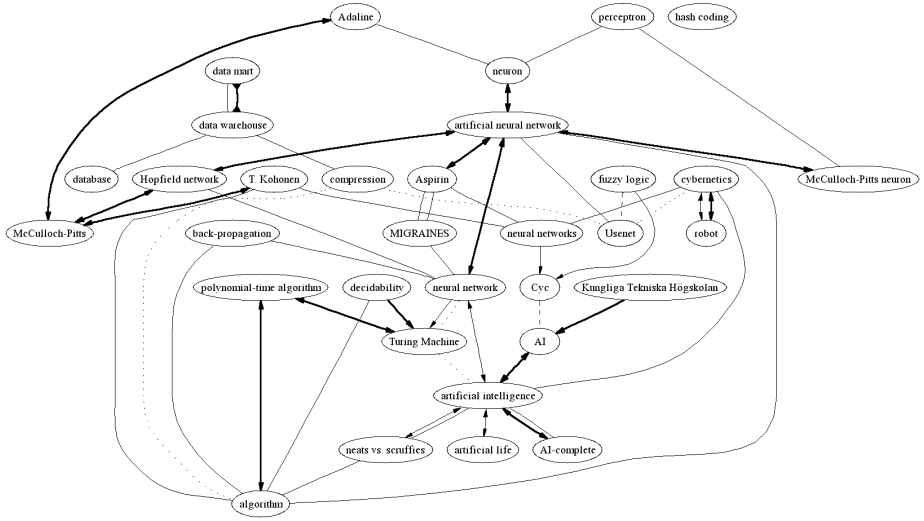


Figure 2 – Model expanded from the concepts *STRATEGY*, *neural networks* and *data marts* with a depth of 0.8 and minimum peerage of 3.

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.

Managing Literature References with Topic Maps

Robert Barta
IT School
Bond University
rho@bond.edu.au

Kate Kelly
Library Services
Bond University
kkelly@bond.edu.au

Abstract

This article introduces Topic Map (TM) authoring and ontology engineering. With a running example of a simple TM-based literature reference database we show how a localized ontology can be defined to constrain and describe our knowledge domain. We then use the same technique to describe the structure of the BibTeX *format. These two ontologies can then be used to formalize a mapping between them. For this purpose we use an experimental TM query language.*

Keywords Topic Maps, References, Ontology

1 Introduction

In the context of literature references the Topic Map standard[2] can be used to store meta data along with references to external objects. One of the objectives was also that users can have highly localized concepts but also can use globally defined ones to simplify exchange and conciliation of multiple maps.

In the following we elaborate on how TM concepts and technologies can be used to provide an open and manageable framework to define and operate with ontologies.

2 TM authoring

Topic Maps are a relatively new technology in the Semantic Web arena[13]. Their most obvious difference to RDF[12] is that they use a two-level approach[9, 10]: The more lexicographical part of a topic map consists of topics which represent (reify) real world objects but also abstract concepts. Here the main focus is on naming issues for different contexts and also URIs to objects external to the map.

The semantic aspect of TMs is covered with associations. Other than RDF statements they are not (subject, predicate, object) triples, but coerce any number of topics together, whereby each of these topics plays a specific role.

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

2.1 Topics

Using the AsTMa=[3] notation for compactness, the following TM fragment defines a topic:

Listing 1: Topic definitions

```
ltm-spec (l-specification) reifies \
   http://www.ontopia.net/.../ltm.html
bn: LTM, The Linear Topic Map Notation
bn @ latex: {LTM}, The Linear Topic \
   Map Notation
oc (cite-code) @latex: urn:bibtex:lmg01
in: This technical report defines ...
```

It carries the id `ltm-spec` and is an instance of the concept `l-specification`. It has a base name (indicated by `bn`) reifying the online document at the given URL.

Other information a topic may contain is inline data (`in`) which may contain descriptive text, and occurrences (`oc`) which hold URIs. Inline data and occurrences can be typed, as it is the case with the `cite-code` occurrence above.

2.2 Associations

Associations are statements about the relationships of various topics. The following

```
(is-author-of)
opus   : ltm-spec
author : p-lars-marius-garshol
```

would state that "the topic `p-lars-marius-garshol` plays the role of the author in an `is-author-of` association whereby `p-lars-marius-garshol` plays the role of an `opus`". All topics have been referenced via an URI and are declared separately.

3 TM Ontology Engineering

Ontologies—once defined—can be used to provide formal and informal rules how to create TM documents. In an integrated TM authoring environment ontologies can be used to guide the authoring process in the same way as XML schemas are used for XML authoring. Other uses include filtering

of TM documents according to their ontology conformance or reconciliation of heterogeneous data sources[7]. As in the following we can also use ontologies to create a *projection*, i.e. a particular view into a topic map.

3.1 Source Ontology

While there exists a general set of requirements for an ontology language[8], at the time there are only proposals for TM ontology specification languages (e.g. [11]). In the following we use AsTMa![5] which extends the authoring language AsTMa= with regular expressions, quantifiers and boolean operators.

To define an ontology $\mathcal{L}$ for the literature references we first have to set up some basic vocabulary and taxonomy:

```
(is-subclass-of)
subclass: l-book
superclass: l-document
# similar for articles, reports, ...
```

Aside from the vocabulary we have to set up rules on individual literature references. The rules can either prescribe (MUST), suggest (MAY) or forbid (MUST NOT) particular patterns found in a conforming topic map.

In our case we police that every document should have an author:

```
every [ $t (l-document)* ]
  => exists ] (is-author-of)
              opus   : $t
              author : $a [
```

First we single out all topics which are instances of `l-document`. The `*` symbolizes that these can be direct instances or instances of a subclass of `l-document`. For all those documents we prescribe the existence of an appropriate association. The][around the association pattern has to be read as *exactly so*, while the [] allow a more liberal interpretation.

Using this technique we add more rules to build the complete ontology.

3.2 Target Ontology

In this section we discuss one particular application where we use our literature ontology to filter only specific aspects out of a topic map. In this process we have to define a mapping between literature reference information in Topic Map form and a conventional database system which follows the entity-attribute paradigm.

Our database format will be BibTeX[6] which prescribes a set of document classes (books, reports, ...). Each of these classes consists of specific mandatory or optional attributes (author, title, ...).

The topic `ltm-spec` defined in listing 1 would be represented in BibTeX as follows:

```
@misc{urn:bibtex:lmg01,
      author = {{Garshol, Lars Marius}},
      title = {{LTM}, The Linear Topic
               Map Notation},
      year = {2001},
      url = {http://www.ontopia.net/...}
}
```

As such, BibTeX follows its own schema definition which serves there the role of an ontology. To allow for a formalized mapping within one single formalism, we have captured this BibTeX schema in an AsTMa! constraint by characterizing the document classes and its specific attributes.

```
b-book (class)

b-report (class)

publisher (attribute)
```

Given that, we can now make explicit the rules specific for BibTeX, such as that a book must have a title (we conveniently store that in a base name):

```
every [ $b (book) ] => exist [ $b
                               bn: * ]
```

All other attributes we plan to save via an generic `is-attribute-of` association. We only have to define that all these associations must have a particular layout:

```
every $a [ (is-attribute-of) ]
   => exists $a ] (is-attribute-of)
                  object    : *
                  attribute : *
                  value     : * [
```

A more application specific rule would be that every book also must have a `publisher` attribute while we do not care about its value:

```
every [ $b (book) ]
  => exists [ (is-attribute-of)
              object    : $b
              attribute : publisher ]
```

In a similar way we proceed with all other attributes and all other BibTeX classes.

4 Ontology Mapping

To mediate between $\mathcal{L}$ and the BibTeX ontology $\mathcal{B}$ we can hardcode the mapping directly into an application. This was actually done, not only to get working code but also to understand the practicalities involved.

4.1 Specification

In a first step the relevant topics of the literature ontology have to be identified as those which are a direct or indirect subclasses of `l-document`. For all these (`l-book`, `l-article`...) we have to define their respective counterparts in $\mathcal{B}$. While obviously a `l-book` in $\mathcal{L}$ will correspond with `b-book` in $\mathcal{B}$, for other document types in $\mathcal{L}$ this choice is less obvious.

According to $\mathcal{B}$ the class then will define which attributes are mandatory and which are optional for this object. For the book `title`, `publisher`, `year` and `author` or `editor` have to be defined, whereas the `volume` and other attributes are optional.

For all the above attributes values have to be identified in the source map. For the example specification document provided in listing 1, the application will have to follow all `is-author-of` associations for that particular document to identify the topics playing the author role there. The base names of the respective topics will be used as value for the author.

One complication is due to the fact that BibTeX citations need a *cite code* which we suggest to exist in $\mathcal{L}$:

```
every [ $t (document)* ]
  => suggested exists
       [ $t
         oc @ latex (cite-code): * ]
```

As the code is only useful in a particular context, we have added a scope `latex` to restrict its validity to that scope. We strengthen this code to be unique within one map:

```
every [ $t1
         oc (cite-code): $code ]
  => not exists
       [ $t2
         oc (cite-code): $code ]
```

This rule first singles out all topics having a cite code; they will be bound to the variable `$t1` and the corresponding value of the code is bound to `$code`. In the second clause it will then be checked whether there is a topic which contains an identical cite code. The—somewhat unorthodox—AsTMa! semantics enforces that two differently named variables cannot be bound to the same values, so the topic ids must be different.

Another issue involves BibTeX expecting a particular layout style for some attributes. Titles, for instance, have to follow a particular capitalization depending on the document class.

As it is difficult to formalize these rules in $\mathcal{B}$, we will have to burden the author of a map conforming to $\mathcal{L}$ to provide appropriate input:

```
every [ $t (document)* ]
  => suggested exists
       [ $t
         bn @ latex : * ]

every [ (is-author-of)
         author: $a ]
  => suggested exists
       [ $a
         bn @ latex : * ]
```

If the mapping application finds a LaTeX variant of the title, then that should have preference over an unscoped title. In the similar way author naming can be tailored for LaTeX.

4.2 Formalization

With these specifications above a dedicated application can now perform the mapping. One of the promises of a uniform formalism, though, is that such a mapping between two ontologies can be defined *within* that formalism. For this purpose we make use of an experimental TM query language, AsTMa?[4].

In the same way as SQL operates on tables to return a table and XQuery[1] operates on XML documents to return an XML document, AsTMa? queries analyze topic maps following the source ontology and return maps conforming to the target ontology.

As an introductory example, let us consider the conversion of books together with their titles:

```
in "literature.tm"
  where
    exists [ $b (l-book)
               bn @ latex : $t ]
  return
    {$b} (b-book)
    bn: {$t}
```

Here we iterate over the sourced topic map and look for all submaps which conform to the condition provided by the *where* clause. We are selecting all submaps which contain a `l-book` topic with a LaTeX-ready title. According to the AsTMa? language semantics only those submaps will be considered which are minimal in that they do not contain unnecessarily other topics or associations while still be conforming. In our case the submaps will only consist of the `l-book` topics. Any duplicates will then be discarded.

For all these submaps the *return* clause is evaluated. The return clause contains AsTMa= code, this time for constructing a new map. We reuse the topic id of a particular `l-book` topic also as id for topic in the target map. As all these ids are bound to the variable `$b`, they can be referred to as `{$b}`.

A more sophisticated query would capture more information about a book:

```
in "literature.tm"
  where
    exists [ $b (l-book)
             bn @ latex : $t ]
       and
    exists [ (is-author-of)
             opus   : $b
             author : $p

             $p (person)
             bn @ latex : $n ]*
  return
    {$b} (b-book)
    bn: {$t}

    (is-attribute-of)
    class     : {$b}
    attribute : author
    value     : {join(", ", $n)}
```

Again we use first a pattern to identify one topic being an `l-book`. In the second *exists* clause we now identify the association which links the author to the book `$b`. A subtle difference is the symbol `*` trailing this pattern. With this we signal *greedy matching* to the TM processor, i.e. that the matched submap should have as many instances of this pattern as possible. This results then in a list of matches. As before the processor will only pass through those minimal maps which do not violate the *where* clause (no junk).

In the construction part within the *return* clause we again refer to a single book adding the matched title as base name. In accordance with $\mathcal{B}$ we also generate an association `is-attribute-of` to add the author information which we have matched before. As we have matched multiple names the processor will have captured the individual names within a list `$n`. We concatenate these strings in the list and use the result as attribute value.

Once the target map has been built it is trivial to convert this into the final BibTeX text format.

5 Conclusion

We have demonstrated how TM engineering can be used to manage structured content and how ontologies can be used to constrain the content. The declarative nature of ontologies allows us to freely combine ontologies. Thus a document which satifies two ontologies can be said to satisfy a combination of the two.

Then we formalized a simple entity-attribute model into an ontology using generic association. This was the basis to formalize the mapping between the two ontologies which would enable a generic query processor to translate a topic map from one ontology into another.

The references for this article are mastered via the aforementioned prototype.

References

[1] XQuery, W3C Working Draft 16 august 2002. W3C.

[2] XML Topic Maps (XTM) 1.0 Specification. TopicMaps.Org, 2001.

[3] Barta, R. AsTMa= language definition, technical report. Bond University, 2001. http://www.it.bond.edu.au/publications/02TR/02-14.pdf.

[4] Barta, R. AsTMa? (Asymptotic Topic Map Notation, Querying), tutorial. Bond University, 2002. http://astma.it.bond.edu.au/astma%3F.dbk.

[5] Barta, R. AsTMa! language definition, technical report. Bond University, 2002. http://astma.it.bond.edu.au/astma!-spec.dbk.

[6] H. Kopka, P. W. Daly. A Guide to LaTeX. *Addison Wesley*, 1999.

[7] H. Wache, T. Voegele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann and S. Huebner. Ontology-based integration of information - a survey of existing approaches. *Proceedings of the IJCAI-01 Workshop: Ontologies and Information Sharing', Seattle, WA, pages 108-117.*

[8] J. Heflin, R. Volz, J. Dale. Requirements for a Web ontology language, W3C working draft 08 july 2002.

[9] Jonathan Robie. The syntactic web - syntax and semantics on the web. *XML 2001*, 2001.

[10] Lacher,-M.-S.; Decker,-S. Rdf, topic maps, and the semantic web. *Markup-Languages:-Theory-&-Practice. Summer 2001; 3(3): 313-31*, 2001.

[11] Lars M. Garshol. The Ontopia Schema Language. ISO/IEC JTC 1/SC34, Information Technology – Document Description and Processing Languages.

[12] O. Lassila and K. Swick. Resource description frame-work (RDF) model and syntax specification, technical report, W3C. 1999.

[13] T. Berners-Lee. Feature article: The semantic web. *Scientific American*, 2001.

Supporting user task based conversations via e-mail

Jyot Boparai
School of Information Technologies
University of Sydney
NSW 2006 Australia
jboparai@it.usyd.edu.au

Judy Kay
School of Information Technologies
University of Sydney
NSW 2006 Australia
judy@it.usyd.edu.au

Abstract

E-mail is commonly used for 'conversations'. These consist of a sequence of messages which deal with a common task. It would be helpful if mail clients could automatically group messages from one conversation. This would facilitate the user's processing of them as it would enable the user to establish the context of the task that is at the core of the conversation.

This paper describes IETMS, a mail client which can employ a range of approaches for this task: standard mail header elements; a TF-IDF classifier and user-lists. As a foundation for improving our understanding of the effectiveness of these mechanisms, we have performed a detailed, small-scale study involving a corpus of mail which contains a collection of conversations about an important subclass of conversations, those concerned with organising meetings. The corpus size was chosen to be comparable to the number of conversations that might run in parallel for one user who is a quite heavy e-mail user. Our study indicates the relative power of each of these as well as their combined power. It also gives insight into the value of modelling individual user's e-mail behaviours and the ways that these interact with classification mechanisms.

Keywords Document Databases, Document Workflow, Document Management, Information Retrieval

1 Introduction

E-mail is widely used for an extremely varied range of purposes. An important core of e-mail messages is the communication between people about common tasks such as organizing to meet each other, working on a joint document such as a poster or planning the details of a social event. The fundamental role of e-mail is the broad support of message-based asynchronous communication. This gives it the flexibility to support an arbitrarily diverse set of user communication. This is an important part of the reason that it has been so successful and widely adopted.

At the same time, it gives no explicit support to any one common class of tasks. This means that there are many times when a user can find it challenging to perform a task. For example, if the user receives a substantial volume of mail, it can be difficult and tedious to work systematically through all the messages related to that task. They will typically be interwoven with the mail for other sets of tasks.

Traditional e-mail clients provide two forms of support for such management of tasks: *folders* for placing related mail items together and *threading* mechanisms for recognizing mail items that may be part of an ongoing task.

In this paper we explore approaches to improving support for management of task-based e-mails by extending the power of the basic mail classification and threading. Our goal is to assist users by automatically grouping the messages belonging to a common task and by supporting the user in tracking the status of tasks.

2 Task-based e-mail conversations

In the context of e-mail, a task can be defined as an exchange of naturally chained messages [8], [1], [9], [4]. We call this a *conversation*. For example, in discussion boards, a user initially composes and sends a message about a specific topic. Other users may contribute to the discussion and the conversation continues. A conversation may involve an arbitrary number of people exchanging messages over an extended period of time. This may be just one user, where the user talks to themselves as they progress through a task. Typically, it will involve several people and considerable time may elapse between replies.

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

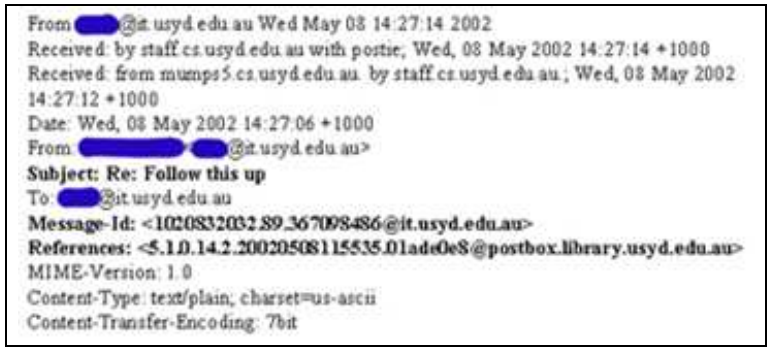


Figure 1: An example of a 'reply' message sent using a MIME-compliant e-mail client. The fields shown in bold are commonly used to thread messages in an e-mail conversation

3 Current Approaches

Current approaches make use of e-mail headers to classify e-mail messages into tasks. These headers include MIME [5, 6] headers and the headers defined in RFC2822 [7] mail standard. Another popular approach is to generate rules which can be triggered based on values of various e-mail fields (eg subject, sender, etc). When triggered, these rules automatically classify messages into folders or tasks. We now discuss these approaches and their limitations.

3.1 E-mail Headers

The mail standards RFC2822 and MIME specify headers which are recognized by most mail clients. Some are very useful for predicting which mail items are responses to an ongoing conversation. These include: *Message-Id*, *References* and *In-Reply-To*. When a person uses a MIME-compliant mail client to reply to a message, the client includes a header with the *Message-Id* of the original message in the *References* or *In-Reply-To* field of the reply message.

This is illustrated in Figure 1 which is a reply to the message with the id listed in the *References* header field. Figure 1 also illustrates the valuable role of the *Subject* line: many e-mail clients preserve the same subject in reply messages, although they may add *re:*.

3.2 Rules

Many e-mail clients allow users to define rules which automatically classify messages into folders or tasks. These rules can then be triggered, based on the values of various e-mail fields. All parts of a message may be used for this purpose including the body of a message.

Rules can be generated, automatically or manually, of the form:

if (subject *contains* "Follow this up") -> paper

to automatically classify incoming mail based on the values of subject field. In the above example, an e-mail message containing *"Follow this up"* in its subject field is classified into *paper* task.

Consider one very widely used e-mail client, Microsoft Outlook Express. It does not have any task management or task delegation support. However, it provides users with an option to group messages into threads. It also allows users to define rules which automatically classify messages based on the sender, subject and keywords present in the body of a message. Users may define these rules to automatically file incoming messages into one of the folders, where one folder represents one task. But defining these rules increases the cognitive load on the user and in some cases it may not be possible to define all the rules because of the complex nature of tasks.

Task management requires users to ensure that information relating to current tasks is grouped together. This both preserves the task context and allows users to determine the progress of an ongoing task [8], [1], [9], [4]. The threading ability of Outlook helps users in managing tasks to some extent as all the information is present in a single conversational thread.

When the option to group messages into threads is selected, the threading algorithm of Outlook Express groups messages with same subjects (after removing *'re:'* and *'fw:'*). It then uses *Message-Id* to thread the grouped messages.

3.3 Limitations

The strategies used by existing systems have some serious shortcomings for automated classification of e-mail messages into conversations. We now outline several forms of this.

The users may use the reply facility of a client to save typing an e-mail address even when the user is not actually replying to that message. If

a MIME-compliant e-mail client is used by a user, using reply feature will preserve the *Message-Id* in the reply message. As a consequence the message may be incorrectly classified.

Users tend to have conversations about multiple tasks in a single e-mail message to avoid sending multiple messages – a separate one for each topic. This introduces problems for a task management system if it classifies the message under that one task where it should, correctly, have classified it under each of the tasks discussed in the message.

Another related, but different, problem arises from drifts in conversations. Initially, the conversation may start discussing a topic but as the conversation evolves, the topic drifts. The task management system, assuming these messages belong to the same conversation, classifies them into a common task. This increases the cognitive load on the user because the information relating to one task is not readily available – they would have to go through all such messages to find the required information.

Yet another way that users can subvert task classification is by composing a fresh message as a reply to an ongoing conversation. Then the header fields (MIME and RFC2822) are not preserved. Approaches which rely on these fields to classify messages into tasks will fail when these fields are missing.

A final problem follows where the user's e-mail client is not MIME-compliant. In this case, the reply mail will not preserve MIME headers when a user chooses to reply to a message. Some of the clients add *'re:'* to the subject field of a reply message. In these cases, threading based on *Message-Id* will fail.

4 IETMS

Intelligent E-mail and Task Management System (IETMS) builds upon iems [2, 3], the intelligent e-mail sorter. IETMS supports task delegation and task management including mechanisms for defining tasks and the ability to automatically classify messages into one of these tasks. A meeting support wizard is also included to help users initiate meetings. We now present a detailed explanation of these features.

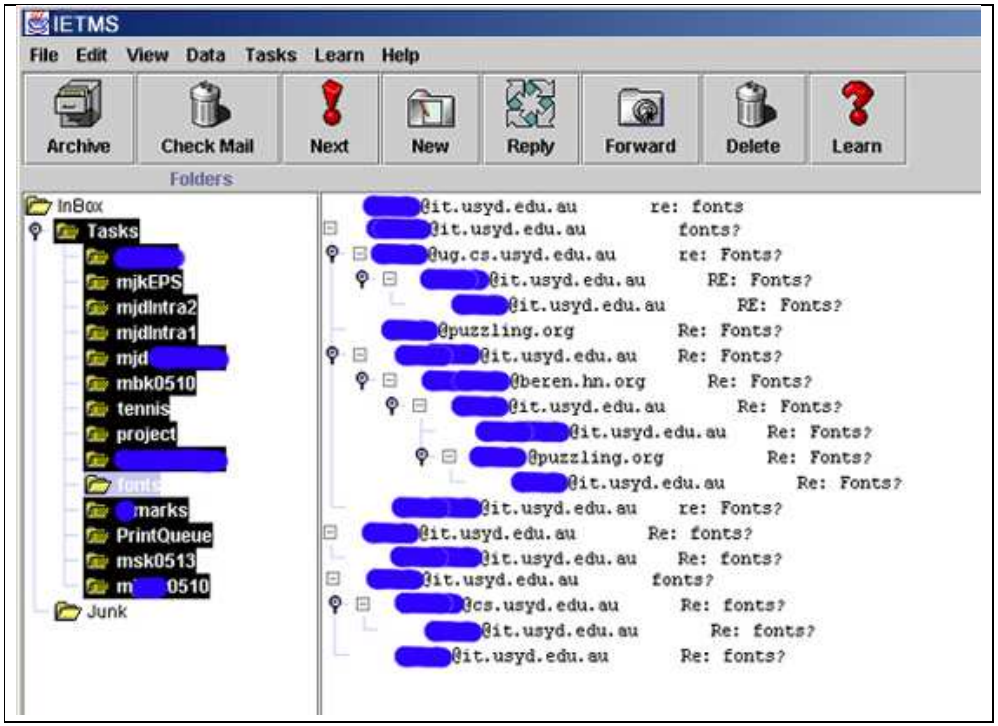


Figure 2: Tasks and Conversation threads in IETMS. Parts of the screen-shot are hidden for privacy

The interface of IETMS is similar to many mail clients, with a window for the currently selected folder, one for the currently selected mail item in that folder and one for the folder names. Unlike other clients, it separates the notion of a task from the notion of a folder. In the interface shown in Figure 2, tasks appear similar to folders (left frame) but they are colour coded to help users differentiate between a folder and a task. The interface also shows the e-mail messages belonging to *"fonts"* task grouped into threads (right frame). The reply messages are indented to help users keep track of conversations as all the conversational e-mail messages are groupped together.

An important element of the iems interface is that it can use automatically induced rules to pre-sort the inbox. IETMS extends this by distinguishing which incoming e-mail messages are part of a conversation. Then the inbox sorts the e-mails within it, placing mail predicted to be part of each conversation together and other mail is still pre-sorted into the predicted folder.

To predict whether a message belongs to a particular conversation, IETMS collects various sources of evidence. It uses the list of people involved in earlier parts of the conversation as one of the sources of evidence. The more common the list of people on two messages, the greater the likelihood they are part of the same message. When a task is initiated, a list of people involved in that task is stored. It is then used to quickly determine if an e-mail message is from one of the people involved in a task. However, the scope of this method is quite limited as people may be involved in many concurrent tasks.

As described in Section 3.1, *Message-Id*, *References* and *In-Reply-To* MIME headers provide a valuable method for classifying e-mail messages into conversations. To cater for inconsistencies in e-mail clients, the values of these fields are pre-processed before matching. Special characters (@, <, > and .) are removed because some clients insert < and > around the values of these fields while others do not. After pre-processing, the value of *References* or *In-Reply-To* fields, of a new message, is matched against the value of *Message-Id* field of task-related messages. If a match is found, the incoming message is classified into the same task as the matched message.

A related, but different, method is to classify incoming messages using subject field. The subject field of an incoming message is matched against the subject fields of task-related messages. For same reasons as MIME headers, the value of subject fields needs to be pre-processed before matching– *'Re:'* is removed along with non-alphabetical characters. When a match is found, the incoming message is classified into the same task as the matched message.

Finally, a TF-IDF text classification algorithm is used to analyze the body of the message as an additional source of evidence. This method classifies an incoming message by comparing the frequency of terms which are common in the message and the tasks but are rare otherwise. A stop list is used to remove common words (a, am, the, etc) from the body as these words do not give us any insight into the content of the message. This method may be rather limited in general since message bodies will tend to be short. However, as one of a range of evidence sources, it has some potential power.

5 Supporting conversations about Meetings

To evaluate approaches to classifying messages into conversation we decided to focus on one class of tasks. Studies of tasks have showed that *'scheduling a meeting'* is one of the key tasks accomplished using e-mail in organizations. From different types of user tasks explored by Takkinen [8] it is certainly the most interesting and the most complex.

The IETMS interface provides a specialized interface wizard for the meeting scheduling task (Figure 3). This ensures that the meeting initiator considers the standard elements of a meeting: the time, duration, place, people to invite, information to be provided to them, such as a description of the purpose of the meeting.

6 Evaluation Experiments

The success of IETMS was measured in its ability to classify e-mail messages into tasks and folders and its ability to group task related messages into threads. It involved measuring the accuracy (precision and recall) of various classification and threading algorithms. It also involved doing a user-by-user analysis to determine the significance of limitations (described in section 3.3) on classification algorithms. We now present the experimental setup and results of these experiments.

Figure 3: A meeting wizard to help user initiate a meeting

Task Number	Number of Messages	Number of Users	Initiated	Completed
1	84	8	20 Feb 2002	24 Oct 2002
2	30	4	4 Apr 2002	10 Apr 2002
3	6	2	19 Mar 2002	19 Mar 2002
4	34	5	4 May 2002	6 May 2002
5	18	5	8 May 2002	3 Jul 2002
6	6	3	8 May 2002	10 May 2002
7	3	3	10 May 2002	10 May 2002
8	5	3	10 May 2002	13 May 2002
9	27	4	10 May 2002	18 Jun 2002
10	16	6	11 May 2002	20 May 2002
11	18	12	22 May 2002	20 Jun 2002
12	37	7	14 Aug 2002	29 Aug 2002
13	25	11	22 Sep 2002	24 Sep 2002
14	13	5	24 Sep 2002	28 Sep 2002
Total:	322	78		

Table 1: Details of the tasks in testing data

6.1 Experimental Design

We were able to make use of a small collection of e-mail. These all involved the task of organizing meetings within a tertiary institution. The messages were part of a study and protocol analysis of the way that a smart personal assistant might manage the organisation of meetings. Accordingly, a person took the role of organising a series of

meetings on behalf of other individuals who needed meetings organised: mailing the participants, collating replies, reviewing plans and going back to the meeting sponsor. All individuals involved were informed that their mail would be studied in order to improve our understanding of how to build a smart personal assistant for organising meetings. The collection contained 322 task-related e-mail messages involving 50 users. The messages were classified into 14 tasks. The smallest task contained only 3 messages where the largest task contained 84 messages (over 25% of total messages). Table 1 summarizes each of the tasks.

Some conclusions could be drawn about the nature of task related e-mails by analyzing the data given in Table 1. Firstly, the table illustrates that some tasks overlapped each other. There were a number of ongoing tasks during May 2002 to Jul 2002 period. Interestingly, Task 1 overlapped all the other tasks. Secondly, the length of tasks from initiation to completion varied significantly. The length of tasks 4, 6 and 12 was just 2 days where the length of Task 1 was over 8 months. Thirdly, some tasks did not have a great deal of communication between participants. In task 7 all three e-mail messages were sent by three different participants, where as, in task 1, over 80 messages were exchanged between eight participants.

6.2 Results

A 'Sliding Window' tester was implemented with varying training window size and a fixed testing window of size 1. Given the nature of conversational e-mail messages, list of e-mail messages [1..N] is chronologically ordered. So we chose $j < N$ and trained on the set [1..j] in order to be able to test on the (j+1)th message. This process was repeated as the size of training set was increased by 1 and the test window was shifted.

In Figure 4, accuracy of a MIME-header based classification algorithm is presented. The average precision of this algorithm was very high (94.39%) as it classified a message if and only if its *References* or *In-Reply-To* fields referred to *Message-Id* field of one of the pre-classified messages. Only 3 messages were ever incorrectly classified (1 in task 3 and 2 in task 14). These features were due to users employing a single e-mail message to respond to multiple tasks.

On the other hand, average recall was very low (only 35.23%) as it missed a lot of messages due of limitations described in section 3.3. For example, two of the participants (users 2 and 8) in Tasks 1 and 4 used a non-MIME compliant e-mail client. As a result the recall for these tasks was about 6% because the mail client did not copy the *Message-Id* of the original message in *References* or *In-Reply-To* fields of the reply message.

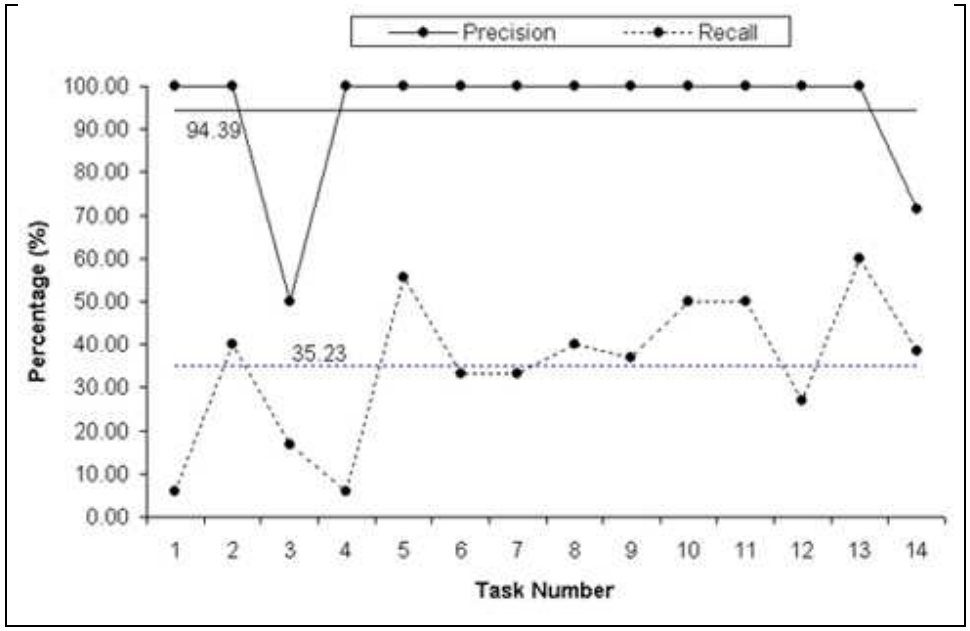


Figure 4: Accuracy of Message-Id based Classifier

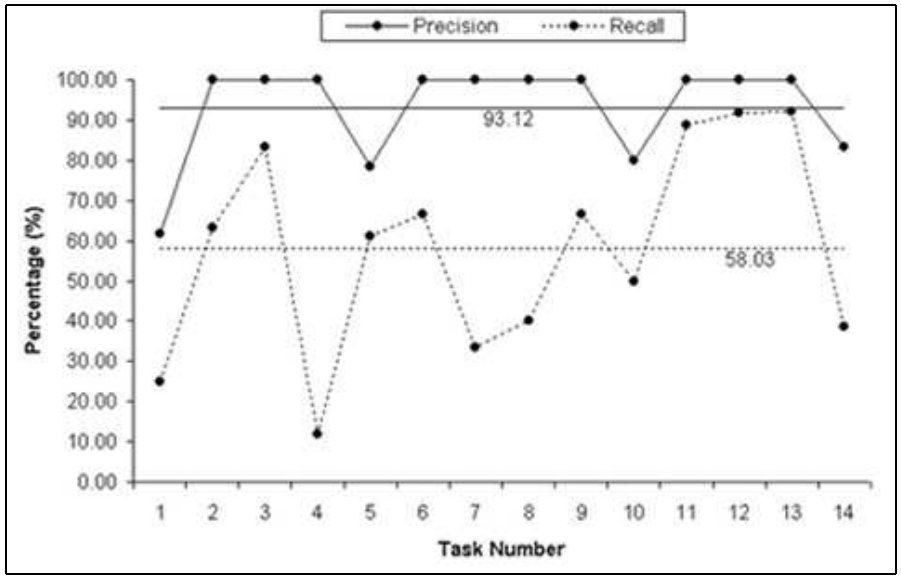


Figure 5: Accuracy of Subject based Classifier

Figure 5 presents the accuracy of a Subject based algorithm. The average precision of this algorithm was high (93.12%) as it classified a message if its subject, after removing *'re:'*, matches the subject of a pre-classified message. In tasks 5 and 10 a total of 5 messages were incorrectly classified; task 5 was a meeting about 'School's Intranet site' and task 10 was a follow-up meeting on the same topic, so both tasks had messages with common subjects. Messages in task 1 had some generalized subjects (eg *'meeting'*) and messages belonging to other categories with same subject were incorrectly classified in task 1.

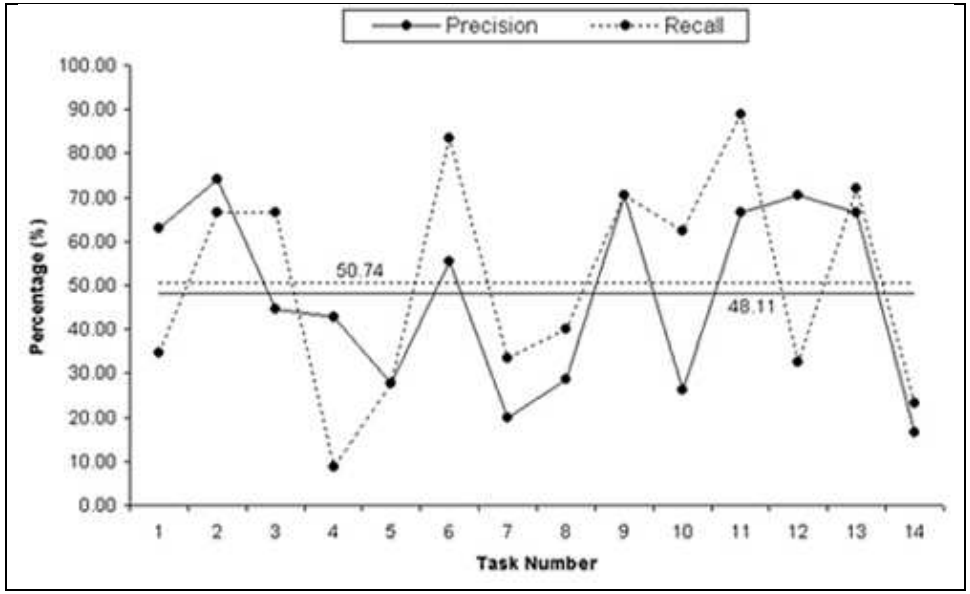


Figure 6: Accuracy of TF-IDF based Classifier

The average recall of 58.03% for Subject based algorithm was an improvement on MIME-header based algorithm, but it still missed a few messages. The reason for this was that users manually edited the subject line of an e-mail message before sending it. In some cases, users sent a fresh message rather than using the reply feature of the e-mail client–the subject line was changed when typing the fresh message. The recall for task 4 was very low (only 11.76%) because messages were one-way interactions where the initiator sent e-mail messages to inform other participants about various facts about the meeting. Task 1 involved many interactions consisting of only a few messages (only 2 or 3 messages). All these interactions had different values for subject fields and thus the Subject-based algorithm was not able to classify them correctly.

The accuracy figures of a TF-IDF based algorithm, which analyzed the body of messages, are presented in Figure 6. The average precision of this algorithm was quite low (48.11%). The biggest problem was that some of the tasks shared common terms which caused this algorithm to classify messages incorrectly. As stated above, tasks 5 and 10 shared a common subject which meant that messages belonging to these tasks contained similar terms (eg 'intranet') which were rare among other tasks. As a result, the precision of about 26% was achieved in both cases. Similar problems also caused the precision of tasks 7 and 14 to be quite low (about 18%).

The TF-IDF based algorithm worked best for tasks 11 and 13. In both cases, the e-mail messages contained distinct terms which were not common in any of the other tasks ('ADCS' was the dominating term for task 11 and 'tennis' for task 13). The body of a message may contain content relevant to multiple tasks as people may employ a single e-mail message to reply to multiple messages. Surprisingly, using TF-IDF algorithm the recall for task 1 was highest as compared with recall for the same task using other algorithms proving that this method had some potential power in classifying task-related messages.

MIME-header and Subject based algorithms classified messages with high precision but they tend to have a fairly low recall values. On the other hand, the TF-IDF based algorithm classified messages with moderate precision and recall values but it improved the accuracy for some tasks (eg tasks 1 and 6). In order to combine the strengths of these three algorithms a hybrid algorithm was implemented. This hybrid algorithm, firstly, attempted to classify messages using MIME-header and Subject based algorithms. If a suitable task was not found, it then used the TF-IDF algorithm. The accuracy of this algorithm is presented in Figure 7. The accuracy figures for some tasks, which showed poor precision and recall values (eg tasks 4) with other algorithms, improved dramatically.

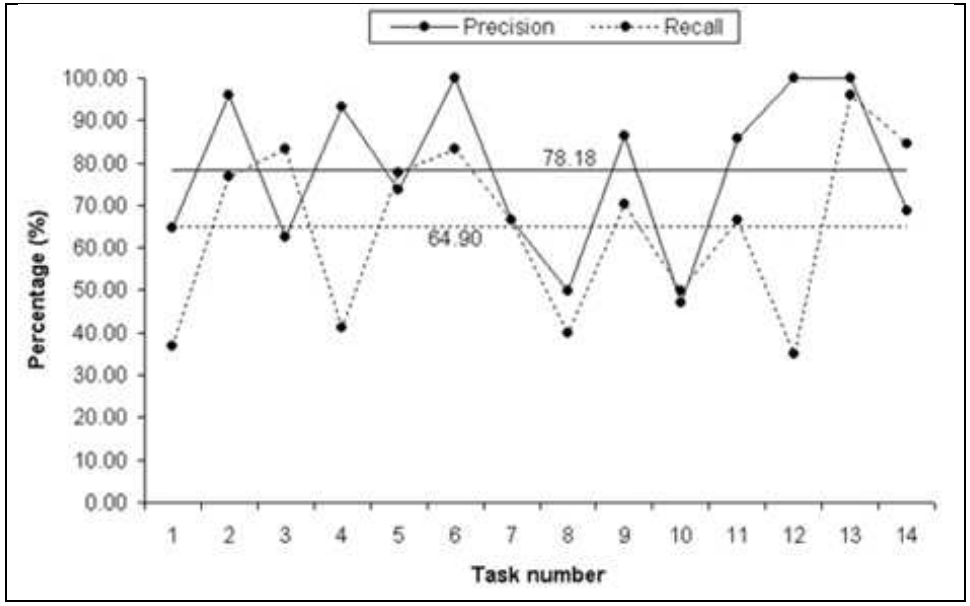


Figure 7: Accuracy of Hybrid Classifier

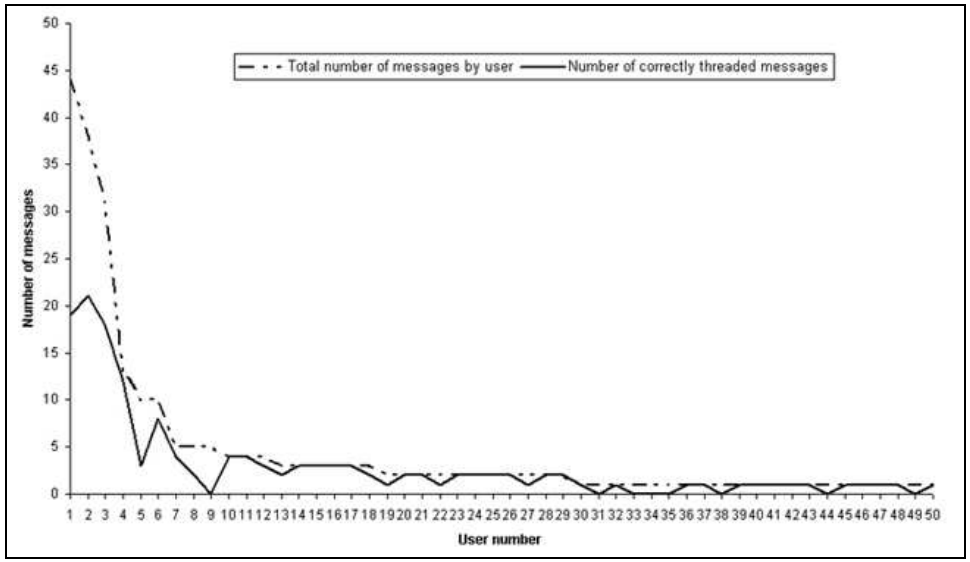


Figure 8: Number of messages sent by a user and number of messages correctly threaded

The results of a user-by-user analysis, using the Hybrid algorithm, are presented in Figure 8. The purpose of this analysis was to explore the potential value of modelling the e-mail addreses which were poorly handled by the Hybrid algorithm. In a deployed system it would be useful to be able to flag messages that the system believed it had poor classification powers. The main problems for the hybrid algorithm follow from user behaviour and the user's choice of mail client. This suggests that marked number of individual users would be responsible for the problems in use of the hybrid algorithm.

User 1 initiated all the tasks and as a result many messages were incorrectly classified for them. Interestingly, users 2 and 8 used non-MIME compliant e-mail clients which meant that MIME-headers could not be used to classify the message sent by them. However, the accuracy figures of 55% and 40% for users 2 and 8 respectively suggested that other parts of the message (subject or body) were able to classify them correctly some of the time. Hence, some of the limitations described in Section 3.3 were oercome by the hybrid algorithm.

The same 'Sliding window' tester was then used with a constant sized training window which was moved through the list of chronologically ordered messages. Figure 9 presents the accuracy figures of the hybrid algorithm with a training window of size 19. In other words, in first window messages numbered [1..19] were used for training and message numbered 20 was used for testing. This process was repeated for all the windows. The average precision of 87.50% and average recall of 91.67% were achieved. But as shown in the figure, precision and recall of most windows were 100% highlighting the fact that users participated in tasks for a short period of time and typical tasks only involved a moderate number of messages (19).

7 Conclusion and Discussion

The goal of this paper was to explore the support for task management using e-mail. This involved using rule-based and machine learning algorithms to automatically classify and prioritize messages into tasks and folders.

A thorough evaluation of these techniques was carried out using a corpus of mail which contained a collection of conversations about meetings. The results of this evaluation showed that while keeping the attractive properties of e-mail, including:

- easy of use,
- wide acceptance, and
- asynchronous nature

new and improved uses of e-mail, *e.g. task management*, could be encouraged by developing specialized applications on top of, otherwise, unstructured e-mail domain.

We also improved our understanding of the prevalence of user behaviours that thwart such applications. The value of modelling individual user's e-mail behaviours was also explored, as was the value of adapting classification mechanisms to changes in these behaviours.

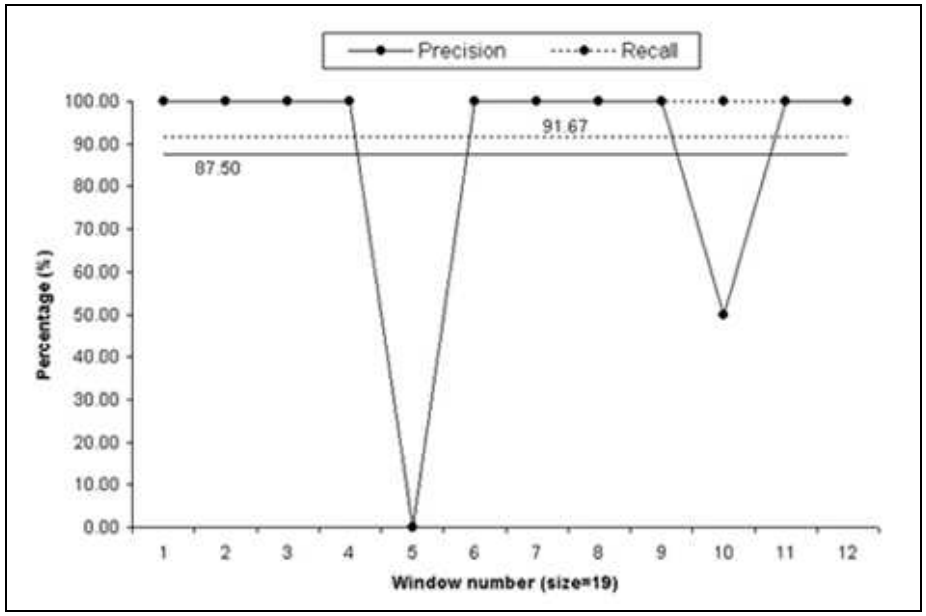


Figure 9: Accuracy for a constant sized window

8 Acknowledgement

We would like to acknowledge Sarah Kummerfeld, Josiah Poon and Kalina Yacef for allowing us to use the data set, compiled for a separate project.

References

[1] D. Comer and L. Peterson. Conversation-based mail. *ACM*, Volume 4, Number 4, pages 299–319, November 1989.

[2] Kay J. Crawford E. and McCreath E. Automatic induction of rules for e-mail classification. In *Proceedings of the Sixth Australiasian Document Computing Symposium*, December 2001.

[3] Kay J. Crawford E. and McCreath E. An intelligent interface for sorting electronic mail. *A short paper in IUI'02*, January 2002.

[4] S. Fleming and A. Kilgour. Electronic mail: Case study in task-oriented restructuring of application domain. In *Proceeding of IEEE: Computers and Digital Techniques*, number 2, pages 65–71, March 1994.

[5] Freed N. and Borenstein N. Rfc2046 multipurpose internet mail extensions (mime) part one: Format of internet message bodies, November 1996.

[6] Freed N. and Borenstein N. Rfc2046 multipurpose internet mail extensions (mime) part two: Media types, November 1996.

[7] Resnick P. Rfc2822 internet message format, April 2001.

[8] J. Takkinen. *From Information Management to Task Management in Electronic Mail*. Ph.D. thesis, Department of Computer and Information Science, Linkopings universitet, SE-581 83 Linkoping, Sweden, 2002.

[9] J. Takkinen and N. Shahmehri. Coordination in message-based environments: Restructuring e-mail to accomplish tasks. In *Proceedings of the CTIS98, the International Workshop on Coordination Technologies for Information Systems*, pages 566–571, August 1998.

Automatic Categorization of Announcements on the Australian Stock Exchange

Rafael A. Calvo
Web Engineering Group
The University of Sydney
Bldg J03, Sydney NSW 2006
rafa@ee.usyd.edu.au

Ken Williams
Web Engineering Group
The University of Sydney
Bldg J03, Sydney NSW 2006
kenw@ee.usyd.edu.au

Abstract

This paper compares the performance of several machine learning algorithms for the automatic categorization of corporate announcements in the Australian Stock Exchange (ASX) Signal G data stream. The article also describes some of the applications that the categorization of corporate announcements may enable. We have performed tests on two categorization tasks: market sensitivity, which indicates whether an announcement will have an impact on the market, and report type, which classifies each announcement into one of the report categories defined by the ASX. We have tried Neural Networks, a Naïve Bayes classifier, and Support Vector Machines and achieved good results.

Keywords Document Management, Document Workflow

1 Introduction

The Australian Stock Exchange Limited (ASX - http://www.asx.com.au/) operates Australia's primary national stock exchange. Companies listed on ASX are required under the Listing Rules to make announcements about their activities "in order to ensure a fully informed market is maintained." [1] In order to guarantee access to this information, stock exchanges such as the ASX publish all recent and historical company announcements. Thanks to language technologies such as automatic document categorization, these corporate announcements can provide new sources of valuable financial information.

Historically, corporate announcements have provided valuable information to traders and the general public for decades. For this reason these announcements are used by regulators as the main tool to keep the market informed of all important events. The law assumes that these public announcements contain all the information needed by an individual trader to keep a reasonable understanding of what is happening with a particular company. This allows investors to make decisions based on information that is up to date and is equivalent to the information that company insiders might have. There is little doubt about the value of the information contained in these announcements, and several research groups are developing novel applications using this data. We describe in this article the evaluation of categorization techniques used to build these applications.

The ASX Data Services is a financial information service providing daily market information from the Stock Exchange Automated Trading System (SEATS), ASX futures and the company announcement service. All daily stock exchange activity is available in different electronic data feeds that the ASX calls "signals." In our work, we have used announcements from the ASX's Signal G, which provides subscribers with company announcements issued by companies or the ASX in accordance with listing rules.

Section 2 of this paper describes the Signal G data set. Section 3 describes the different machine learning techniques and the categorization framework that we have used to perform these types of categorization tasks. Section 4 describes the quantitative results and section 5 concludes.

2 Data Description

In this paper we assess performance on two tasks: "report type" and "market sensitivity" categorization. Report type is "a code to categorize company announcements" [1] and may take one of 144 values like "annual report" or "takeover announcement." Market sensitivity is a boolean category indicating whether an announcement contains information that may influence trading in the issuing company. This allows users of this data to select which announcements are critical.

Currently, both kinds of category assignments on the Signal G documents are made manually by the ASX. Table 1 shows the distribution of docu-

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

	Sensitive	Nonsensitive	Total
Training	32458	63067	95525
Test	14072	27033	41405
Total	46530	90100	136630

Table 1: Sensitivity category distribution in Signal G

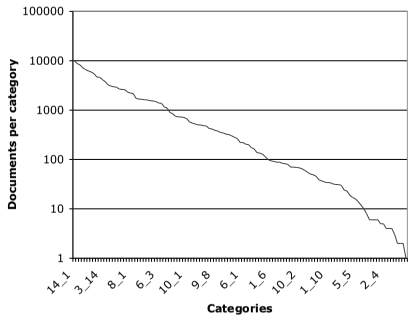


Figure 1: Report type category distribution in Signal G

ments with respect to the sensitivity category, as well as our split between training and testing documents. Figure 1 shows the report type category distribution.

Note that the report type labels are highly skewed across categories. The most common category contains 10,064 documents, while ten categories have fewer than five documents each.

Signal G is available to the public, member organizations, and information vendors. Our data set was supplied by the Capital Markets Collaborative Research Centre. For future groups who receive permission from the CMCRC to work with this data, we have converted it into an XML format.

3 Methods

We have compared three machine learning methods that have provided some of the best performances in other classification tasks [16]: Support Vector Machines (SVM) [12] [6], Naïve Bayes (NB) [7], and Neural Networks (NN) [5] [4]. It is not possible to describe them thoroughly in this article, so we will only summarize those issues that might be required to reproduce the results. Our Naïve Bayes and SVM implementations are described in [15].

Because of the inability of our SVM implementation to handle the size of our training set, we had to only train on 8000 documents at a time in order to finish the experiments in a reasonable amount of time. The SVM was not able to perform the report type task, only the sensitivity task. Future work may dramatically improve training speed by using recently developed training optimizations in the literature, enabling us to train on a larger number of documents and presumably to improve our correctness as a result.

Our Neural Network architecture [5] [4] used a backpropagation algorithm that minimizes quadratic error. The input layer has as many units (neurons) as document features retained. The number of hidden units is determined by optimising the performance on a cross-validation set. There is one output unit for each possible category, with activation values between 0 and 1. For each document, the classifier will assign any category whose output unit is greater than 0.5. In our experiment, we used 3-fold cross-validation and averaged the weights of the three resultant neural networks.

Each categorizer was trained on the 95,525 training documents for each task, report type and sensitivity. For the Neural Network categorizer, 13,711 training documents were set aside from the training set to function as a validation set when tuning the weights of the network. The trained categorizers were then evaluated on the 41,405 test documents. Since documents in the test set have not been used to adjust the parameters of the classifier, it is normally assumed that the performance on new data would be similar.

Table 2 summarizes the general steps followed in the preparation of the experiments. TF/IDF weights are given in the notation followed by [11]. Our experimental process was as follows:

1. Linguistic dimensionality reduction: A list of stopwords [10] was removed from the document collection and the Porter stemming algorithm was applied [8].

2. Statistical dimensionality reduction: Chi Squared or Document Frequency criteria were employed to reduce the feature vector dimensionality [8] [13].

3. Vectorization and weighting: The resulting documents were represented as vectors, using TF/IDF weighting [11] [17].

4. Architecture: The selected terms were used as input features to the classifier. Some of the algorithms allow several architectures, and the best algorithm was chosen by optimising the results on a cross-validation set.

5. Training: We generated a cross-validation set randomly. These documents were set aside and the Neural Network was trained on the remaining ones.

4 Results

In evaluating our classifiers on our data set, we use the common statistical measures *precision*, *re-*

	Stopwords	Stemming	Feature Reduction	TF/IDF	Architecture
NN	SMART	Porter	χ^2	tfc	1000 features, 50 hidden units
NB	SMART	Porter	DF	tfx	1000 features
SVM	SMART	Porter	DF	tfx	1000 features, linear kernel

Table 2: Comparative description of algorithms used

	Micro			Macro		
	p	r	F_1	p	r	F_1
NN	0.89	0.89	0.89	0.88	0.88	0.88
NB	0.83	0.84	0.83	0.90	0.90	0.90
SVM	0.82	0.82	0.82	0.80	0.79	0.80

Table 3: Performance for the market sensitivity task

	Micro			Macro		
	p	r	F_1	p	r	F_1
NN	0.87	0.71	0.78	0.45	0.34	0.37
NB	0.62	0.67	0.64	0.46	0.61	0.46

Table 4: Performance for the report type task

call, and F_1. [16] [14] When dealing with multiple classes there are two possible ways of averaging these measures, *macro-averaging* and *micro-averaging*. The macro-average weights equally all the classes, regardless of how many documents they contain. The micro-average weights equally all the documents, thus biasing toward the performance on common classes. Since different learning algorithms will perform differently on common and rare categories, both micro-averaged and macro-averaged scores are typically reported to evaluate performance.

It is important to note that the performance results are based on comparing the automatic categorization of each document with the tagging of human experts at the ASX. The manual classification is a subjective decision process affected by the ASX's legal liabilities and the normal human classification disagreements. It has been shown in various studies that there could be considerable variation in the inter-indexer agreement [3] [2]. For example, in a Reuters news collection correction rates averaged 5.16% [9] with some editors being corrected up to 77% of the time. Similar disagreements can be expected in the ASX's assignments on the Signal G corpus. In the light of these disagreements we can imagine that there might be a limit to the performance that can be obtained by automatic categorization.

Figure 2 shows a histogram of classes and documents for different performance ranges using the Naïve Bayes classifier. It shows how most categories that do well have large number of documents, except for some that have very few documents (fewer than 5). This shows the well-known result that the machine learning algorithms such as Naïve Bayes perform better on well-populated categories. Similar results can be obtained for the other classifiers.

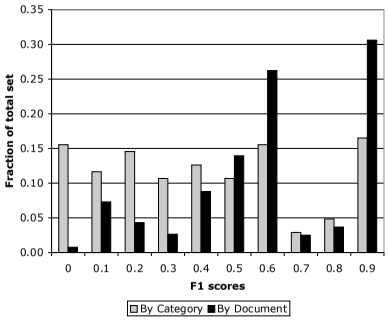


Figure 2: Histogram of report type results for different performance ranges

5 Conclusion

In this paper we have applied several machine learning techniques (Neural Networks, Naïve Bayes, and Support Vector Machines) to the categorization of announcements of companies publicly traded by the ASX. Two tasks were evaluated: the categorization of documents as sensitive or not, and the categorization in one of the 144 report types defined by ASX. The results show that is possible to obtain classifiers with more than 88% precision and recall on the sensitivity task and 86% precision, 74% recall on the report type task.

The results are somewhat better than the ones obtained by several researchers working on the Reuters news cable database [5] [4]. This database has fewer categories (90) than the report type task but also fewer documents (10,000). Although it is risky to try to extrapolate the results, we believe that due to the similarity in the documents, other financial databases with documents in English should also have similar performance. Future work includes testing adding statistical feature selection to the classification framework, and improving the efficiency of the algorithms so they can be used for even larger data sets.

The excellent performance shows the possibility to use these classifiers in commercial applications

for both tasks, sensitivity detection and report type categorization.

Acknowledgements

The authors gratefully acknowledge financial support from the Capital Markets Collaborative Research Centre and the University of Sydney.

References

[1] Australian Stock Exchange web site. http://www.asx.com.au/, 2002.

[2] Thorsten Brants. Inter-annotator agreement for a german newspaper corpus. In *2nd International Conference on Language Resources & Evaluation*, 2000.

[3] R. Bruce and J. Wiebe. Word sense distinguishability and inter-coder agreement. In *3rd Conference on Empirical Methods in Natural Language Processing (EMNLP-98)*, Granada, Spain, June 1998. Association for Computational Linguistics SIGDAT.

[4] Rafael A. Calvo. Classifying financial news with neural networks. In *6th Australasian Document Symposium*, page 6, December 2001.

[5] Rafael A. Calvo and H. A. Ceccatto. Intelligent document classification. *Intelligent Data Analysis*, Volume 4, Number 5, 2000.

[6] Thorsten Joachims. Transductive inference for text classification using support vector machines. In Ivan Bratko and Saso Dzeroski (editors), *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pages 200–209, Bled, SL, 1999. Morgan Kaufmann Publishers, San Francisco, US.

[7] David D. Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. In Claire Nédellec and Céline Rouveirol (editors), *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 4–15, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

[8] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999.

[9] Tony G. Rose, Mark Stevenson and Miles Whitehead. The reuters corpus volume 1 - from yesterday's news to tomorrow's language resources. In *3rd International Conference on Language Resources and Evaluation*, page 7, May 2002.

[10] Gerard Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, Reading, Pennsylvania, 1989.

[11] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management: an International Journal*, Volume 24, Number 5, pages 513–523, 1988.

[12] Bernhard Schölkopf, Christopher J.C. Burges and Alexander J. Smola (editors). *Advances in Kernel Methods – Support Vector Learning*. MIT Press, 1999.

[13] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys (CSUR)*, Volume 34, Number 1, pages 1–47, 2002.

[14] C. J. Van Rijsbergen. *Information Retrieval, 2nd edition*. Butterworths, 2 edition, 1979.

[15] Ken Williams and Rafael A. Calvo. A framework for text categorization. *7th Australasian Document Computing Symposium*, 2002.

[16] Y. Yang and X. Liu. A re-examination of text categorization methods. In *22nd Annual International SIGIR*, pages 42–49, Berkley, August 1999.

[17] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In Douglas H. Fisher (editor), *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 412–420, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.

XML Document Retrieval with PADRE

Nick Craswell[1] *David Hawking*[1] *Alexander Krumpholz*[2] *Ian Mathieson*[2]

James A. Thom[3] *Anne-Marie Vercoustre*[2] *Peter Wilkins*[2]

Mingfang Wu[2]

[1]CSIRO Mathematical and Information Sciences
GPO Box 664, Canberra, ACT 2601, Australia

[2]CSIRO Mathematical and Information Sciences
Private Bag 10, South Clayton MDC, VIC 3169, Australia

[3]School of Computer Science and Information Technology, RMIT University
GPO Box 2476V, Melbourne 3001, Australia

Email for correspondence: *Anne-Marie.Vercoustre@csiro.au*

Abstract

One paradigm of XML retrieval is database-style querying of semi-structured data. Another paradigm is based on information retrieval involving ranking of documents or document fragments. The INEX project attempts to integrate these paradigms and provide an environment for conducting retrieval experiments on semi-structured data. This paper discusses our participation in the INEX project and what we discovered about combining these paradigms.

Keywords Document Databases, Document Standards, Information Retrieval

1 Introduction

Managing semi-structured data, such as collections of XML documents, involves the use of both database and text retrieval technology. As the volume and complexity of semi-structured data available to users increases better retrieval mechanisms are required to support users' increasingly sophisticated information needs.

XML retrieval falls into two broad approaches: database-style querying of XML semi-structured data (supported by query languages such as XQuery) and text retrieval of marked-up documents and portions of documents using text-based search engines. In a recent survey of indexing and searching XML Documents [7] several approaches for combining structural information with ranking are described. The two communities have also had a very different approach to evaluation. The database approach is interested in exact matches to the query and concentrates more on the power of expression of the query language and efficient query interpretation. The information retrieval approach is focusing on effective ranking of the answers through precision/recall measures, in relation to user evaluation of answer quality. Some retrieval languages such as XIRQL [4] attempt to bridge the divide between the database and information retrieval approaches, however there has been no easy way to evaluate such languages until now. The experiment devised for INEX (Initiative for Evaluation of XML retrieval) [3] attempts (with only partial success) to integrate these two very different approaches. The aim of the INEX initiative is to provide the means, in the form of a large testbed (test collection and queries) and appropriate scoring methods for the evaluation of XML documents retrieval. The queries include two different sets of queries: content-based queries and content-and-structure-based queries. In both cases, the scoring method takes into account the granularity of the answer.

This paper reports our participation in the INEX project. In Section 2 we introduce the INEX project. In Section 3 we present our approach and techniques for addressing the INEX retrieval tasks. We analyse and discuss some current limitations of our approach in Section 4. Section 5 presents some related work.

2 INEX - XML document search

The INEX [3] *Initiative for the Evaluation of XML retrieval* is organized by the European DELOS Network of Excellence for Digital Libraries. INEX

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

involves nearly 50 groups from around the world undertaking a series of retrieval tasks (topics) on a collection of XML documents (XML sources for about 12,000 articles in IEEE Computer Society publications since 1995).

```
<?xml version="1.0"
    encoding="ISO-8859-1"?>
<!DOCTYPE INEX-Topic SYSTEM
    "inex-topics.dtd">
<INEX-Topic topic-id="17"
    query-type="CAS"
    ct-no="088">
  <Title>
    <te>bb</te>
    <cw>W. Bruce Croft</cw>
    <ce>bb/au</ce>
    <cw>not(W. Bruce Croft)</cw>
    <ce>fm/au</ce>
  </Title>
  <Description>
    Retrieve bibliographic references
    for works by W. Bruce Croft where
    Croft is not the author of the
    paper containing the reference.
  </Description>
  <Narrative>
    Relevant items will consist of a
    set of article doi's and the bb
    bibliographic entries from that
    article where one of the authors
    is W. Bruce Croft. These should be
    extracted only from articles where
    Croft is NOT one of the authors of
    the article.
  </Narrative>
  <Keywords>
    W. Bruce Croft, W.B. Croft
  </Keywords>
</INEX-Topic>
```

Figure 1: INEX topic 17

Each group submitted suggestions for topics, and altogether 60 topics were selected from those submitted by the participating groups. Figure 1 is an example of such a topic. Of these 30 topics only specified *content words* using the <cw> elements within the <Title>; these correspond to the query terms in a conventional information retrieval query. More constrained were 30 topics with structural aspects as defined by either or both *target element* (as specified by the <te> tag) and *content element* (<ce>) constraints within the <Title>.

The <te> element specifies a target element (or elements) for the topic; in the example shown in Figure 1 the target is the element <bb> — each element contains one bibliographic entry from the references listed at the end of an article. For topics where no target was specified the retrieval system should return an element or elements from matching documents that most closely meet the information need expressed in the query.

The <ce> elements specify that the content words (<cw>) are contained in particular elements; in the example there are two pairs of such constraints — the first requires that Bruce Croft is one of the authors in the bibliographic element, the second requires that Bruce Croft is not an author of the article containing the reference. The <cw> elements can appear without a <ce> constraint, in which case the content words might appear anywhere within the answer element.

For each topic, the participating groups returned (up to) the top 100 answer elements as ranked highest by their system. The answers from all groups were then combined into a pool of between one or two thousand answer elements for each topic that needed to be assessed. Relevance assessments in INEX are undertaken by the participating group who had (usually) formulated the original topic. Our group assessed three of the 60 topics. These assessments involved four-scales of relevance (from *irrelevant* to *highly relevant*) and four degrees of document coverage (*no coverage* to *exact coverage*) for assessing both answer elements as well as surrounding and component elements.

3 System used for INEX experiments

Our aim in participating in the INEX XML-search experiments was to gain experience with a large collection of XML documents and to identify the requirements for a system that can answer realistic queries from people as opposed to queries sent by applications over XML data repositories.

Our approach was to extend our in-house enterprise search engine, PADRE [6], to deal with XML documents and then analyse the results and potential limitations.

3.1 PADRE

PADRE [6] is the core of the Panoptic Enterprise Search Engine [2]. It combines full-text and metadata indexing and retrieval. After all query terms have been processed the matched documents are sorted, first on the basis of how many terms in the query match and then on the basis of the relevance score. The particular relevance formula used [6] is a slightly modified form of the Okapi BM25 function developed by Robertson et al. [8].

3.2 Architecture

As shown in Figure 2 the system includes two main components:

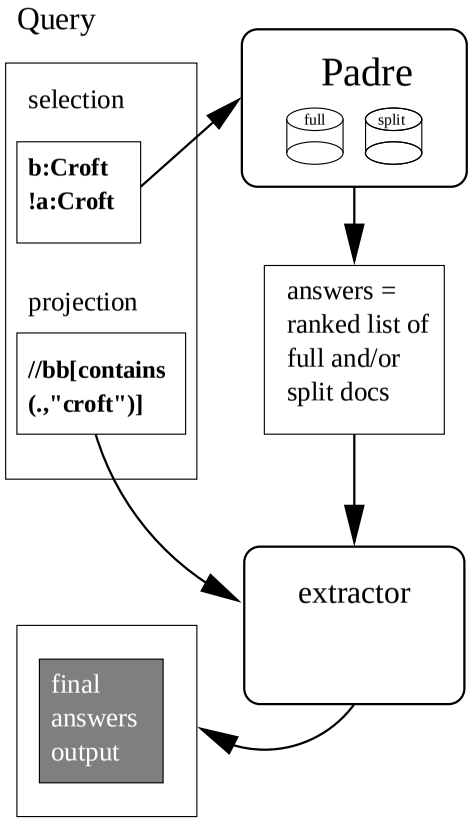


Figure 2: Architecture

- The PADRE search engine selects and ranks the more relevant documents or document fragments that fulfil the constraints (selection).
- The Extractor identifies the type of expected answers and extracts them from the previously returned fragments or documents (projection).

An interface takes the input query and sends the selection query to PADRE and the projection type to the extractor. It also generates the final results in the INEX format.

A separate translation program takes as input the INEX topics and generates a list of queries in a format that is recognised by our system.

3.3 Indexing

We used PADRE's capability for field-based indexing of metadata. For example the index terms for `<p>XML Search</p>` may be represented as `c:XML` and `c:Search`, where the prefix `c` specifies the element in which the information appears.

The fields that will be indexed are explicitly declared in a configuration file of mappings between a path and a given letter. Examples of mappings:

1. `//article/fm/au` → `a`
2. `//bb` → `b`
3. `//p` → `c`
4. `//p1` → `c`

Mapping (1) specifies that a word w occurring in an `<au>` element occurring within an `<fm>` element in an `<article>` will be indexed as `a:`w, while mapping (2) specifies that words occurring a `<bb>` element (for example authors in `//bb/au` in the bibliographic part) will be indexed as `b:`w. Mappings (3) and (4) specify that a word w occurring in a `<p>` or `<p1>` element will be indexed as `c:`w.

A query such as "give me the documents authored by Bruce Croft" can be expressed as: `a:Bruce a:Croft` or alternatively as `a:"Bruce Croft"`. The first query will return initially the documents that contain Bruce and Croft in one article's author element (4 fully matching for the IEEE collection), then documents that contain either word (106 partial matching). The second query will return only documents that contain both words (4 matching), in that order, and may then miss other occurrences such as Bruce W. Croft (not applicable here). The query could also be expressed `a:Bruce +a:Croft` to make the word Croft mandatory then returning only the 4 fully matching.

The above example shows that PADRE mappings will ignore unmapped sub-tags. For example:

```
<au sequence="first">
<fnm>W. Bruce</fnm>
<snm>Croft</snm>
</au>
```

The previous mappings will result in "Bruce" and "Croft" indexed as `a:Bruce, a:Croft`, ignoring the tags `<fnm>` and `<snm>`.

We limited the mappings to what we intuitively thought would be necessary for actual queries, given the DTD and the type of INEX topics.

3.4 Splitting the documents

As described in Figure 2, the system relies on PADRE to select the documents and rank them. Wilkinson [9] demonstrates that extracting elements, such as sections or paragraphs, from the ranked documents is the worst strategy for ranking those elements. So we decided to split the collection in documents fragments to study different strategies that could make better use of the initial PADRE ranking for the ranking of the

final answers. It was expected that it would result in good ranking for the answers to content-only queries, as well as improving the ranking for structured queries, although further extracting may be necessary to return the expected result elements (see next section).

After short analysis of the collection, we split the documents into XML elements corresponding to a small number of tags that take in account two aspects:

- a reasonable granularity of fragments; we did not split into elements as small as titles or bibliographic references; and
- the expected types of result elements, such as sections, paragraphs, front head, figures, table, etc.

To support further processing, some context information was added to the documents fragments. It includes:

- the name of the document this fragment was part of (for example, `/ex/1995/x6059.xml`);
- the actual path of the fragment in the full document (for example, `/article[1]/bdy[1]/sec[2]/p[3]`);
- the skeleton of the tree structure in which the element was embedded; and
- a DOCTYPE declaration corresponding to an extension of the initial DTD that made this fragment a valid document (this was achieved by making most elements optional in the initial DTD).

The two first types of information were the ones expected to be returned as INEX answer in case the fragment was selected. The last two were necessary to parse the fragment in case further processing was necessary (see extraction). Creating document fragments was an interesting experience in itself. Beside the problem of parsing fragments as mentioned above, there was the issue of dealing with entities and special characters. XML parsers interpret them correctly when reading a document but fail to output them back as entities when outputting a sub-tree.

After splitting, the resulting collection was made of the initial documents and all the fragments, making the number of documents increasing by a factor of 100, and the size (in byte) of the collection increasing by a factor of 10.

3.5 Extractor

The role of this module is to process each document (fragment) returned by PADRE and to check whether it is of the expected type(s) for the answer. If it is the case, then the fragment is the answer. If not, some processing is done to identify and extract the relevant elements within the fragment or its embedding document. We call this process the projection.

For example with topic 17, returned elements may be articles, sections or paragraphs, whilst the answers must be bibliographic references (bb). Since bibliographic references cannot be found in sections or paragraphs, the extractor must load the full article and extracts the bibliographic references to papers by Bruce Croft. The actual projection is defined as an XPath with possibly a predicate `contains`, such as `//bb[contains "Croft"]`.

More generally the algorithm works as follows (for a given query): If there is a projection defined, for each returned fragment f:

1. load the fragment, get the name of the embedding article, load the full article A.
2. apply the XPath projection to the article A; this return $e_1, e_2, \ldots e_n$ elements.
3. if $f = e_i$ for one e_i, return the XPath of f; end;
4. if not, calculate $f = father(f)$, if f is not nil go to step 3.
5. if there are e_i that are descendants of the original f, return all of those; end;
6. return the e_i (if any)

Figure 3 (a) and (b) give the XPaths that we use for defining the type of answer for topic 17. As you can see the constraints on `<bb>` elements are very coarse. First this is due to the query been translated automatically from the INEX topic. Second we were using XPath for a fast implementation and XPath does not support similarity measure. For some queries this resulted in returning many irrelevant fragments, and in any case unranked elements within the initial selected fragment. This will be discussed further in Section 4.

3.6 Query translator

Each topic was converted into a query containing a selection component and an optional projection component.

The selection component is translated from the `<ce>` and `<cw>` elements in the topic title and the projection component from the `<te>` element in the topic title. If there is more than one path specified in the `<ce>` element, a Cartesian product of the query terms and corresponding mappings is included in the query. For example, a topic with

```
<cw>Bruce Croft</cw>
<ce>fm/au|bb/au</ce>
```

would be translated as

```
a:Bruce a:Croft
b:Bruce b:Croft
```

We first convert target elements <te> to a standard XPath expression used in the projection component of the query. For example bb -> //bb. Then we use the mappings presented in Section 3.3 to translate this XPath into a Padre command (e.g. b:Bruce). However some elements such as /au may present problems, we could translate

```
au -> //au
```

but alternatively

```
au -> //article/fm/au
```

may be preferable so as to only get authors of the article. We tune this by ordering the mapping rules so the most useful rule matches first.

Examining the behaviour of manually constructed queries we also developed the following heuristics for the query translation process.

Where the selection involves dates, for example if the selection tag within the <ce> element is in the set {yr,pdt/yr,pdt} then we use metadata 'd' and convert <cw> to a numerical value. Also for dates we generate a strong (mandatory) selection condition. For example

```
<cw>1996 or 1997</cw>
<ce>yr</ce>
```

is translated to

```
+d>1995<1998
```

Compound phrases (especially in the keywords) should be searched using phrases. It is difficult to recognise all such phrases, however in some cases (for example in lists of keywords) where phrases are separated by a comma, we can easily identify them. We add the keywords (or phrases) as a list between brackets thus ranking documents with the keywords (or phrases) more highly. For example the phrase "information retrieval" should be translated to the additional restriction on the query ["Information Retrieval"].

When a projection is part of a selection, the same constraints as for the selection should be added to the projection. For example in topic 17, since there are both a selection and a projection on article/bm/bib/bibl/bb we can add the constraints to projection.

There were other cases where manual queries could be carefully crafted but we were unable to discern general translation rules, for example +b:croft (instead of b:croft) as there are plenty of other Bruces as well as Bruce Croft.

3.7 The experiments

We provided three runs for INEX. Each run involved converting the INEX topics into queries sent to PADRE/extractor.

queries on full articles: In this run the queries were generated automatically using the <Title> and <Keywords> from the topic, and only the full articles were searched.

queries on split articles: In this run the queries were generated automatically using the same components of the topic, but fragments as well as full articles were searched.

manual queries: In this run the queries were improved by hand and sent against the split collection.

By comparing the scores of the two first runs we want to evaluate the benefice on indexing fragments. By comparing the last two runs we expect to evaluate the importance of generating good queries, although the manual queries could have been further improved.

As an example, Figure 3 shows each query used for topic 17 in the three runs. Figure 3(a) shows the query generated for both the run on the full articles and the run on the split articles. Figure 3(b) shows the manually constructed query.

4 Analysis of our approach

The results for these runs are not available until the INEX workshop in December. So we are unable to comment yet on whether querying on the splitted collection was more effective than querying the on the full collection. However we are able to make some initial observations of our approach.

Mappings

As we explain above, PADRE allows for field-based indexing and we define the fields to index by mapping paths to a given field. The way it is currently implemented, those fields are regarded as independent and disjoint, and do not account for field hierarchy. The drawback is that any specific mapping will invalidate more generic queries. For example if a mapping is defined for //bb/au, it would be possible to query about one bibliographic reference's author (for example "Bruce Croft"), but not about bibliographic references (<bb>) that contains "Bruce Croft", because "Bruce" and "Croft" would not have been indexed in the field corresponding to "b". Since, XML documents are intrinsically hierarchical, we think that the PADRE strategy for tag indexing should be modified to accomodate this.

Another current limitation in our mapping is that it is not possible to map to attributes. Given

```
<query topic-id="17">
  <selection>
    b:W b:Bruce b:Croft
    a:!W a:!Bruce a:!Croft
    ["W Bruce Croft" "W B Croft"]
  </selection>
  <projection>
    //bb[contains(.,"W") or
         contains(.,"w") or
         contains(.,"Bruce") or
         contains(.,"bruce") or
         contains(.,"Croft") or
         contains(.,"croft")]
  </projection>
</query>
```

(a) query on (i) full and (ii) split documents

```
<query topic-id="17">
  <selection>
    b:"W. Croft" b:Bruce +b:Croft
    -a:"W. Bruce" !a:Croft -a:"W.B."
    ["W. Bruce Croft" "W. B. Croft"]
  </selection>
  <projection>
    //bb[contains(.,"Croft")]
  </projection>
</query>
```

(b) manually constructed query

Figure 3: Topic 17 — full, split and manual queries

the often interchangeable nature of attributes and elements in XML DTDs, it is certainly a limitation.

When the DTD is very large and flexible, it may be difficult to define proper mappings. For example if we want to distinguish between authors of an article in the collection and authors of articles referenced in the bibliographic part, we need two different mappings. However, author of articles can be found with various paths such as `article/fm/au`, `index/index-entry/au`, but also `body/au` or worst in any paragraph with a free syntax (for example "by Bruce Croft"). The last case is obviously beyond XML-search capability, but could be still part of user requirements forsearching XML documents.

Since our mappings are defined at indexing time, they must be carefully designed in anticipating the kind of queries that will be issued. Each time the mapping are redefined the entire collection must be reindexed.

Splitting the collection

Splitting the document collection into a large collection of fragments does not seem a long term option, even though it may prove to be successful. We have already mentioned the increased size of the resulting split collection and the processing involved. Another drawback is that words are indexed several times: for example, once in their original paragraph, once in their embedding section and once as part of the full paper. The consequence is that it introduces a bias in the frequency of words that is used in the PADRE relevance score. It would be more convenient and flexible to define indexes that can emulate fragmentation, if only for evaluating various fragmentation strategies.

Extractor

The main weakness in our current implementation is in the Extractor. As mentioned before it uses a Boolean function to extract the target answers (through an XPath predicate) while a similarity measurement would be adequate. This could be easily implemented using a similarity measurement (for example the cosine measure) between the constraint in the projection and the potential target elements within the current article. Frequency of words could be calculated on the fly related to the embedding article. Following Wilkinson [9], we can also limit the number of answers to a given number (for example 3 or 4), or to fragments that have a similarity measure above a given threshold.

5 Related Work

There has been interest for many years in retrieval of semi-structured information; originally for SGML but with the advent of XML there has been renewed interest in this area.

Indexing and splitting document fragments

A lot of previous work has focused on the following questions.

- What should the fragments be?
- How should relevance of fragments be used to determine the relevance of full documents?

In particular, Wilkinson and Zobel[10] compare various fragmentation schemas in Documents retrieval. They demonstrate that pages are better units of retrieval than either paragraphs or sentences. Another observation is that using fragments as a basis for retrieval, the longer relevant documents are more likely to be retrieved. That is, it helps eliminating bias in the cosine method against long documents. However, their experiments aim at evaluating the benefit of fragmentation for document retrieval, which, again, is not the XML-search objective. Giving advantage to long documents may well play against our objective of finding the most specific fragment.

Wilkinson [9] proposes several similarity measures for ranking documents from section retrieval,

or ranking sections from document retrieval. In the first case, combining the weight of the fragments (related to its type) and the rank of the fragment is improving the precision of retrieved documents, ordered based on their higher ranked sections. In the second case, the experiments determine that extracting the relevant sections from ranked documents results in very poor precision, while ranking sections within ranked documents, and discarding those sections below a threshold provides acceptable results. Since we have been using mostly the first strategy in our INEX experiments (by lack of time), there is room for improvement.

Fuller et al. [5] describe splitting and indexing an SGML-based hypertext collection. The markup language is used to create nodes and links between nodes. For example, a section node will contain its title and the introductory paragraph, while the paragraphs themselves will define other nodes linked back to their embedding section. In this model, a node section is not a viewable element, and the full section need to be rebuilt from its descendant paragraphs. In contrast, we split the XML documents into viewable fragments according to predefined elements such as front-head, sections, paragraphs, figures. A section fragment will contain the text of the full section. A consequence is that words are indexed several times: once in their original paragraph, once in their embedding section and once as part of the full paper.

Amann et al. [1] describe the use of mapping between a community ontology and XML sources. The ontology is understood by all members of the community and used as the common interface component for integrating, querying, structuring and exchanging information. The mapping is used for translating an ontology query into XML XPath queries that are sent to the XML sources for retrieval. The translation occurs at the time the query is issued. The advantage of this approach is that the underlying sources are kept independent and could be indexed by different systems. The only requirement is that they can interpret XPath queries. We use the mapping at indexing time. It makes it more efficient, although for a somehow simplified ontology (no relationship between the "concepts").

Displaying answers

Fuller et al. [5] propose two ways of displaying ranked fragments of documents through a table of content. In the first one the order of ranked fragments is kept across the display of a local table of contents for the corresponding document. In the second one, a ranking for the documents is calculated by propagating the ranks of the fragments to the document and presenting the documents in that order, while the initial rank of the fragments is still displayed in a local table of contents.

The INEX system for assessing results offers an interface for displaying fragments defined by their path within a document. The viewer opens the full document which displays the path of the fragment at the top of the document. The path can be clicked on to jump on the corresponding element, although this element would not be highlighted. Only the element at the end of the path is highlighted in red.

We have not worked on a displaying results yet. We believe that user studies should be carried out, taking in account the information task. The difficulty is to be able to distinguish between improving the precision of answers, and the bias introduced by the interface itself.

6 Conclusions and open questions

In this paper we have described our approach to XML-search based on PADRE, our full text and metadata search engine, with some extensions to support returning specific target fragments. This is still work in progress however preliminary results suggest splitting the documents is an effective strategy. Evaluation of our experimentations, especially with splitting the documents in the collection, will be available at the time of the conference.

INEX gave us the motivation and environment to carry this work, although in its first year it has mostly generated a lot of questions. Within the INEX community there is some disagreement between those approaching the problem from a database perspective (for example placing great importance to structural constraints) and those approaching the problem from an information retrieval perspective (placing more importance on the actual information needs). How to assess relevance of fragments and score answers is still a matter of discussion.

The collection used for the INEX experiment in 2002 contains a particular kind of XML documents, a significant portion of the markup relates to presentation rather than content of the documents. Many other kinds of semi-structured XML is used in a variety of applications ranging from data interchange to semi-structured documents. How well the types of information needs expressed in the INEX dataset apply to other classes of document could be explored.

Another issue is how hypertext links should be handled in querying structured documents. We may be interested in internal links within a document; for example, the connection between a figure and paragraph that contains a reference to the figure. Such queries cannot be expressed very easily in the current format for INEX topics, and answers

may not fit the restriction of just returning elements.

How results should be displayed to the user has not been addressed yet, and it is not clear whether returning fragments helps users to access more easily the information they need. A more open question may be: are fragments the right approach for more meaningful answers?

Nevertheless the INEX experiment has a provided a useful environment for investigating semi-structured retrieval and the collection will provide an important benchmark for future work in spite of its limitations.

Acknowledgements

This work was done while James Thom was a visitor to CSIRO.

Thanks to the INEX organisers.

References

[1] B. Amann, I. Fundulaki, M. Scholl, C. Beeri and A. Vercoustre. Mapping XML fragments to community web ontologies. In *Proceedings Fourth International Workshop on the Web and Databases (WebDB'2001)*, 2001.

[2] CSIRO and Australian National University. P@noptic enterprise search engine. `http://www.panopticsearch.com/`.

[3] DELOS Network of Excellence for Digital Libraries. Initiative for the Evaluation of XML retrieval. `http://qmir.dcs.qmw.ac.uk/inex/`.

[4] N. Fuhr and K. Grojohann. XIRQL: A query language for information retrieval in XML documents. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 172–180, New Orleans, USA, September 2001.

[5] M. Fuller, E. Mackie, R. Sacks-Davis and R. Wilkinson. Coherent answers for a large structured document collection. In R. Korfhage, E. Rasmussen and P. Willett (editors), *Proceedings of the 16th Annual International Conference on Research and Development in Information Retrieval*, pages 294–213, Pittsburg, USA, June 27-July 1 1993.

[6] David Hawking, Peter Bailey and Nick Craswell. Efficient and flexible search using text and metadata. Technical Report TR2000-83, CSIRO Mathematical and Information Sciences, 2000. `http://www.ted.cmis.csiro.au/~dave/TR2000-83.ps.gz`.

[7] R. W. P. Luk, H. V. Leong, T. S. Dillon, A. T. S. Chan, W. B. Croft and J. Allan. A survey in indexing and searching XML documents. *Journal of the American Society for Information Science and Technology*, Volume 53, Number 6, pages 415–437, 2002.

[8] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu and M. Gatford. Okapi at TREC-3. In D. K. Harman (editor), *Proceedings of TREC-3*, Gaithersburg MD, November 1994. NIST special publication 500-225.

[9] R. Wilkinson. Effective retrieval of structured documents. In W. B. Croft and C.J. van Rijsbergen (editors), *Proceedings of the 17th Annual International Conference on Research and Development in Information Retrieval*, pages 311–317, Dublin, Ireland, July 3-6 1994. Springer-Verlag.

[10] Ross Wilkinson and Justin Zobel. Comparison of fragmentation schemes for document retrieval. In D. Harman (editor), *Proceedings of the Third Text REtrieval Conference*, pages 81–84, Gaithersburg, Maryland, 1994.

A Multi-Learner Approach to E-mail Classification

Elisabeth Crawford
School of Information Technologies
The University of Sydney
NSW 2002 Australia
ecrawfor@it.usyd.edu.au

Irena Koprinska
School of Information Technologies
The University of Sydney
NSW 2002 Australia
irena@it.usyd.edu.au

Jon Patrick
School of Information Technologies
The University of Sydney
NSW 2002 Australia
jonpat@it.usyd.edu.au

Abstract

The volume of e-mail which users receive has grown over the past few years. Most users try to cope with this problem by sorting e-mail into folders. In this paper we look at machine learning approaches to performing this classification automatically. In particular we describe a test and select approach to choosing both single learners and ensembles of learners to classify an individual user's e-mail. The results show that it is possible to select the most effective single learner for an individual user, but that selecting an ensemble without overfitting is more difficult. We also show that Widrow-Hoff is a highly effective algorithm for e-mail classification, and discuss the reasons for this.

Keywords Document Management, E-mail Management, Text Categorization, Machine Learning

1 Introduction

The management of personal e-mail is a significant and growing problem for individuals and organisations. Most e-mail users receive a wide variety of e-mail including spam, personal and business communications and e-mail associated with interest groups and mailing lists.

A study by Whittaker and Sidner [27] showed that e-mail is used for a wide variety of important tasks in offices. They found that it was used as a means of receiving and delegating tasks, for keeping track of tasks, and for ongoing correspondence. On average the users in this study received 49 new e-mails a day. Managing e-mail of this volume so it can be retrieved when necessary is a time consuming task.

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

Not only do people receive a lot of e-mail associated with their work, most also receive a great deal of *spam* or *junk* e-mail. Junk e-mail has proliferated since the mid-1990s and it is an increasingly annoying and time wasting task for users to filter their e-mail. Cranor and LaMacchia [5] report that the "spam rate" per day in 1997 for the AOL system was approximately 30%.

E-mail users commonly try to manage their e-mail by sorting it into folders. Filing is a cognitively demanding task [11], thus it would be useful if it could be done automatically. Most commonly used e-mail managers such as Microsoft Outlook, allow the user to hand construct rules to automatically place e-mails containing particular keywords into specific folders. However, Duncheneaut and Bellotti [7] found that despite the potential usefulness of these rules most users do not create them. The majority of users in general avoided customising the software, many also find it difficult to figure out how to use software. In addition, manually constructing a set of robust rules is a very difficult task. Moreover, users are constantly creating, deleting and reorganising their folders. Even if the folders remain the same, the nature of the messages within the folder may well drift over time. The characteristics of the incoming junk e-mail (e.g. topics, frequent terms) also change over time. Hence the rules must be constantly tuned and refined by the user. This is a tedious and error-prone process. A system that can automatically *learn* how to classify e-mail messages into a set of folders is highly desirable. This is where Machine Learning (ML) can help. By studying how a user has classified their email in the past, ML algorithms can be used to build classifiers to associate incoming e-mails with folders.

Research in the area of Text Categorization (TC) has shown that it is possible to automatically build effective classifiers using ML techniques [23].

As a result several systems for automatic e-mail classification have been developed. Cohen [4] uses the propositional learning algorithm RIPPER to induce so called keyword-spotting rules. Bayesian approaches (Naive Bayes and more expressive Bayesian nets) were utilised in [14], [21], [20] and [1]. Nearest-neighbour techniques were used in [24], [12], [15]. In the domain of junk e-mail filtering, Provost [18] found that Naive Bayes outperforms RIPPER and Androutsopoulos et al. [1] reported that Naive Bayes compares favourably with a keyword-based filter similar to that used in Microsoft Outlook.

An important characteristic of e-mail data is the diversity of classification across users. Firstly, there is a lack of unique pre-defined categories; instead each user develops their own categories. Moreover, the type of these categories can vary greatly between users. For example, some users categorize their email according to how or when they need to deal with it; others categorize according to sender or topic. It is also not uncommon for the same user to create some categories according to topic, and others according to actions. This diversity poses a problem for determining what the best ML algorithm for email classification is. In fact there is unlikely to be one best algorithm for all users. This is supported by the results of Crawford, McCreath and Kay [6] who compared the following approaches on e-mail data of five different users: Naive Bayes, keyword rule learner similar to RIPPER, sender based rule learner, tf-idf based approach (see Section 3.2.1) and composite rule learner. It was found that the sender approach performed the best in terms of accuracy on two of the email sets, the keyword rule learner on another two, and the composite rule learner on one. Brutlag and Meek [2] also found that no one learner consistently outperformed the other.

In this paper we propose novel approaches to automatic classification of e-mails, that utilise a combination of ML algorithms. We explore the issues associated with managing the different types of folders created both by different users and by the same user to sort their e-mail. The first approach involves selecting the best single ML algorithm to use for a particular user. The second involves determining the best ensemble of ML algorithms for a particular user. We will first discuss our approach in more detail and then the evaluation of its effectiveness.

2 Multi-Learner Approach

Our approach is two-faceted. Firstly, the differences between the way in which users classify their e-mail has led us to explore an approach that attempts to determine the best ML algorithm to use for a given user. Secondly, as the same user often defines a number of different types of folders (categories), we wish to explore an ensemble of ML algorithms. Our aim is to determine if these two approaches can improve classification effectiveness and how they compare to each other.

In all our experiments we have used separate binary classifiers for each category. It is possible that a user may desire some e-mails to be stored in more than one folder, thus, we do not have to do any extra work (e.g. use error correcting codes) to combine the classifiers to solve the multi-class problem. Instead, when a new e-mail is presented for classification, each binary classifier determines whether or not it belongs in that category.

To determine the best ML algorithm and the best ensemble we propose a test and select method, similar to that described by Sharkey et al. [25] where a validation set is used to build an ensemble for fault diagnosis of a diesel engine. In the first case of our approach, we use the validation set to select the best performing learner overall, and in the second, we use it to select the best ensemble of learners, where a different learner may be chosen for each category.

There are two main differences between the approach of Sharkey et al. and ours. First, in their system the ensemble members are learners from the same type (only neural networks) and are generated using different initial conditions, architecture and data sampling. In contrast, we combine both modules and ensembles. Each module is responsible for one category and the ensembles are formed using one learner from each module. Second, in our approach the ensemble combines learners from different types.

Recently some other multi-learner approaches have been successfully applied for junk e-mail filtering. Sakkis et al. [22] combined a Naive Bayes and memory-based classifier by stacked generalisation and found that the ensemble achieved better performance. Several tree boosting methods were used in Carreras et al. [3] and it was shown that they compared favourably with decision trees, Naive Bayes and k-nearest neighbour. We study the potential of ensembles for general e-mail classification into several folders

2.1 Evaluating Classifier Effectiveness

In many applications of ML, the effectiveness of each algorithm is evaluated by the algorithm's *accuracy*. In TC however, the most commonly used evaluation metric is the *F-measure*. Below these two measures are defined in terms of the two-way contingency table for a binary classifier shown in Table 1.

E-mails	# from c_i	# not from c_i
# assigned to c_i	tp	fp
# not assigned to c_i	fn	tn

Table 1: Contingency Table (tp-true positives, tn-true negatives, fp-false positives, fn-false negatives)

Accuracy is defined as follows:

$$accuracy = \frac{tp + tn}{tp + tn + fp + fn}$$

In TC, however, each class typically has a small number of positive examples and a very large number of negative examples. It is thus possible for a classifier to have a very high accuracy value, without ever detecting a positive example of the class, as the *tn* term dominates the result. Hence, to get a clearer picture of how well a TC classifier has really learnt a class, the measures *precision* and *recall* are used:

$$precision = \frac{tp}{tp + fp}$$

$$recall = \frac{tp}{tp + fn}$$

To obtain an overall measure of effectiveness, the values of precision and recall are combined for the category using the *F-measure*:

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}}$$

where P stands for precision, R for recall and α determines their weighting. In some applications a high precision may be more important than a good value of recall or vice versa. In TC these terms are normally assigned an equal weighting and hence α is set at 0.5. When this is the case the metric is referred to as the *F1-measure* and it is this metric that we use in this paper.

To measure the average performance of a binary classifier over multiple categories, *micro-averaging* [23] is typically used. A summary two-way contingency table is formed by adding up the individual entries in the per-category tables. Then this table is used to compute the summary precision and recall measures.

By evaluating the micro averaged F-measure on the classifiers produced by each of a set of different ML algorithms, we can select for the user, the algorithm that we think will give the best performance on the basis of the validation set. Selecting the best ensemble, however, is more involved.

2.2 Forming the Ensemble

We want to choose the ensemble that when evaluated as a whole achieves the best micro-averaged F1-value. If we simply select the best learner for each category on the basis of the F1-value of the binary classifiers and combine them in an ensemble, there is no guarantee that this will be the best ensemble. Ensemble combinations work best when there is some diversity amongst their components, i.e. when they have different generalisation patterns and few coincident errors [25]. The following example shows that diversity among ensemble members might be more important than combing the best learners in an ensemble.

Suppose that we are selecting the ML algorithm for the ensemble and that all but one of the ML algorithms achieved a F1-measure of 0 on this folder. If we were selecting the best algorithm for the ensemble, we would select the ML algorithm with the positive F1-value. However, this may not be the best choice. Zero F1-values imply that the algorithm found no *true positives*. If an algorithm which found no true positives had very few *false negatives* (due to there being few positive examples of the class) and few *false positives* it may well add more to the ensemble than an algorithm that found only a few *true positives* at the expense of getting many *false negatives*.

What we are faced with is an optimisation problem, where we want to select the ML algorithm for each category that will maximise our objective function - the micro averaged F1-measure for the ensemble. For users who do not have a large number of different folders, we can easily find the ensemble that maximises the objective function using brute force. However, this approach is exponential in complexity; as the number of folders grows it becomes an unrealistic method to use. We can instead use a standard local search technique, such as hill climbing or tabu search to approximate the best solutions. We used a hill climbing approach, starting from an ensemble containing the best learner for each category.

3 Experimental Setup

3.1 Corpora Used for Evaluation

In 2001 Crawford et al. [6] collected a corpus of e-mail messages from five different users. The participants kept back any folders or e-mail messages they felt contained highly sensitive information.

Each of the users used different criteria to determine in what folder their e-mails belonged. For instance, user 1 categorized e-mails mostly on the basis of topic and sender information while user 4 also categorized messages according to when they needed to be acted upon. User 2 was different again, categorizing e-mails solely by the actions performed on them e.g Delete, ReplyAndKeep etc. Thus, the corpora is representative of a number of different styles of classification, despite its relatively small size (a problem common in all the studies of e-mail classification).

User	*Number of E-mails*	*Number of Folders*
1	526	7
2	430	9
3	869	12
4	972	39

Table 2: Statistics for the E-mail Corpora

To evaluate our approach we have used the first four corpora. They range form 430 to 972 e-mails in size and have between 7 and 39 categories, see Table 2. Crawford, Kay and McCreath [6] found that the User 2 and User 4 corpora were the hardest to automatically classify, which is in part explained by their relatively large category/number of e-mails, ratio.

3.2 Preprocessing of the Corpora

3.2.1 Text Representation

The first step in the process of constructing an automatic classifier is the transformation of the e-mails into a format suitable for ML algorithms. In TC this typically involves representing a document as a vector of weighted terms. The most common approach - the *bag of words* approach - uses words as terms with non-binary weights.

In the bag of words approach, all the unique words in the entire training corpus are identified. Each document is then represented by a vector that contains a weighting for every word in the corpus, which represents the importance of that word in the document. Methods for assigning weights to the different words vary, but the most popular is the normalised *term frequency, inverse document frequency - tfidf* approach. We have used the following tfidf function from [23]:

$$tfidf(t_k, d_j) = \#(t_k, d_j) * log\frac{|Tr|}{\#Tr(t_k)}$$

where $\#(t_k, d_j)$ is the number of times the term t_k occurs in the document d_j, and $\#Tr(t_k)$ is the *document frequency* of the term t_k, which is simply the number of documents in the training set Tr in which t_k occurs.

The tfidf values are usually normalised to fall inside the interval $[0, 1]$. We have used the cosine normalisation formula given in [23]. After applying stemming using the Porter Stemming algorithm [17] and removing stop words, this method of representing the text resulted in feature vectors of dimensionality between 4500 and 8600 for each of the corpora. This level of dimensionality was tractable for our machine learners and preliminary experimentation showed no consistent improvements in accuracy from reducing the number of features, therefore we have used the full feature set in our experiments.

3.3 Training, Validation and Testing Sets

To test our approach we have split the e-mails in each corpora into a training set, a validation set and a test set. The e-mails were ordered by the date the user received them and the first 40% were placed in the training set, the second 40% in the validation set and the final 20% in the test set. We created binary classifiers for each category and trained each of the ML algorithms described in the next section on the training set. We then tested each algorithm on the validation set, and used the results of these tests to select the best single learner for the user and the best ensemble according to the micro averaged F1 measure. Finally, we retrained each classifier on the combined training and validation set, and tested the performance of each learner and our ensemble on the test set. In accordance with the standard procedure in TC research, we only train classifiers for folders that contain at least one positive test example (note that this eliminates some folders).

3.4 Machine Learning Algorithms

We have used six different ML techniques to build the email classifiers. We have tried to select a large variety of learners, so that our ensembles may be as rich as possible. We have also been careful to include SVMs and Widrow-Hoff as these algorithms are two of the best performers in TC [23]. The purpose of this is two-fold. Firstly, it means that these highly effective algorithms can be included in our ensembles, and secondly, it ensures that we are comparing our ensemble approach with state of the art classifiers. Below is a brief description of the ML algorithms we have used and their parameters.

- *Support Vector Machines*
  Support Vector Machines (SVMs) [26] are a recently developed ML technique. They find the maximum margin hyperplane between two classes using the the training data and applying an optimisation technique. SVMs have shown good generalisation performance on many classification problems, including TC [8]. What makes them particularly suitable for TC is their ability to handle high dimensional features and their automatic tuning of parameters [10]. We have used the Weka implementation [28] of the SMO algorithm for SVMs [16] with both linear and quadratic kernels.

- *K-Nearest Neighbour*
  K-Nearest Neighbour (KNN) is the most commonly used memory based ML algorithm. It is a lazy learner as it stores all the training samples and does not build a classifier until

a new example is presented for classification. A distance metric is used to find the similarity between the new example and all training examples. The class is then decided by the majority vote of the closest k instances. In our experiments we used the Weka [28] implementation of the kNN algorithm, with cosine similarity as the distance metric and distance weighted voting, for k equal to one, five and thirty.

- *Decision Tree*
  Decision trees (DTs) are the most popular inductive learning algorithm. The nodes of the tree correspond to attribute tests, the links - to attribute values and the leaves - to the classes. To induce a DT, the most important attribute (according to the information gain) is selected and placed at the root; one branch is made for each possible attribute value. This divides the examples into subsets, one for each possible attribute value. The process is repeated recursively for each subset until all instances at a node have the same classification, in which case a leaf is created. To classify an example we start at the root of the tree and follow the path corresponding to the example's values until a leaf node is reached and the classification is obtained. To prevent overtraining DTs are typically pruned. The most popular DT learning package is Quinlan's C4.5 [19]. We have used the Weka implementation [28] of the C4.5 algorithm in our experiments.

- *Perceptron*
  A perceptron [9] is a simple type of neural network. It tries to induce a linear decision boundary that separates the positive and negative examples in the training data. The learning process involves a number of passes over the entire training set. After the presentation of each example, the boundary is adapted (moved towards or away from the example) depending on how the example was classified (correctly or not). The algorithm terminates when all the training examples are correctly classified or when the epoch limit is reached. A new example is classified by determining which side of the boundary it lies on. The main disadvantage of the perceptrons is that they can only separate linearly separable data and are not able to produce a theoretically good boundary, e.g the one that defines the maximum separation between positive and negative examples. However, perceptrons are quite fast and have been very effective in the TC domain. We have used the Weka implementation [28] of the perceptron algorithm in our experiments.

- *Widrow-Hoff*
  The Widroff-Hoff (WH) algorithm [9] is a neural network algorithm that learns a linear decision boundary similarly to the perceptrons. It adjusts the weights of the network in order to minimise the mean squared error of the difference between the actual and target outputs of the network for each training example. This amounts to an approximate steepest gradient descent on the error surface. Widrow-Hoff can not learn non-linear decision boundaries, however if run on non-linear data it will try to find the linear boundary that results in the lowest mean squared error. This makes it more noise tolerant and allows for better generalisation. We have used our own implementation of the standard Widrow-Hoff algorithm in our experiments.

- *Naive Bayes*
  Naive Bayes (NB) is a simple but highly effective Bayesian learner. It uses the training data to estimate the probability that an example belongs to a particular class. It assumes that attribute values are independent given the class. This assumption clearly has no basis in many learning situations including TC, nonetheless Naive Bayes can produce very good results. For more information on Naive Bayes see [13]. We have used the Weka implementation [28] in our experiments.

4 Results and Discussion

4.1 Determining the Best ML Algorithm for Each User

Table 3 presents the results for each user on validation and test set. Each entry represents the micro averaged (over all folders) F1 measure and the best results are given in italics. A comparison between the results on the validation and test set for the different learners indicates similar performance patterns. Hence the validation set can be effectively used to select the best single learner for each user. Once the best learner for each user is selected using such a test and select approach, the classifier can be retrained on all e-mails sorted by the user (i.e. training, validation and test set) and used for future automatic classification.

Contrary to [6] and [2], we found that the same learner performed the best for all the users. WH, which to the best of our knowledge has not been applied to e-mail classification before, performed better than any other single learner on each corpus. Linear SVMs (SVM1) also performed well, coming second for users 1 and 3 and third for user 4. Another linear classifier - perceptron - showed relatively good performance, while nearest neigh-

Learner	User1 Validation	User1 Test	User2 Validation	User2 Test	User3 Validation	User3 Test	User4 Validation	User4 Test
KNN1	0.157	0.506	0.149	0.406	0.642	0.73	0.563	0.34
KNN5	0.607	0.719	0.056	0.247	0.498	0.596	0.568	0.484
KNN30	0.65	0.715	0.0	0.475	0.281	0.468	0.366	0.364
SVM1	0.668	0.794	0.377	0.406	0.807	0.807	0.551	0.585
SVM2	0.662	0.758	0.161	0.379	0.603	0.756	0.428	0.516
C.45	0.621	0.713	0.362	0.434	0.769	0.782	0.528	0.573
NB	0.614	0.697	0.365	0.387	0.624	0.471	0.265	0.241
Perceptron	0.667	0.72	0.373	0.397	0.562	0.773	0.329	0.593
WH	*0.688*	*0.806*	*0.434*	*0.485*	*0.838*	*0.824*	*0.589*	*0.609*

Table 3: Single Learner Results for Each User

bour was inconsistent and Naive Bayes performed the worst of all.

It is worth considering why our results do not support the previous results of Crawford et al. [6], indicating that no one learner consistently outperformed the others. We cannot directly compare the results due to our different experimental setup, however the answer probably lies in the difference between the learners being evaluated. Unlike the learners used in [6] WH and SVMs are two of the most effective TC methods. Furthermore, the strong performance of WH, linear SVM and perceptron imply that sophisticated linear methods work well for email classification. Crawford et al. [6] have largely used inductive rule based approaches; the only linear approach they use is NB which was the worst performer in our experiments. The results strongly suggest that WH is a more consistently good performer across users than previous approaches.

The results shown in Table 3 demonstrate a strong difference in the difficulty of automatically categorizing different users' e-mails. Consistent with [6] our results show that the coarser grained e-mail sets of users 1 and 3, where many e-mails were classified according to topic, were easier to classify than the finer grained user 2 and user 4 corpora. The fact that these corpora were more finely grained meant that for each binary classifier there was less training data, making the classification harder to learn. Further, these second e-mail sets were also classified more according to actions, which intuitively seems like a harder sort of classification to learn with TC methods.

With the exception of Naive Bayes our results indicate that holistic linear algorithms (WH, SVM1, perceptron) perform relatively well compared to non-linear approaches (KNN1, KNN5, KNN30) and inductive keyword based ones (C4.5). The linear models have a strong advantage over models like DTs in that not only can they recognise key features (e.g. in WH connection weights indicate the importance of features), but they can also make an assessment using all available information which is important given the amount of concept drift found in e-mail.

The very poor performance of Naive Bayes can be attributed to overzealous predictions that e-mails belong to the class in question. The Naive Bayes classifiers all had very high recall, but at the expense of very low precision due to many false positives. This problem could perhaps be lessened by thresholding the predictions, so that the classifier cannot predict membership unless the probability of membership is significantly higher than the probability of non membership.

K-nearest neighbour is a simple method which was found to perform best on a TC task(Reuters) when compared to 13 other methods [29]. However, we found that it was inconsistent and often performed poorly on e-mail classification. Nearest neighbour approaches perform well when the data is not noisy and when each attribute contributes approximately equally to the classification. Compared to news wire texts (e.g Reuters), e-mail contains a great deal of noise, due to the vastly different writing styles of e-mail senders and most probably inconsistencies in classifications. Further, the lack of a generalised model in a domain where certain key-words really do matter more than others (e.g. the sender's name is an important feature) is a serious disadvantage.

Finally, it is worth considering why WH performs better than the perceptron and SVM classifiers. WH classifiers are more powerful than perceptrons as they find a boundary which minimises the classification error while perceptrons simply look for a boundary that works. The weaker performance of the SVMs with the quadratic kernel (SVM2) versus the linear kernel simply implies that a linear model more effectively represents that data. It is surprising however, that WH performs better than the linear kernel SVM and this issue requires further investigation.

4.2 The Evaluation of the Ensemble

The results are summarised in Table 4. It shows the ensemble members selected for each folder based on

User	Ensemble	Validation	Test
User1	[WH,SVM1,SVM2,WH]	0.688	0.8
User2	[C4.5,KNN30,SVM1,SVM1,SVM2,WH]	0.473	0.504
User3	[WH,KNN5,WH,SVM1,C4.5,SVM2,WH,SVM1,SVM1,WH]	0.858	0.788
User4	[WH,WH,KNN5,WH,KNN1,KNN1,WH,NB,WH,WH,WH,WH,SVM1,WH]	0.625	0.524

Table 4: Ensembles and F1 Results on Validation and Test Set for Each User

their performance on the validation set, together with the overall micro averaged F1 measure for the respective ensemble on the validation and test set. By comparing the results on the test set in Tables 3 and 4, it can be seen that the ensemble outperformed the best single learner (WH) on user 2's e-mail, obtained almost the same performance on user 1 and was less successful on users 3 and 4.

The failure of the approach on three of the corpora has more to do with the method of ensemble selection, than with a problem inherent with the idea. It is possible to choose on the test set an ensemble that is at least as good as the best single learner (as the ensemble members can all be from the same learner), however such ensembles are not being chosen on the basis of the validation set. By comparing the performance on the validation sets for users 2, 3 and 4, it can be seen that the ensembles (Table 4) performed considerably better than the best single learners (Table 3). The issue is then to explain, why the best ensemble for the validation set is not the best ensemble for the test set and how we can go about selecting a better ensemble.

Clearly we are overfitting the validation set when we choose the ensemble. This problem is most likely compounded by concept drift and the problem of sampling. During the period of time covered by the validation set, the user for example may have received a lot of e-mails about certain topics and the ensemble selected reflects this. However, during the period covered by the test set, the user may not have received many e-mails on these topics at all. This could either be due to true concept drift (the user may never receive many e-mails on this topic again redefining the overall nature of the folders) or it may be a problem with sampling - perhaps the user will receive more e-mails on these topics in a few weeks time. It is most likely that both of the above contributed to the overfitting. This is reinforced by the fact that user 2's email was collected over a two week period only, whereas the other users' e-mails range over periods from 1 to 5 months. Thus it is likely that the approach worked well for user 2 as there was much less room for concept drift and sample induced differences.

We have done a small amount of experimentation with different training/validation/testing splits that has suggested that some members of the ensemble are more stable than others (i.e. some were regularly selected for particular categories). This indicates the need for an ensemble selection process that is more resistant to sampling problems.

5 Conclusions and Future Directions

This paper has shown that a test and select method can effectively be used to choose the most effective learner for building a classifier for an individual's email. On each of our four test corpora the Widrow-Hoff algorithm achieved the best performance in terms of micro averaged F1. Neural network classifiers are often thought of as being inscrutable, however simple neural classifiers such as Widrow-Hoff can be made understandable for email users. Using rule extraction techniques from trained neural networks [30], we can communicate to the user the reasons for classification decisions in the same way we can when using rule based techniques [6] [4]. Thus we can take advantage of the superior performance of the Widrow Hoff algorithm without sacrificing scrutability.

We have also explored a test and select approach for forming an ensemble that uses different learners for different folders. This approach was successful in improving classification performance on one of the four corpora. We believe ensembles with better generalisation could be selected for the other three corpora using more sophisticated methods. To determine the ensemble it would be worth exploring the use of the sliding window tester described in [6] as this approach should be more resistant to sampling problems. We would also like to collect datasets of e-mails that span a number of months from users that receive a large amount of email, so that we can test our ideas on more corpora. Finally, the work in this paper further confirms the need for benchmark e-mail corpora allowing consistent evaluation of the approaches for automatic e-mail classification.

6 Acknowledgements

The first author would like to thank the Capital Markets CRC, CMCRC Ltd. for their support.

References

[1] I. Androutsopoulos, J. Koutsias, K. Chandrinos and C. Spyropoulos. An experimental

comparison of naive bayesian and keyword-based anti-spam filtering with personal e-mail messages. In *23rd Int. ACM SIGIR Conf. pp. 160-167*, 2000.

[2] J. Brutlag and C. Meek. Challenges of the email domain for text classification. In *17th Int. Conf. on Machine Learning*, 2000.

[3] X. Carreras and L. Marquez. Boosting trees for anti-spam email filtering. In *4th Int. Conf. on Recent Advances in NLP*, 2001.

[4] W. Cohen. Learning rules that classify e-mail. In *AAAI Spring Symposium on Machine Learning in Information Access, pp. 18-25*, 1996.

[5] L. Cranor and B. LaMacchia. Spam! *Comm. of the ACM v.41 n.9 pp.74-83*, 1998.

[6] E. Crawford, J. Kay and E. McCreath. Iems - the intelligent email sorter. In *19th Int. Conf. on Machine Learning, ICML*, 2002.

[7] N. Ducheneaut and V. Bellotti. E-mail as habitat: an exploration of embedded personal information management. *Interactions v.8, n.5, pp.30-38*, 2001.

[8] S. Dumais, J. Platt, D. Heckerman and M. Sahami. Inductive learning algorithms and representations for text categorization, 1998.

[9] J. Hertz, A. Krogh and R. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley, 1991.

[10] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *10th European Conf. on Machine Learning*, 1998.

[11] A. Kidd. The marks are on the knowledge worker. In *CHI'94 Conf. on Human Factors in Computing Systems pp.186-191*, 1994.

[12] P. Maes. Agents that reduce work and information overload. *Comm. of the ACM, v.37, n.7, pp.31-40*, 1994.

[13] T. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[14] P. Pantel and D. Lin. Spamcop: A spam classification & organization program. In *AAAI-98 Workshop on Learning for Text Categorization pp.95-98*, 1998.

[15] T. Payne and P. Edwards. Interface agents that learn: An investigation of learning issu es in a mail agent interface. *Applied Artificial Intelligence, v.11, p.1-32*, 1997.

[16] J. Platt. Fast training of support vector machines using sequential minimal optimization. *Advances in Kernel Methods - Support Vector Learning*, 1998.

[17] M. Porter. An algorithm for suffix stringing. *Program, v.14, n.3, pp. 130-137*, 1980.

[18] J. Provost. Naive-bayes vs. rule-learning in classification of email, 1999.

[19] J.R. Quinlan. *C4.5: Programms for Machine Learning.* Morgan Kaufmann, 1993.

[20] J. Rennie. ifile: An application of machine learning to e-mail filtering. In *KDD-2000 Text Mining Workshop, Boston*, 2000.

[21] M. Sahami, S. Dumais, D. Heckerman and E. Horvitz. A bayesian approach to filtering junk e-mail. In *AAAI-98 Workshop on Learning for Text Categorization*, 1998.

[22] G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C. Spyropoulos and P. Stamatopoulos. Stacking classifiers for anti-spam filtering of e-mail. In *6th Conf. on Empirical Methods in NLP, pp.44-50*, 2001.

[23] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys, v.34, n.1, pp.1-47*, 2002.

[24] R. Segal and M. Kephart. Mailcat: An intelligent assistant for organizing e-mail. In *3d Int. Conf. on Autonomous Agents, pp.276-282*, 1999.

[25] A. Sharkey and E. Sharkey. The "test and select" approach to ensemble combination. In *First Int. Workshop on Multiple Classifier Systems.* LNCS, 2000.

[26] V. Vapnik. *Statistical Learning Theory.* Wiley, 1998.

[27] S. Whittaker and C. Sidner. Email overload: Exploring personal information management of email. In *CHI, pp. 276-283*, 1996.

[28] I. Witten and E. Frank. *Data Mining - Practical Machine Learning Tools and Techniques with Java Implm.* Morgan Kaufmann, 2000.

[29] Yiming Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval v.1, n.1/2, pp.69-90*, 1999.

[30] Z. Zhou, Y. Jiang and S. Chen. Extracting symbolic rules from trained neural network ensembles. *AI Comm., in press*, 2003.

Information Extraction in the KELP Framework

Robert Dale, Marc Tilbrook
Centre for Language Technology
Macquarie University
E-mail {rdale|marct}@ics.mq.edu.au

Cécile Paris
Intelligent Interactive Technology Research Group
CSIRO Mathematical and Information Sciences
Cecile.Paris@cmis.csiro.au

Abstract

In this paper, we describe some early steps in a new approach to information extraction. The aim of the KELP *project is to combine a variety of natural language processing techniques so that we can extract useful elements of information from a collection of documents and then re-present this information tailored to the needs of a specific user. Our focus here is on how we can build richly structured data objects by extracting information from web pages; as an example, we describe the extraction of information from web pages that describe laptop computers. A principle goal of this work is the separation of different components of the information extraction task so as to increase portability.*

Keywords Information extraction, natural language generation, document personalisation.

1 Introduction

Information Extraction (IE [1]; the process of identifying a pre-specified set of key data elements from a free-text data source) is widely recognised as one of the more successful spin-off technologies to come from the field of natural language processing. The DARPA-funded Message Understanding Conferences resulted in a number of systems that could extract from texts, with reasonable results, specific information about complex events such as terrorist incidents or corporate takeovers. In each case, the task is manageable because (a) some other means has determined that the document being analysed falls within the target domain, and (b) the key information required is typically only a very small subset of the content of the document. A major component task is named entity recognition [3], whereby people, places and organizations are located and tracked in texts; other processing can then take the results of this process to build higher order data structures, establishing, for example, who did what to who and when. In this paper, we describe a new mechanism that relies on a hand-constructed knowledge template, along with a general inference mechanism we call **path merging**, to reconcile and combine the informational elements found in a text.

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

2 Background

The goal of the KELP[1] project is to develop technology that can extract information from a collection of web pages that describe similar things, and then collate and re-present this information in such a way as for a user to make it easy to compare those things. Suppose you are interested in purchasing a new cell phone: you could check out a consumer magazine, or visit a web site that presents comparisons of the different available models, but those sources are typically not up-to-date. You might visit the manufacturers' web pages to obtain information from the original sources, but this is a painful, slow process, and comparison is hindered by the different terminology each vendor uses; further, all these sources provide information for a 'typical' visitor, rather than tailoring what they present to your particular needs and interests.

In KELP, we aim to build technology that can mine the information from these source web pages, and then, using techniques we have discussed elsewhere (see [2]), re-present it in a form that is tailored to needs and interests captured in a specific user profile. Our initial experiments have been in the context of web pages that describe laptop computers. Based on an analysis of around 120 web

[1]KELP stands for Knowledge Extraction and Linguistic Presentation, emphasising that the project is concerned both with document analysis and document production techniques. The KELP project is funded by CSIRO, whose support we gratefully acknowledge.

```
<?xml version="1.0" ?>
  <laptop_info>
    <laptop_id>
      <manufacturer>NEC</manufacturer>
      <series>Versa</series>
      <model>Premium PIII 1GHz</model>
    </laptop_id>
  <components>
    <cpu>
      <cpu_type>Pentium III</cpu_type>
      <cpu_speed>
        <number>1</number>
        <unit>GHz</unit>
      </cpu_speed>
    </cpu>
  ...
</laptop_info>
```

Figure 1: A portion of a filled knowledge object

pages describing laptop computers, we developed a template that captures the range of information we might want to extract regarding a laptop computer, and then set about developing techniques to extract the relevant data from these web pages. Part of a typical filled template, or, as we call these structures within KELP, a **knowledge object** or KO, is shown in Figure 1. We have elided this structure to make it fit the space available; it should be obvious that the quantity and complexity of the data to be extracted is significant.

3 Our Approach

Our starting point is a definition of a KO, as shown in the previous section; defined in our current system via an XML DTD, this is a hierarchical structure that specifies the nature of the data to be extracted from the source documents.

The simple key to our approach is to recognize that fragments of the text can be correlated with different parts of the KO structure. For example, we know that the manufacturer will be a company name; and we know that the hard disk capacity will be measured in Mb or Gb. Thus, we can assign type information to the leaf nodes in this structure. At the same time, words in the text can serve as indicators of particular attributes: so, if a sentence contains the phrase *removable storage*, we can use this to hypothesise the presence of a particular attribute in the text. We think of each such text fragment as a piece of evidence; faced with evidence from the text of a collection of attributes and values, the goal is then to combine this information to populate a KO.

The architectural model we use thus consists of the following components. An **annotation resource file** provides a correlation between between arbitrarily complex textual patterns and the knowledge object constituents for which these patterns provide evidence.

A **text scanner** processes each input document, searching for the patterns specified in the annotation resource file. Each time a match is found, this generates a hypothesis, in the form of a **path fragment** associated with a piece of text. The consequence of processing a document is thus a collection of path fragments that capture the evidence found in the document.

The **path combiner** then takes this set of path fragments and attempts to put these together to build a complete knowledge object. Path fragments may contribute to multiple hypotheses about the object being constructed, so the combiner uses the target knowledge object template as a source of constraints on what is possible.

In most situations, this will not produce a single KO as a result; there will still be ambiguities resulting from textual fragments providing evidence for more than one KO constituent. At this point, we resort to a collection of **inference strategies** to resolve the ambiguities.

Of course, if we had full broad-coverage natural language processing available, then this would achieve the same result: we could just parse the entire text, carry out semantic analysis, and build a representation of the text. However, such an approach is not feasible given the current state of NLP technology, so our aim here is to build on the simpler pattern-matching approaches found in the IE literature, augmented with more sophisticated higher-level processing. Our architectural breakdown stratifies the knowledge used in a way that supports easy maintenance and portability: the annotation resource file is a relatively simple declarative knowledge source developed anew for each domain; the text scanner and path combiner are generic components that do not embody any domain-specific knowledge; and higher-level knowledge of the domain is factored out into the KO template and the inference strategies.

3.1 Paths

We can view a KO as a graph structure (and for present purposes a tree) into which extracted information can be placed. The arcs between nodes in this tree are labelled with the same names as the element tags in the XML DTD; a path is a sequence of arcs in the tree. Each attribute corresponds to path from the root of tree, and each value is a data element that resides at the end of that path. Paths may share common initial subsequences: this means that we use hierarchy in the tree to cluster related pieces of information into subtrees.

We use the notation A:B:C to notate **path fragments**. If a path is from the root of the tree, the path contains an initial ':'; if the path has a value

at its end, then we use a terminal ':' to indicate this. A **path equation** indicates the value at the end of a path: for example

:laptop:iodevices:keyboard:numkeys: = 131

Each piece of evidence we find in a text can be annotated with a path fragment: this fragment can be of various kinds depending upon how strong the evidence is. The example above is a **complete** path fragment, where there is no doubt that a particular value is the value of a particular attribute.

A path fragment can be **initial**: this means that we don't have a complete path but we do have some initial sequence of arcs in a path. So, the string *on-board memory* might correspond to the following annotation:

:laptop:memory:on-board

We don't know at this point what aspect of the on-board memory is being described.

A path fragment can be **medial**: this means that we don't have a complete path but we do have the some sequence of arcs in the middle of a path. So, a string like *memory capacity (maximum)* may correspond to main memory, graphics memory, or perhaps some other kind of memory; this corresponds to the following medial path fragment:

memory:maximum:bytesize

Finally, a path fragment can be **final**. This means we have some sequence of arcs at the end of a path. So, the string *2Gb* corresponds to the following pair of path fragments:

bytesize:unit: = Gb
bytesize:number: = 2

To operationalise these notions, the annotation resource file correlates arbitrarily complex textual patterns with path fragments. These patterns are then used by the text scanner to generate hypotheses about the information in the text, expressed in terms of the path fragments; the complete analysis of a text thus results in a set of textual clues and their corresponding hypothesised path fragments.

3.2 Path Merging

A KO consists of set of paths, and a fully instantiated KO has a value for each path that makes up the KO. We can characterise an instantiated KO by a set of path equations:

:laptop:model:manufacturer: = Dell
:laptop:model:series: = Inspiron
...
:laptop:iodevices:mouse:numbuttons: = 3

From a text, we derive a set of path fragments like those shown in the previous section. Our goal is to take this collection of path fragments and to derive from them a set of path equations that define an instantiated KO. Formally there are many combinations of path fragments that we could entertain.[2] A number of cases need to be considered.

First, we do not have to do anything to path fragments which are complete; note, however, that we do not want to just forget about these, since their presence may rule out some other possible combinations (in the example above, if we are tempted to merge two path fragments that would give us a different number of keys, then we have a good reason for not doing this).

Second, two path fragments may share some arcs: so, in the above, a possible combination results in

memory:maximum:bytesize:unit: = Gb

This is a possible combination but not a necessary one: arc labels are not unique, so it's possible that this particular bytesize:unit fragment does not belong with the memory:maximum:bytesize fragment.

Third, from a formal point of view, any pair of paths can be combined, with the exception that an initial path can only appear at the front of a combined path, and a terminal path can only appear at the end of a combined path. In such cases, where there is no overlap, there is essentially missing material in the middle, which we indicate using '...'. So, we might have a combined path that looks something like the following':

:laptop:memory:...:bytesize:unit: = Gb

This is a case where we have some possible evidence for a memory size but we don't know if it is the standard on-board memory, the expanded-to-maximum memory size, or some other aspect of memory size. Of course, not all formally possible combined paths are actually possible. The KO definition provides a way of ruling out impossible path combinations. Our technique for filtering the paths is as follows.

First, we take the set of paths that constitute a KO, called the KO **path set**; this will look something like the following:

:laptop:model:manufacturer:
:laptop:model:series:
...
:laptop:iodevices:mouse:numbuttons:
...

Then, we take the set of path fragments derived from the document. We first separate out the medial paths and the complete paths, leaving the initial paths and final paths. We then produce all

[2]The ideas discussed here have been strongly influenced by work in graph-based unification [4].

possible initial × final combinations, resulting in what we call the IF **path set**. Each element of the IF path set has the form

:A:B:C:...:X:Y:Z:

We then compare each element of the IF path set against the KO path set. Any match of an IF path against the KO path set constitutes an **instantiation**: it provides a possible substitution for the '...' part. Note that any IF path may result in multiple instantiations. If an IF path results in no instantiations, then it is not completable given the KO definition, and can be discarded. The remaining IF paths make up the **filtered IF path set**; and each element of this set may correspond to multiple instantiations. We notate instantiations as follows:

:A:B:C:[P:Q:R]:X:Y:Z:

This indicates that P:Q:R is a possible completion derived from the KO path set. In effect, material between square brackets is hypothesized on the basis of top-down knowledge; we have not extracted direct evidence for it from the text.

Next we take the medial paths and see if they support any of these instantiations by matching them against the instantiations. Again, a medial path may support multiple instantiations. A medial path may actually overlap with the initial or final path in an instantiation. By combining this information, we produce a set of paths built from the initial, medial and final path fragments in the text, and filtered using the KO path set. We then need to reduce the set of instantiations in the filtered IMF path set so that we end up with a set where each I, M or F element only plays a role once. Some combinations of the contributing I, M and F fragments are are more likely than others. We therefore need to assess each combination, and where there is a competing demand for a piece of evidence, determine which of the alternative uses of that evidence is most plausible.

3.3 Adding Reasoning

The set of hypotheses constructed so far uses a combination of bottom-up knowledge from the textual source itself, and top-down knowledge from the KO. This does not necessarily result in a completely instantiated KO, and so we need to add heuristics that allow us to choose between competing hypotheses.

Note that the architectural separation we have adopted allows the incorporation at this stage of arbitrary intelligence to the process, thus focussing the knowledge-based processing in one place; here, we describe the incorporation of a simple heuristic based on a distance metric.

Ultimately, where we have competing instantiations, we can assign probabilities to each. One simple probability measure is based on the distance between the initial path fragment (which generally corresponds to the attribute being considered) and the final path fragment (which corresponds to the value being assigned to that attribute): we can assign scores so that those instantiations that have closer I and F evidence will score higher. Then, we select from this set a smaller set of paths that (a) uses each piece of evidence only once and (b) takes account of the scores. The result is a set of hypotheses that populate the KO using all the data located in the source document, with each piece of evidence being used once.

This is, of course, a very simple technique, but one that seems to work well on the basis of our initial experiments, at least for information extracted from free-form text. Far more elaborate heuristics could be incorporated in the same way.

4 Conclusions

We have presented an approach to information extraction that separates out the different knowledge sources and types of knowledge required. The approach makes use of both top-down and bottom-up knowledge sources, and neatly partitions domain-specific and domain-independent processing: we believe the mechanisms described here should be easily portable to a new domain by constructing a knowledge object template for that domain, and an appropriate annotation resource file. The text scanner and path combining modules are domain independent, as are the current inference strategies; as the work develops, we would expect the inference strategies to break down into those which are domain-dependent and those which are domain-independent.

References

[1] J. Cowie and W. Lehnert. Information extraction. *Communications of the ACM*, Volume 39, Number 1, pages 80–91, 1996.

[2] R Dale, S J Green, M Milosavljevic, C Paris, C Verspoor and S Williams. Using natural language generation techniques to produce virtual documents. In *Proceedings of the Third Australian Document Computing Symposium (ADCS'98)*, Sydney, Australia, August 21 1998.

[3] Andrei Mikheev, Claire Grover and Marc Moens. XML tools and architecture for named entity recognition. *Markup Languages*, Volume 1, Number 3, pages 89–113, 1999.

[4] S. Shieber. *An Introduction to Unification-Based Approaches to Grammar*. CSLI Lecture Notes. Chicago University Press, Chicago, 1986.

Workflow Based Just-in-time Training

J. Davis, J. Kay, K. Lin, J. Poon, A. Quigley, G. Saunders, K. Yacef
School of Information Technologies
University of Sydney
Australia
{*jdavis,judy,kenl,josiah,aquigley,gsaunder,kalina*}*@it.usyd.edu.au*

Abstract

This paper focuses on the problem of information overload for newcomers in an organisation. We propose to address it by constructing a smart personal training assistant based upon workflow tools to drive temporal management of a just-in-time workplace training system which will deliver a personalised and structured presentation of organisational documents.

Keywords Document Workflow, Document Management, Information Retrieval.

1 Introduction

Improving newcomers' learning and integration into an organisation, institution, board or project, is a crucial problem encountered in almost any domain. Newcomers to a project team or organisation often face information overload that impedes their learning [4] and their resultant performance. If the newcomer holds an executive position, the situation becomes critical for the entire organisation. As with any training, structuring the exposure of newcomers to this knowledge helps them to learn more quickly and effectively, thus improving their efficiency and motivation, and also reducing financial costs for the organisation.

We propose to combine the benefits of research in personalised training, personalisation based on a user modelling framework for representing a user's knowledge and preferences and on knowledge management and the use of workflow technology to tackle this problem.

The combination of workflows, personalisation and tutoring is novel. However, there has been some work exploiting workflow technology to support teaching. The Flex-eL project [2, 1, 3] aims to support students as they work their way through a University level course. It provides "flexible learning pathways" based on workflow models of the study processes for the course. This approach relaxes constraints on the timing of learning processes, enabling the student to proceed at their own pace. It is also capable of providing the student with multiple pathways through the course, enabling them to tailor their learning to their own individual style.

The work reported in this paper is funded in part by the Cooperative Research Centres Program through the Department of the Prime Minister and Cabinet of the Commonwealth Government of Australia.

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

2 Overview of Architecture

Figure 1 illustrates the architecture of our JITT system. The training assistant makes use of workflow tools and draws upon user modelling and a document repository with links to the workflow.

Workflow tools. The workflow tools enable the training assistant to deliver information in a timely fashion. A workflow model of each organisational process is generated which defines the temporal ordering of the various stages of the process. In addition, the workflow contains information about the time intervals between the various stages. When the user starts an organisational process, a workflow instance is created enabling the training assistant to track the progress of the user as they go about their various tasks. The assistant is then able to determine, based on the current stage in the workflow, what information the user requires in order to complete the current stage and prepare for imminent stages in the workflow.

User modelling. The system maintains a model of each user of the system. This model contains information about the user's present knowledge and past experience and is used by the training assistant to personalise the presentation of information to the user. The personalisation enables the user to easily find the information they need without having to sift through irrelevant data or data which they already know. In addition, the user may modify their user model to incorporate their personal preferences about the frequency and form of information delivery.

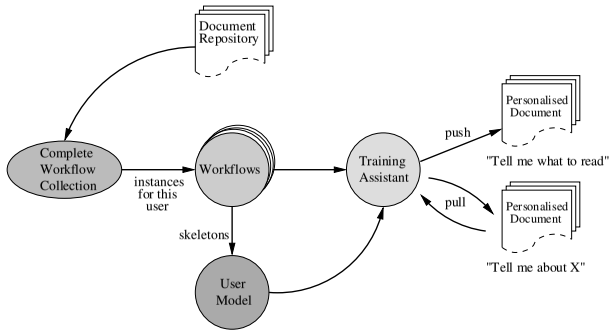


Figure 1: Just-in-time Training architecture

Document repository. The system maintains a repository of training documents which form the basis for the content provided by the training assistant. These training documents are linked with various stages in the workflow models of organisational processes. In addition, the documents are formatted in such a way as to enable the training assistant to extract the most relevant information and format it in a consistent manner. It is also possible to link a trainer, that is, a human being with expertise on a given topic, to various stages in the workflow. Essentially, this generalises the notion of a 'document' so that the system can deliver the details of the expertise and contact details of a potential human advisor. This is particularly important for enabling JITT to take advantage of tacit knowledge, which is available as in a formal document.

In summary, the training assistant examines the current stage(s) in the various workflows and generates a list of documents from the document repository that are associated with these workflow stages. This list of documents is then processed to remove irrelevant or previously known information based on the user model. Finally, the assistant formats the documents according to the user's preferences and makes suggestions to the user about the documents they may be interested in or ought to read. In addition, the user may request information on a certain topic and the training assistant is able to answer this request with personalised documents on the relevant subject matter.

3 Workflows for JITT

Figure 2 is a mockup of our JITT system with an intelligent conversational tutor interface called Justin.[1] At the top right an example workflow can be seen describing the process of claiming pay. In this workflow, a casual lecturer can elect to be paid fortnightly, in a single lump sum or by issuing an invoice from a company.

A number of documents may be associated with the "Claiming Pay" workflow. For example, the casual lecturer contract; a bank account declaration form; examples of completed versions of this form; tax details declaration form; University policy documents for casual academics etc. In Figure 2, Justin has suggested that the next logical step for the user is to fill in a personalised casual (pay) claim form which can be seen at bottom right.

4 Personalisation for JITT

Underpinning the personalisation in our JITT system is the model of each user. This needs to represent relevant aspects of their knowledge and learning preferences. In general, the design of a user model involves a tradeoff between the desirable levels of detail that might be useful for personalisation against the practicality of capturing all relevant information about the user.

The workflow foundation of the JITT system provides a systematic basis for designing the user model. Essentially, each workflow in the system can be regarded as one context for the modelling. Within that single workflow context, the elements of the workflow constitute elements of the user model. Once the elements of the workflow have been defined, it is possible to automatically generate the skeletal elements of the associated user model.

To make user modelling feasible, we need to ensure there are mechanisms for determining whether the user knows each of these elements. Natural sources of user modelling information follow from the JITT architecture. The most obvious is available if the system is able to determine that the user has accessed a document. This is one form of evidence that the user knows something about each workflow element that has a link to that document. Another core source of evidence follows when

[1] Virtual Charter Image, kindly reproduced with the permission of the Centre for Speech Technology, Sweden.

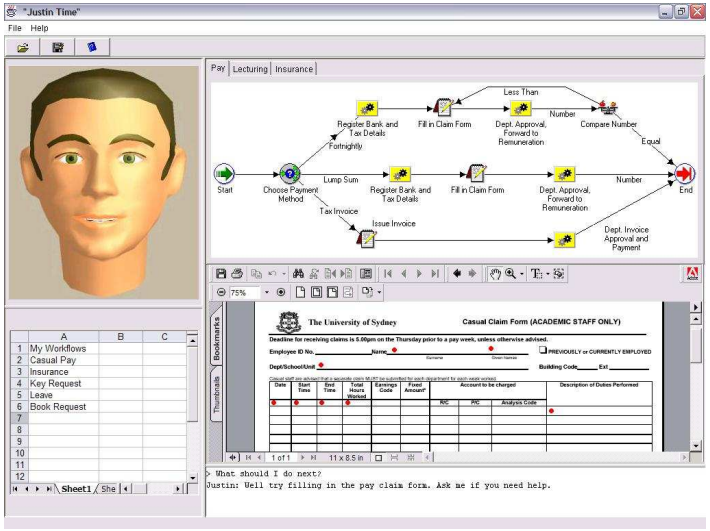


Figure 2: Mockup of "Justin", Our Intelligent Just-in-time Trainer.

the user succeeds in performing one cycle of the workflow. For example, if the system has access to information that the user has acquired keys, it can use this as strong evidence that the user knows all the critical components of the workflow. It can also serve as weaker evidence about other components. We collect evidence from sources such as these to reason about the user's knowledge and manage it in an evidence-based reasoning system.

The architecture has been designed with the expectation that each workflow within the system will be quite simple. This reflects that each individual process within this learning domain is quite small. The problem for the user is that there are many different processes in which they engage. We now illustrate this with an example scenario.

Sandra is a Professor of Computer Engineering and has just joined the University. She has several roles, including those of researcher in computer engineering, teacher of postgraduate and honours thesis students and teacher of an undergraduate course. These roles each need to be translated into a collection of workflows, one for each of the processes relevant to it. For example, her research activities mean that she needs to know about several processes, such as those for applying for university research grants, gaining internal funds for equipment, interacting with support staff when she has problems with equipment, managing university accounts for her existing grants and a diverse range of other processes such as those for organising conference travel.

Each process involves a quite small workflow and its small collection of associated documents. At any one point in time, Sandra has successfully moved to a point in each workflow, perhaps not yet having started some of workflows at all, or perhaps having partially completed others and fully completed yet others. JITT can select information that might be suitable for pushing to Sandra by making a list of all those documents associated with points just ahead of her current position on each of the workflows that are active for her. It is the task of the training assistant to select from these and decide on the delivery of documents.

For the pull process, the currently active workflows constitute a form of context for Sandra's understanding of organisational processes. This means that requests she makes to JITT can exploit this context in searching the document base. It can rate documents as more likely to be relevant to a query if they are just ahead of her current position on each of the workflows that are active for her. It can also rate documents as more likely to be relevant to a query if they are on workflows she has not started to date. The training assistant combines this assessment with other document retrieval techniques based on term matches.

5 Teaching Strategies

Teaching strategies are concerned with what to teach, how to teach and when.

What In traditional intelligent teaching systems, the ordering of learning topics, i.e. the curriculum, is either explicit or relies on the structure

of the knowledge being taught. However, in our approach the ordering is provided by the workflow. Each organisational process is represented by a workflow, with appropriate organisational documents attached to relevant points in each workflow. The advantage of workflows is that by their nature they sequence the steps through a particular process, making them naturally suited for suggesting the delivery of just-in-time documentation.

When Documents are suggested to the user in two ways: *pull* and *push*.

- The user may query the system to retrieve specific documents. By doing so, they may activate a new workflow process.
- The system identifies documents that are becoming relevant to the user according to the currently active workflows and pushes them to the user on a needs basis.

At any time several organisational processes are active for an employee, e.g. a teaching process, the process of writing a grant proposal, and so on. This means that several documents are relevant to the user at any given time. Our JITT system, behaving as an assistant, presents a list of documents to the user, sorted by several dynamic criteria (such as priority, associated workflow and due date).

We plan to explore a range of interfaces and modalities to support interaction with the user. The style of interface shown in Figure 2 enables us to explore the use of the workflow in helping the user appreciate the context. We also intend to explore a more conventional style of interface which is a combination of email and search-engine interfaces, with a browser search-engine interface for the pull mechanism. These have the advantage of being familiar to the user and offer space for additional text about each document, allowing the user to click through to the actual document. Notification mechanisms such as email can also be added.

How The level of recommendation varies according to the necessity of the document. Some documents are crucial, such as the pay claim form in our "Claiming Pay" example, while others are merely suggested, such as when the system cannot determine whether the knowledge contained in a document is already known to the user. Suppose the user returns to the "Claiming Pay" workflow a year later, would they want to download the claim form again?

Explanations help learners to understand. Workflows contribute in two different ways. First, they provide causal links between a sequence of actions and documents, in a way that users can easily understand. Users do not have the time to read a large number of documents, so knowing why they need to read a specific document, and how this document will help them do a certain task, is crucial. Second, they can be simulated: the user could decide to simulate the actions of a part of a workflow to have a clearer picture of the actions to come, documents to be read and can also identify potential pitfalls or overloads before they arise.

6 Conclusion

In this paper we have described an architecture for a just-in-time training system which exploits the structure afforded by workflow technology. In an organisation which has tools to manage its document collection and to capture workplace processes, the JITT architecture offers the possibility of additional leverage from the work invested in the workflow system so that this can be extended to support employee training.

The workflow also serves as a foundation for generating a user model systematically, at the precise level of granularity of the workflow itself. This means that effort invested in modelling the workflow would be reused to generate the structure of the user model for JITT.

The teaching strategy also benefits from the presence of workflow technology, allowing for the structured presentation of information and for the easy identification of relevant information. Additionally, the workflow allows for the simulation of a process, indicating documents that should be read and helping to identify problems before they occur.

References

[1] J. Lin, C. Ho, W. Sadiq and M. Orlowska. On workflow enabled e-learning services. In *Proceedings of the IEEE International Conference on Advanced Learning Technologies, ICELT'2001*, Madison, USA, August 2001.

[2] O. Marjanovic and M. Orlowska. Making flexible learning more flexible. In *IEEE International Workshop on Advanced Learning Technologies*, New Zealand, December 2000.

[3] S. Sadiq, W. Sadiq and M. Orlowska. Workflow driven e-learning: Beyond collaborative environments. In *Proceedings of the World Congress on Networked Learning in a global environment, Challenges and Solutions for Virtual Education*, Technical University of Berlin, Germany, May 2002.

[4] J. Sweller. Some cognitive processes and their consequences for the organisation and presentation of information. *Australian Journal of Psychology*, Volume 45, Number 1, pages 1–8, 1993.

Visual Displays for Browsing RDF Documents

Peter Eklund
School of Information Technology and Electrical Engineering
The University of Queensland
St. Lucia QLD 4072
peklund@itee.uq.edu.au

Introduction

Hyperbolic browsers are motivated by the "Circle Limit IV" woodcut of M.C. Escher. The hyperbolic tree view was introduced in graph drawing by Lamping and Rao [2] who observed that large structures could be compactly displayed by projecting a tree onto a hyperbolic plane. The effect of the projection is that components appear diminishing in size and radius exponentially the further they move from the centre of the diagram. The arguments for the hyperbolic display are twofold: an order of magnitude more nodes of a tree can be rendered in the same display space and the focus is maintained on the central vertex of the display and its immediate neighbourhood. The hyperbolic view is particularly useful for hierarchical diagrams with large numbers of leaves and branches and where neighbourhood relationships are meaningful. Examples of the hyperbolic view are INXIGHT's Star Tree[1] and HYPERPROF[2]. Given that the pure hyperbolic geometric projection is patented, projection onto a sphere, and diminishing radial layout views are the drawing approaches we have experimented with.

Recent interest in the Semantic Web has invigorated interest in programs that render ontological data sources (structured metadata) in the style of semantic networks well known in AI. The ONTOBROKER [1] project used a hyperbolic browser to view semantic relationships between frames in F-Logic. ONTORAMA traces its lineage to the ONTOBROKER[3] Java source code, but the code has been substantially re-written since 1999. The ONTORAMA effort involved refactoring the Java for the WEBKB-2 project [3, 4] which uses its own internal formats (conceptual graphs).

OntoRama and RDF

RDF is a widely used standard for knowledge exchange. Thus, by using RDF, RDF(S) and DAML+OIL as inputs, we broaden the application base for ONTORAMA. Further, access to a selection of RDF libraries encourages testing and generalising program operation to handle different input files. We are progressively adapting ONTORAMA to handle DAML+OIL compliant RDF(S) and reified RDF: the program is therefore still under development.

Some of the more obvious challenges for a general purpose RDF, RDF(S) and DAML+OIL browser are: (i) many RDF ontologies are not strictly hierarchical and include multiple inheritance; (ii) many properties may connect Resources and Properties so there is more than a single transitive edge type in the RDF graph; (iii) often objects are isolated and may not be reachable by following relation links from the top (root) item: not all RDF objects found by search are connected. An input file may therefore be a forest rather than a tree and the browser must be able to navigate across the individual components. (iv) Further, we need to be able to represent RDF Resources/Descriptions so that the visual representation of the documents does not loose any information. Less obviously, (v) RDF that is reified, predicates that have a situational content, must be dealt with in a visual way and (vi) there are also many syntactically equivalent forms of RDF. These create design issues for ONTORAMA since, not only is the RDF input syntax complex, but also the usual tree-view metaphor for the hyperbolic-like view breaksdown.

```
...
<rdfs:label xml:lang="en">artifact</rdfs:label>
<rdfs:label xml:lang="en">artefact</rdfs:label>
...
<dc:Creator>http://cogsci.princeton.edu/~wn/
</dc:Creator>
<rdfs:comment>a man-made object taken as a whole
</rdfs:comment>
<rdfs:subClassOf
 rdf:resource="http://wekbkb.org/pm#CreatedThing"/>
<rdfs:subClassOf
 rdf:resource="http://wekbkb.org/wn#PhysicalObject"/>
<rdfs:subClassOf
 rdf:resource="http://wekbkb.org/wn#WholeThing"/>
<daml:disjointWith
 rdf:resource="http://wekbkb.org/wn#NaturalObject"/>
...
```

RDF(S) objects may inherit properties in multiple ways, point (i) above; so some adaptation of the strict tree-structure browsing principles, particularly to the hyperbolic browsers view, needs to be addressed. Any RDF file can be considered an acyclic graph where a child can have multiple parents (as is the RDF shown above) where a single Artifact has four superclass parents, CreatedThing, PhysicalObject, WholeThing and NaturalObject.

[1] http://www.inxight.com
[2] http://www.physics.orst.edu/~bulatov/HyperProf/
[3] http://ontobroker.aifb.unikarlsruhe.de/index_ob.html

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

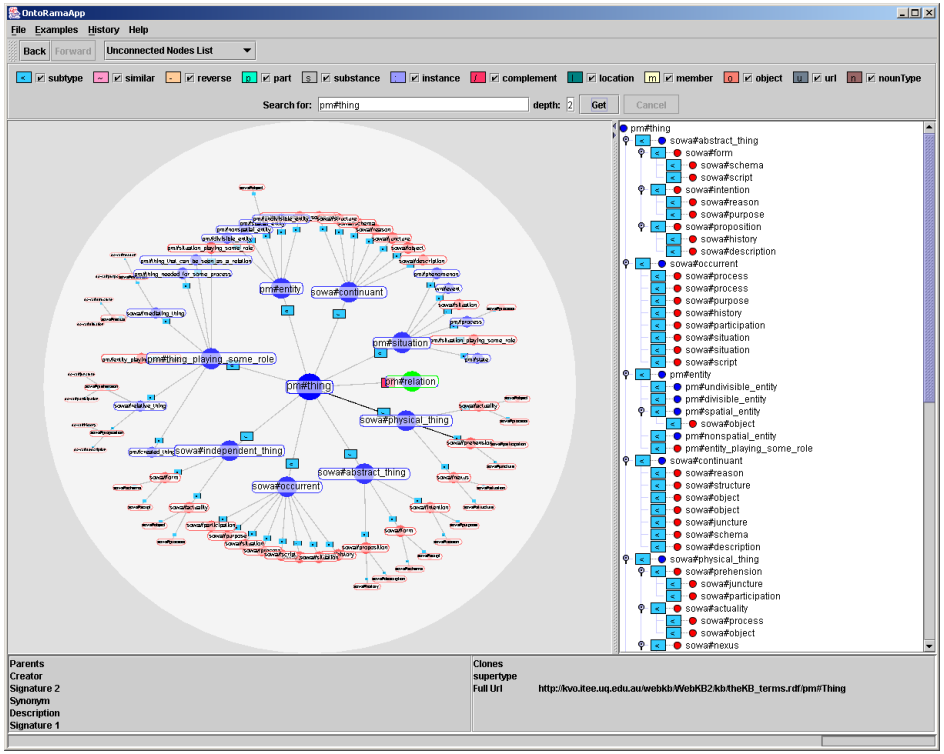


Figure 1. ONTORAMA: **The Property Hierarchy shows the subclass/superclass relations between relation types.**

The data structure must therefore conform to a tree and non-planar graphs transformed to this form. The solution is to copy entire sub-branches in the graph for vertices with multiple parents: producing a tree structure where vertices have only one incoming edge. Once a vertex (and its ancestors as a sub-tree) have been copied, each vertex can be drawn in the coordinate plane using either hyperbolic or radial layout algorithms. Copying vertices and sub-graphs in this way has a disadvantage, the size of the overall tree increases, reducing the effectiveness of the hyperbolic view and an argument for its use. In addition, the GUI needs to be designed to highlight copied vertices and sub-graphs, so that these replicated structures are easily identifiable. Copied vertices and sub-trees are rendered red to distinguish them in the interface and copies are connected to this ring to enable the user to identify and unify the copies, locating them in the view.

A disadvantage of using a tree metaphor for browsing RDF is that an RDF file may contain multiple trees (or simply objects and attributes that disconnected). Initially, ONTORAMA relied on the fact that all input items were interconnected directly (or indirectly) connected to a "top" ontology item. Overcoming this problem, to allow ONTORAMA to browse a *forest* rather than a *tree*, the various components of the tree are made accessible by a pull-down menu (top-left) called *Unconnected nodes list*, here the tree components are named after the root node name of each of the component trees. ONTORAMA must display two tree structures: one for the object or class hierarchy and the other for the property hierarchy. To illustrate, consider the following RDF at the top of page 4.

The solution to rendering property hierarchies is to introduce two graphical structures that interoperate. Color and shape are useful device to draw attention to the points where the interactions between the RDF Resource hierarchy and the Property hierarchy inter-connect. This is shown in Fig. 1, note the *green* Property Relation immediately to the right of Thing, the notation of WEBKB-2 indicates that these are exclusive types, namely that the Resource/Object hierarchy is exclusive of the Property/relation hierarchy.

We advocate multiple visual views over an ontology, and ONTORAMA includes both the hyperbolic-style view and a tree view widget. Other visual views, such as true 3D displays and

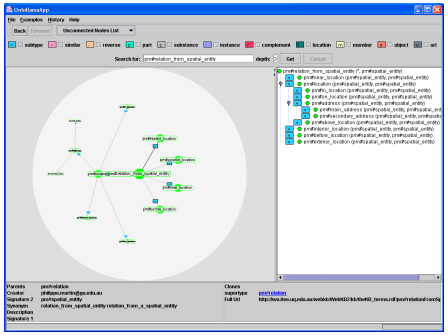


Figure 2. ONTORAMA: The Property Hierarchy shows the subclass/superclass relations between relatioon types. Note that the tree widget now displays the relational signature.

concept lattices, may also be appropriate for browsing ontological data. Most computer users are familiar with viewing files in a directory structure using a so called "tree widget". However, a tree view does not usually represent the idea of a file belonging to more than one sub-directory since the physical (or logical) structure of most file systems insists on a strict hierarchy.

Unlike the hyperbolic view, where a fixed envelope of space displays the children and there may be hundreds of children crowded into a small radial space, the tree view allows the structure to be drawn downwards with no limiting viewing space. Therefore, the advantage of the hyperbolic view is that users can understand the ontology structure by seeing the relational links between the vertices however this needs to be supplemented by a tree view explorer when there are large numbers of children.

ONTORAMA exploits the advantages of both views in different circumstances by allowing them to complement one another. Each view should allow the user to execute similar commands, the user can click on a vertex in the hyperbolic view and gain focus in both hyperbolic and file explorer view. The vertex is highlighted in both. Vertices in the hyperbolic view can be double clicked to drill-down or roll-up in a similar manner to collapsing the directory structure in a file explorer. When a node is folded in the hyperbolic view, it is represented by a square and the rest of the subbranch becomes invisible. Pruning branches in the hyperbolic view enables the user to adjust the focus and details of the display and enhance the cognitive effect of navigating the ontology with the GUI.

ONTORAMA can be configured to parse different RDF and XML inputs. The ONTORAMA configuration file allows the user to specify various relation links. Assigned to relation links are a symbol and colour, used in ONTORAMA to distinguish link types. These images are used to display the relation types. The user may select relations they wish to view by selecting the link check boxes (see Figure 1 (top)). Queries are then filtered to show the specified relation links.

The object type properties in ONTORAMA are fully configurable via the configuration XML file. This means if we need to display an ontology with different relation links, or different properties for each object type, we need only alter the configuration file without re-building the application. ONTORAMA reads the configuration file on start up. It consists of two sections: the `ontology` section and the `rdfMapping` section. The `ontology` section describes all details relevant to loading an ontology. The section includes a set of `relation` elements, each defining an ontology relation link. Similarly, we define concept type properties. Concept type properties are properties applicable to a concept type in the current ontology. For example, concept types have properties such as `type, description, creator, synonyms`, etc. Each relation is defined as a `relation` element is assigned an id. The id is a reference to the corresponding relation that ONTORAMA uses. We also define a name for each relation link. Some relations can have two names, which means that they inverse relations. An example of such relations are subtype/supertype. Consider the term `wn#dog`. This term is connected to the term `wn#canine` via relation link *subtype* (`wn#dog` is subtype of `wn#canine`). The inverse relation link to *subtype* is *supertype* and the following is a fragment from the configuration file describing the subtype/supertype relation:

```
<relation id="1">
 <relationType name="subtype" mappingSymbol="&lt;"/>
 <relationType name="supertype" mappingSymbol="&gt;"/>
 <display color="#33CCFF" symbol="&lt;"/>
</relation>
```

Each relation is identified by a relation id and name. Note that ONTORAMA will display only one relation link for each relation and the first declared link in the `relation` is displayed. Display properties for each relation link, such as color and symbol, are also included. Concept type properties are defined using the element `conceptProperty`. At present this element has only an id: a string describing this property. Once these elements are read by ONTORAMA, the application knows that it needs to create corresponding entries in the description panel.

```
<rdfMapping>
  <relationLinks>
    <map id="1" type="supertype" tag="subClassOf"/>
    <map id="3" type="part" tag="part"/>
    <map id="6" type="inclusive" tag="jointWith"/>
  </relationLinks>
  <conceptProperties>
    <map id="Description" tag="comment"/>
    <map id="Creator" tag="Creator"/>
    <map id="Synonym" tag="label"/>
  </conceptProperties>
</rdfMapping>
```

The `RdfMapping` section of the configuration file specifies how the application will be able to

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
         xmlns:dc="http://purl.org/metadata/dublin_core#"
         xmlns:daml="http://www.daml.org/2000/10/daml-ont#"
         xmlns:pm="http://wekbkb.org/pm#">

<rdf:Property rdf:about="http://www.daml.org/2000/10/daml-ont#equivalentTo">
  <rdfs:label xml:lang="en">equivalent_to</rdfs:label>
  <dc:Creator>http://www.daml.org/2001/03/daml+oil.daml</dc:Creator>
  <rdfs:subPropertyOf rdf:resource="http://wekbkb.org/pm#equal"/>
</rdf:Property>

<rdf:Property rdf:about="http://wekbkb.org/pm#equal">
 <rdfs:label xml:lang="en">equal</rdfs:label>
 <dc:Creator>philippe.martin@gu.edu.au</dc:Creator>
 <rdfs:comment>"=" in KIF</rdfs:comment>
 <rdfs:subPropertyOf rdf:resource="http://wekbkb.org/pm#equivalenceRelation"/>
 <daml:complementOf rdf:resource="http://wekbkb.org/pm#different"/>
</rdf:Property>

 <rdf:Property rdf:about= "http://wekbkb.org/pm#equivalenceRelation">
    <rdfs:label xml:lang="en">equivalence_relation</rdfs:label>
    <dc:Creator>philippe.martin@gu.edu.au</dc:Creator>
    <rdfs:comment>instance of equivalence_relation_class</rdfs:comment>
    <rdfs:subPropertyOf rdf:resource="http://wekbkb.org/pm#orderingRelation"/>
 </rdf:Property>

     <rdf:Property rdf:about="http://wekbkb.org/pm#orderingRelation">
       <rdfs:label xml:lang="en">ordering_relation</rdfs:label>
       <dc:Creator>philippe.martin@gu.edu.au</dc:Creator>
       <rdfs:comment> e.g. pm#kind, rdfs#sub_class_of, pm#part, pm#equal</rdfs:comment>
       <rdfs:subPropertyOf rdf:resource="http://wekbkb.org/pm#relation"/>
     </rdf:Property>

        <rdf:Property rdf:about="http://wekbkb.org/pm#relation">
          <rdfs:label xml:lang="en">relation</rdfs:label>
          <rdfs:label xml:lang="en">related_with</rdfs:label>
          <dc:Creator>philippe.martin@gu.edu.au</dc:Creator>
          <rdfs:comment>type for any relation (unary, binary, ...) </rdfs:comment>
          <rdf:type rdf:resource= "http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
          <daml:complementOf rdf:resource="http://wekbkb.org/pm#Thing"/>
        </rdf:Property>
</rdf:RDF>
```

find items defined in the ontology section in the source RDF file. The preceding example illustrates an `rdfMapping` element. `rdfMapping` consists of two subsubsections: `relationLinks` maps relation links defined in the ontology section to RDF tags. `conceptProperties` maps concept type properties to RDF tags. For instance, the element `<map id="1" type="supertype" tag="subClassOf"/>` is a mapping relation type *supertype* to the RDF tag *subClassOf*. This is an interesting example because (as stated previously), ONTORAMA displays the relation link *subtype*, and the application has to be told of the inverse *supertype* links to be able to display them. The following illustrates;

The last statement is translated by ONTORAMA as: `wn#TrueCat` is *supertype* of `wn#Cat`. However, ONTORAMA needs to reverse this statement as only *subtype* relation links are displayed resulting in: `wn#Cat` is the *subtype* of `wn#TrueCat`.

Conclusion

ONTORAMA has emerged as a practical tool for viewing ontologies described in RDF using Java Swing. We describe a solution to hyperbolic-style layout that is complicated by the prevalence of multiple inheritance in most ontological structures. We further describe extensions to the usual hyperbolic-style view allowing multi-link types to be rendered and filtered. Issues of application generality, configuration files for the application in XML, and the correspondence between behaviours in the application and RDF descriptions are discussed. ONTORAMA can be trialed at http://www.ontorama.org.

References

[1] Fensel, D., S. Decker, M. Erdmann, and R. Studer, "Ontobroker: Or How to Enable Intelligent Access to the WWW," *Proc. 11th Knowledge Acquisition Workshop* (KAW98), Banff, Canada, April 1998, pp. 8–23.

[2] Lamping, J. and Ramana Rao: The Hyperbolic Browser: A Focus+Context Technique for Visualizing Large Hierarchies, *Proceedings of CHI'95, ACM Conference on Human Factors in Computing Systems*, New York, pp. 401-408, 1995.

[3] Martin, P. and P. Eklund: Embedding Knowledge in Web Documents, The Eighth International World Wide Web Conference, (WWW8), pp. 324–341, Elsevier, 1999.

[4] Martin, P. and P. Eklund: Knowledge Indexation and Retrieval and the Word Wide Web, *IEEE Intelligent Systems* — Special Issue on "Knowledge Management and Knowledge Distribution over the Internet, July, pp. 18-25, 2000.

Tibianna: A Learning-Based Search Engine with Query Refinement

Clint Heyer *Joachim Diederich*
Information Environments Program
School of Information Technology and Electrical Engineering
The University of Queensland,
Brisbane, 4072 Australia
clint@thestaticvoid.net *joachimd@itee.uq.edu.au*

Abstract

While web search engine technology has improved over time, there is often a fundamental reliance on keyword matching for searches. What happens however, when the user does not know what keywords to use?

This paper presents preliminary learning results of a prototype learning search engine that attempts to address this problem. Tibianna allows a user to manually rank a set of results based on their own relevancy function. Once a required number of results are ranked, the set is downloaded, processed and presented to support vector machines (SVMs) for learning. Once learned, Tibianna can actively reorder or discard search engine results based on the model it has learned. This provides a way of improving search results without requiring query refinement. Learning outcomes from experimental trials with Tibianna are presented, demonstrating the implications of using different preprocessing techniques and corpus sizes.

Query refinement functions are also available to the user, which can enable exploration of query words via the WordNet database, and allows quick query refinement via a dynamic HTML interface.

Keywords Information Retrieval, Personalised Documents, Search Engine Technology

1 Introduction

This paper attempts to address an inherent problem with keyword-based search engines – what happens if the user doesn't know what keywords to use? This situation can present itself when the user is searching new domains; when the user is uncertain about what to search for or when they do not know what keywords to use to get the desired results. There is a divide between the user's search *intent* – their mind view of what kind of results they wish to find - and the effectiveness of the query they craft to match their intent. How effectively a user can craft their query is largely dependant on their knowledge of the search domain and proficiency with the search engine used.

The prototype implementation, named *Tibianna*, presents the user with query refinement options by using contextual and semantic data about the user's query, gathered from WordNet[1]. Supervised machine learning is used by the system in order to provide the user with results that more closely match their search intent without necessitating modification of the search query. Learning what the user's information needs are is a powerful technique; and has the possibility of increasing the effectiveness of a search engine. This latter method will be the primary focus for this paper, and indeed the system itself.

By providing the user with more tools for refining their search, the system can better capture what the user values. This value metric varies from query to query and user to user, even though keywords used are identical[1, 2].

Support vector machines (SVMs)[3] have been shown to be highly effective in text classification problems, primarily due to their ability to handle large dimensionality and a large number of features[4].

Tibianna takes the form of a traditional search engine, with interface and result-formatting similar to Google. The user interacts via a web browser, with Tibianna residing on either a remote server or the local machine. In preliminary experiments, Tibianna utilised Google as its underlying provider of search results. After the user enters a query, each result is displayed with a drop-down box allowing the user to rank its relevancy, after either viewing the page or by reading the page summary.

Figure 1: Initial search result page

After ranking a number of results the user can opt to send the rankings back to Tibianna. Tibianna then downloads each result and processes it, displaying the next page of results to the user. Once a requisite number of results are ranked by the user, Tibianna switches to classification mode, and reorders Google results according to the model it has learned.

Section 2 examines how features are extracted and processed from web documents in Tibianna, with Section 3 introducing the query refinement options briefly, and showing screen shots of the features. Section 4 details parameter experiments, with the results presented in Section 5. Section 6 discusses

[1] http://www.cogsci.princeton.edu/~wn/

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

future work with conclusions drawn in Section 7. More background on Tibianna is presented in [5].

2 Feature Extraction

Resources in Tibianna undergo a series of transformations before they can be presented to the SVM for learning. Firstly, HTML mark-up and any other special symbols that do not form part of the content are removed. The obvious side-effect of this stage is that any directly or indirectly encoded metadata is discarded, and cannot influence the learning result. Other work such as [2, 6] maintains this data to add weighting to subsections of content.

In the next stage, a stop list is used to remove the most 300 commonly used words in the English language. Words remaining from this stage are then stemmed according to the Porter algorithm[7]. Stop word removal takes out a number of features that have little bearing on the meaning of a resource and stemming reduces the number of features by converting plural and tense variations of a word to a common stem. N-grams (up to 5-grams are used in Tibianna) are then created based on remaining words.

Finally, features extracted from the entire corpus are brought together to be counted and weighed. Inverse document frequency (IDF) weighing is used to reduce the weight of commonly occurring features. The output of this stage is a training file used by SVM^{light} [8] to create a model from the data. This model is later used in the classification stage, after resources have gone through the same pre-processing stages as the training resources. Tibianna uses SVM ranking classification[6], as opposed to the more traditional binary classification.

3 Query Refinement

To assist the user in refining their search, query refinement options are available. These options are of particular value when the user is searching in a new domain, and makes use of the lexical database, WordNet. WordNet has been used in other work for query refinement, such as [9, 10].

Search queries are split into individual words and cross-referenced with WordNet. Data such as senses, synonyms, hyponyms, holonyms and meronyms for each word is presented to the user in a dynamic HTML interface within the search results. Client-side scripting is used for the interface to allow the user to explore the refinement options quickly, without the extra roundtrip to the server. Usage examples of query refinement using the query "*java*" are shown in Figures 2-4.

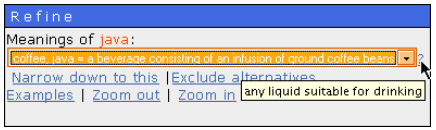


Figure 2: Each query word has a 'Refine' box with lets the user explore the word, and modify their query. The '?' text provides a tooltip for the meaning of a sense.

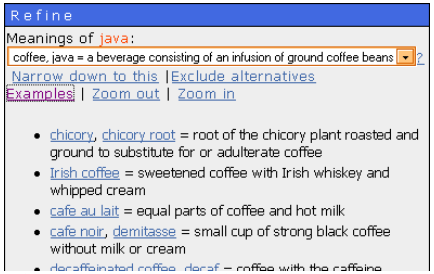


Figure 3: A user can see a list of examples of a sense meaning. Each example is hyperlinked and adds the example to the query when clicked.

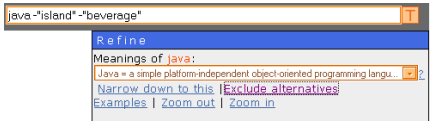


Figure 4: Clicking 'Exclude alternatives' attempts to reduce result set size by explicitly excluding alternative senses. In this example, the original *"java code"* query is refined to *"java code –'island' –'beverage'"*

4 Experimental Methodology

Many of the pre-processing stages have parameters that need to be determined for optimal running of the system. To discover these parameters, various experiments were run with real-world data sets. The training document set of 28 pages (see discussion below on the size of the training set) were sourced from the first 30 results for the Google search of *'java'*. Each result was hand-ranked on a 0 to 10 scale, with 10 being the highest. Another set of 11 pages were sought, coming from different sites than those used for training data. Some of these pages were on-topic, others completely irrelevant. These test pages were again hand-ranked using the same scale as the training pages so the classification results could be compared against a consistent benchmark. The rank frequency of the training documents is shown in Table 1.

Rank	*Frequency*
0	5
1	7
2	7
3	2
4	1
5	1
6	1
7	1
8	0
9	0
10	3
	28

Table 1: Rank frequency distribution for training documents

Note the unbalanced frequency as it makes learning more difficult for the SVM, but in turn is a more realistic test case: the search engine does not return many relevant results - exactly the problem Tibianna is designed to address.

To determine the effectiveness of the classification results consistently, a simple scoring algorithm was created. The score of a ranking classifier output is more difficult to determine than the traditional binary class based classifier output because a ranking is more complex, being multidimensional. The algorithm maximises the score for orderings that most closely match the target ordering. Relevant attributes are: (1) ranks that are in exact sequence (even though possibly incorrectly ranked), (2) distances between actual and target ranks, and (3) the number of ranks which are "roughly" in sequence (see below). An exact sequence is defined as an ordering such as *2, 3, 4, 5* in which ranks are in exact order, but may be offset from the correct rank. A "rough" sequence is defined as a series of ranks which maintain the correct ordering relationships, but may have one or more missing ranks, for example: *1, 3, 4, 6*.

A comparison between scored output and leave-one-out (LOO) SVM error rates may serve to highlight the differences, where the target ordering is *0...10*:

Ordering produced	*LOO Error*	*Score*
1 6 3 2 4 8 0 5 9 7 10	5.59%	18
1 2 3 6 4 8 5 0 7 9 10	6.83%	31
1 2 3 6 4 8 5 0 7 9 10	10.56%	31

Table 2: An illustration of the differences in leave-one-out (LOO) error and score.

As shown in Table 2, the score is determined by how well the SVM ranked a test set, and may or may not be indicative of a good learning result.

5 Experimentation

5.1 The effect of pre-processing options and transformations

In this trial, a static training and test document set was used, with feature extraction parameters systematically modified. There are several interesting conclusions that can be drawn from this trial, which are graphed in Figure 5.

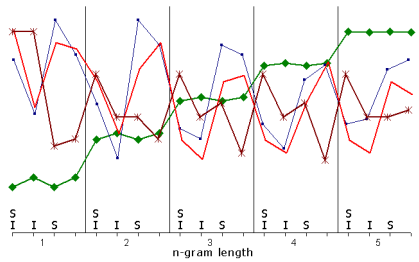


Figure 5: Parameter tuning results. The major horizontal axis is the n-gram length. The minor horizontal axis represents whether stemming and IDF weighing was used; just IDF was used; just stemming was used 'S'; or neither, denoted 'SI', 'I', 'S' and '' respectively.

Most importantly, it is apparent that the learning error rate (leave-one-out error rate, indicated by unmarked line) drops significantly when resources are pre-processed using inverse document frequency (IDF) to weigh features. Using both IDF weighing and word stemming was not as efficient however.

The number of support vectors (SVs, shown with a square-marked line) required for learning closely tracks the error rate. This is an indication that as a result of a difficult to learn problem, the number of SVs required to represent the problem grows. As the n-gram count grows (and with it, feature counts – denoted in diamond-marked line), the maximum bound for the number of SVs and the error rate appears to decrease; although the lower bounds remain relatively stable.

If we look at the score (defined in section 4, shown as a star-marked line) which has been calculated for each test, we can see that that it has a roughly inverse relationship to the error rate. This lends weight to the accuracy of the scoring algorithm, as a high error should result in a low score. Interestingly, when the error rate is in troughs (in the IDF processing column for each n-gram group), the score drops to a median level when it should be highest. Over the period of 2-5 grams, a clear pattern emerges, whereby the score is highest when both stemming and IDF-based weighing are utilised, and drops to its lowest when neither are used. While the latter point is consistent with the error rate, the first is not.

5.2 Corpus Size

Exactly how many documents Tibianna requires the user to rank before it can accurately begin to classify results was an open question at the beginning of this work. A balance needs to be sought between how much time and effort is expected of the user, and how many documents are required for classification. Figure 6 plots the results of this trial.

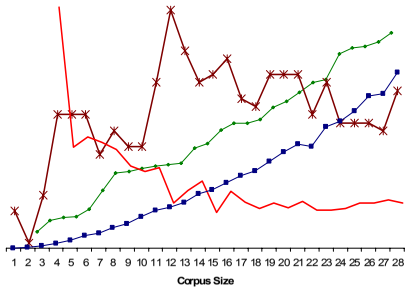


Figure 6: Corpus size trial results

With the rising number of documents (horizontal axis) feature counts and support vector quantities (diamond-marked and square-marked lines respectively) also rise, as expected. The leave-one-out error rate (unmarked line) appears to follow a decay function, dropping very rapidly when the corpus size is low, and remaining stable once the corpus size exceeds 20.

From repeated runs of this trial (with random orderings each time, and with different datasets), it is evident that the error rate does not decrease significantly past 15 documents, and in practise could be set as low as 12. While training on a larger document set can often improve accuracy of learning, the cost is increased processing time and more time and effort required of the user. In general usage by the authors, it was found that training on 15 documents was acceptable in both learning result and user effort terms. The perceived accuracy of the learning result does depend on the number of positive training cases, and the use of a consistent ranking method by the user.

6 Future Work

Future analysis and experimentation could improve the final system. How stop-word list size and contents (e.g. using most common words on web rather than in general language usage) affects learning is something not investigated in this paper, but would be beneficial. While the corpus size was intentionally kept small to simulate real-world usage, an investigation into how Tibianna scales using larger corpus sizes would be beneficial. Other work to be done is to investigate the scoring algorithm more exhaustively, experiment with feature reduction methods, and examine how the frequency of differently ranked results alters learning.

7 Conclusion

Tibianna is a unique system which addresses shortcomings in traditional keyword-based search engines. Primarily, how the user is to find what they desire without knowing the best keywords to use, and how the divide between query terms and the user's search intent be minimised. Tibianna offers users the ability to rank search results based on their own relevancy function. After the user has trained the system on a minimum number of results, Tibianna learns the user's relevancy function with a high degree of accuracy. This learned function is then applied to subsequent search results by reordering results according to learned model.

Assisted query refinement via a client-side HTML interface is used in Tibianna to improve the precision of returned results. By selecting terms from the interface, the user can dynamically refine their search by adding terms to the query.

In general usage by the authors, Tibianna has shown to be highly valuable. With the combined SVM learning and query refinement, Tibianna is an effective system for improving existing search engines.

8 References

[1] E. Glover, S. Lawrence, G. Michael, W. Birmingham, and C. L. Giles, "Web Search - Your Way," *Communications of the ACM*, 1999.

[2] E. Glover, G. Flake, S. Lawrence, W. Birmingham, A. Kruger, C. L. Giles, and D. Pennock, "Improving Category Specific Web Search by Learning Query Modifications," presented at Symposium on Applications and the Internet, SAINT, San Diego, CA, 2001.

[3] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning*, vol. 20(3), pp. 273-297, 1995.

[4] T. Joachims, "Text Categorization with Support Vector Machines: Learning With Many Relevant Features," presented at Proceedings of ECML-98, 10th European Conference on Machine Learning, Heidelberg, Germany, 1998, pp. 137-142.

[5] C. Heyer, K. Hollingsworth, J. Madden, P. Heydon, K. Bartlett, and J. Diederich, "MyNewsWave: User-centered Web search and news delivery," (to appear) 7th Australasian Document Computing Symposium, Sydney, Australia, 2002.

[6] T. Joachims, "Optimizing Search Engines Using Clickthrough Data," presented at ACM Conference on Knowledge Discovery and Data Mining, 2002.

[7] M. Porter, "An Algorithm for Suffix Stripping," in *Readings in Information Retrieval*, vol. 14. San Francisco: Morgan Kaufmann, 1997, pp. 130-137.

[8] T. Joachims, *Making Large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning*: MIT-Press, 1999.

[9] P. D. Bruza, R. McArthur, and S. Dennis, "Interactive Internet Search: Keyword, Directory and Query Reformulation Mechanisms Compared," presented at 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2000.

[10] D. Moldovan and R. Mihalcea, "Using WordNet and Lexical Operators to Improve Internet Searches," *IEEE Internet Computing*, vol. 4(1), pp. 34-43, 2000.

MyNewsWave: User-centered Web search and news delivery

Clint Heyer, Jamie Madden, Kelly Hollingsworth,
Peter Heydon, Keiran Bartlett, Joachim Diederich

Information Environments Program
School of Information Technology and Electrical Engineering
The University of Queensland
Brisbane Q 4072, Australia

clint@thestaticvoid.net *kelly@infenv.com*
jamie@itee.uq.edu.au *heydonp@acslink.net.au*
keiranbartlett@keiranb.cjb.net *joachimd@itee.uq.edu.au*

Abstract

MyNewsWave uses machine learning (including support vector machines) for a user-centred approach to full-text information retrieval as well as news delivery. The system uses knowledge sources such as WordNet to refine keyword queries and learns user-preferences with regard to web search. MyNewsWave includes an audio mining system for topic detection in conjunction with background search to facilitate the retrieval of relevant multimedia information.

A special feature of MyNewsWave is the assessment of incoming information with regard to the "mood" or personal relevance to a user. DigiMood is a component of MyNewsWave that classifies web pages into mood categories. Business news, for instance, can be classified by DigiMood to access market sentiment. Marconi analyses incoming news streams and uses machine learning to adjust parameters of a text-to-speech system. The objective is to learn the appropriate voice for news items as part of a speech user interface.

Keywords Multimedia resource discovery, Personalised documents, information retrieval.

1 Overview

We aim at integrating information search and delivery by use of machine learning systems that allow adaption to an individual user. MyNewsWave has five parts: (1) The *Tibianna* search engine uses support vector machines (SVMs) for learning ranking functions utilising user feedback and ontological knowledge. (2) *DigiMood* assesses web pages with regard to the mood expressed in the document. (3) *Peeping Tom* is a topic classification and user modelling system that classifies documents into categories relevant to the user. (4) The delievery component *Marconi* provides a *speech user interface* that learns to select voices based on user preferences. (5) The *Emily* audio mining system performs background search and analyses audio files to retrieve additional information.

Our aim is to combine active web search, including background research for related audio and textual information, with a user-centered approach to information delivery. Most components of MyNewsWave are implemented and are currently being tested. A user interface similar to a standard web browser integrates the subsystems (Figure 1).

A full introduction of all five MyNewsWave components is beyond the scope of this paper. Hence, the focus here is on text mining by use of machine learning techniques as used in the Tibianna and DigiMood subsystems.

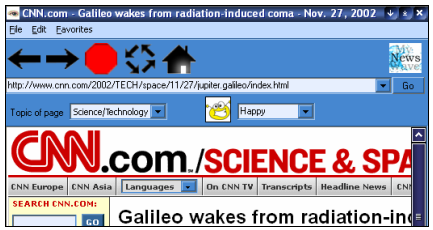


Figure 1: The MyNewsWave browser.

2 Introduction: Web search and machine learning

The process by which a user can refine a search is an emerging research field with many approaches and techniques. Tomita & Kikui [1] use graphical query refinement, whereby a *query graph* is created from

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

the user's search query. For each search result, a *subject graph* is created, and its similarity to the query graph determines the ranking. The user hones their search by directly manipulating the query graph as well as indicating if a returned document is relevant.

Most search engines use a global relevance ranking which is linked to the query and does not take into account the users' subjective *value* of a resource [2, 3]. This value metric is not necessarily encapsulated in search queries and documents returned from identical queries by two users may have entirely different values. Better capturing this value metric should increase the precision of the search.

Glover [2] uses an 'information need' query modifier, which refines search queries to return resources of a certain type, for example research papers, or personal home pages. This modifier allows the engine to extract more qualifiers for the search, without the user having to think about structuring the query, therefore further improving precision.

Bruza & McArthur [4] take a user-centred approach by empirically comparing various methods for query modification: standard searching, phrase-based query reformulation and hierarchical directory browsing. One of the query reformulation methods investigated takes advantages of *WordNet*[1], a large lexical database. Moldovan & Mihalcea [5] further investigate using WordNet and describe various algorithms for using an original user's query and WordNet reference data to restructure a user's search query, resulting in significant improvement in performance.

A metasearch engine is a search tool that combines the results of one or more external search engines, applying its own ranking function and then presenting the hybrid results to the user [6]. Whereas some metasearchers rely only on the results pages from the source search engines to form the meta result set, Lawrence & Giles [7] present a metasearch engine, *NECI*, which downloads the top ranked pages from each source, and performs its own analysis. This extra layer of analysis can help to weigh differences in page ranking algorithms used by the various engines, and provides a final, consistent ranking for all engines. NECI also transforms particular user queries into a style that is more likely to be present in a web page. For example, transforming "What does NASDAQ stand for?" into "NASDAQ stands for", "NASDAQ is an abbreviation" and "NASDAQ means". These queries are searched in parallel with the user's original search, with the combined results shown to the user.

[1] http://www.cogsci.princeton.edu/~wn/

2.1 Web search and support vector machines

Machine learning can be used to improve search results. Cortes & Vapnik [8] introduce support vector machines which are a novel approach to machine learning. Support vector machines are based on the structural risk minimisation principle. Support vector machines find the hypotheses out of the hypothesis space *H* of a learning system which approximately minimises the bound on the actual error by controlling the VC-dimension of *H*. SVMs are very universal learning systems [9]. In their basic form, SVMs learn linear threshold functions. However, it is possible to "plug-in" kernel functions so that they can be used to learn polynomial classifiers, radial basis function (RBF) networks and three or more layered neural networks.

The most important property of SVMs for text classification is that learning is independent of the dimensionality of the feature space [9]. SVMs evaluate hypothesis by use of the margin they use for separating data points, not the number of features or attributes. This allows for good generalisation even in the presence of a large number of features. Joachims [10] lists the following reasons why SVMs are a preferred method for learning text classifiers:

(1) SVMs can process high-dimensional input spaces: If every word of a text is a feature, the input space can easily be larger than 100,000. SVMs control overfitting internally and, therefore, large feature spaces are possible.

(2) Few irrelevant features: Feature selection is normally used to avoid input spaces of high dimensionality. In text classification, this is either not practical or many features are equally important. Therefore, SVMs are a convenient way to learn a text classifier with limited preprocessing.

(3) Document vectors are sparse: For the reasons mentioned above, SVMs are ideally suited for sparse input vectors of high dimensionality.

(4) Most text categorisation problems are linearly separable: This has been empirically determined by a number of authors.

Glover et al. [11] describe a system that uses SVMs in conjunction with learned query refinement to increase search relevancy. This approach looks at categorising resources into groups such as papers, research papers and product reviews. Their SVM is trained on the top 100 features from each resource, taking into account HTML mark-up (for example, words appearing in the title of the page were weighted higher than those in -2 size font) as well as positioning of terms within the document.

A more complex system is proposed in [12]. SVMs are employed to screen out irrelevant resources. Bayesian networks are then used to learn

regular expressions to filter documents for relevant blocks of text. Kwok [9] used SVMs in an automated manner to categorise the Reuters corpus to much success, even with minimal preprocessing.

Most relevant to the work presented in this paper is that of Joachims [13] in which SVMs produce a *ranking* rather than the usual binary positive/negative decision. This new development in SVMs allows the re-ordering to be much more specific; ranking results according to their actual relevancy.

3 Machine learning to complement traditional keyword-based search: The *Tibianna* search engine

Keyword-based search engines rely on good metadata derived from content or supplied by a human indexer. If the right keywords are not known (for example, when searching a topic area that is completely new to the person), results are inferior compared to a search with a perfect combination of keywords. *Tibianna*, a new search engine embedded in MyNewsWave, gives better accuracy to these 'fuzzy' searches by way of (1) reordering existing search engine results, (2) gathered result relevancy user feedback and (3) provision of ontology-based mechanisms provide query refinement functions.

Tibianna is particularly suited for multimedia content, where exact keywords or metadata for the resource is often hard to quantify (as opposed to a document, where the search can be performed on the document itself). Of course, any metadata that is available for a resource should still play a large part in determining search result rankings[2].

In more detail, *Tibianna* uses search session history (i.e, what the user has searched for previously in this session), and ontological data to help the user to refine search. The system works as follows:

- The user starts a web search, the server keeps track of a session.
- Where possible, the server adds lexical refinement options to provide more context for the user. Sources for this data are databases such as WordNet. Ontologies can be used for semantic or lexical disambiguation (e.g. 'Java' which is a drink, an island and a programming language) by allowing the user to select the relevant meaning.
- After viewing a result link, or result summary, the user has the option to rate the 'fitness' of the page, according to their own relevancy function.
- As the user progresses through the search, the SVM learns progressively more about the search intent of the user. After a reasonable number of results have been ranked, Tibianna begins to reorder search results based on what it has learned. The user can also delve deeper into the search by considering the lexical and semantic data that is presented.

[2] A potential use of Tibianna is the correction of deliberately misleading metadata.

4 DigiMood: Classifying web pages based on emotional content

For a prediction of the potential impact of a news item, an assessment of the "emotional" content of an article can be as valuable as a ranking with regard to personal relevance. DigiMood assesses the mood of any web page. An iconic representation is displayed in the browser once the mood has been established.

The learning component predicts the mood of the web page after an initial SVM learning period. The user classifies web pages during this phase, teaching the SVM to match the user's own mood categories. The number of web pages needed for the learning will be determined based on information gathered from a testing phase, and also informed by experimentation conducted in [14].

DigiMood takes the form of a web browser plug-in component. When a web page is loaded, the user selects the mood that best describes the page. During the learning period, the URL, page content (stripped of HTML encoding) and emotional state selected are appended to an XML file. Learning starts once a sufficient number of documents are available for training by SVM^{light} [9]. Once SVM training has completed, DigiMood commences mood classification of pages viewed by the user. The user can adjust the predicted emotion if necessary, thereby providing feedback for further learning periods.

5 Searching for multi-modal background information: The Emily audio mining system.

Topic Detection and Tracking (TDT) refers to computerised techniques for finding topically related material in streams of data of various type (audio, video, text, image etc.). Hence, TDT is multi-modal by definition. Emily includes a method for automatically extracting content from speech so that MyNewsWave approaches TDT functionality. Like the other components of MyNewsWave, the method utilises machine learning.

Emily has two parts: (1) "background" searching on a topic, i.e. the engine will gather related, multi-modal information and (2) topic detection by use of audio data. Standard speech recognition is used to generate a transcript which is then input to a machine learning system that performs topic categorisation.

The learning component of Emily is based on the audio data processed during the input stage of the system. It is assumed that there will be at least a 50% error rate in the transcription from audio to text. The outcome of the learning process is to decide whether a specific piece of audio belongs to a topical category. The newly categorised transcription can then be used

as additional input to the background search. Transcripts are also added to the original input document in an attempt to help with retrieving documents within the correct category.

Background searches take the topic classifications by Emily of audio or web HTML resources to construct search queries. After determining the topic(s) for a resource, Emily queries WordNet for related terms that are then assembled into Google queries. Several of these searches are run, with the union of the results presented to the user in brief form as a side bar.

6 Conclusions

MyNewsWave can support journalists, editors and other knowledge workers by providing a range of web search facilities. MyNewsWave ranks search results according to personal preferences and allows for the classification of multimedia documents into topic categories. Furthermore, web pages can be assessed with regard to the mood they express, and even if the user is away from a machine, a speech user interface allows communication.

Acknowledgements

Special thanks to Harald Schuetz from *Deutsche Welle* (German Foreign Broadcasting Cooperation) who guided this research by providing valuable feedback. Thank you also to Stephen Viller for his comments.

References

[1] J. Tomita and G. Kikui, "Interactive Web search by graphical query refinement," presented at 10th World-Wide Web Conference, Hong Kong, 2001.

[2] E. Glover, S. Lawrence, G. Michael, W. Birmingham, and C. L. Giles, "Web Search - Your Way," *Communications of the ACM*, vol. 44(12), pp. 97-102, 2001.

[3] S. Lawrence, "Context in Web Search," *IEEE Data Engineering Bulletin*, vol. 23(3), pp. 25-32, 2000.

[4] P. D. Bruza, R. McArthur, and S. Dennis, "Interactive Internet Search: Keyword, Directory and Query Reformulation Mechanisms Compared," presented at 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2000.

[5] D. Moldovan and R. Mihalcea, "Using WordNet and Lexical Operators to Improve Internet Searches," *IEEE Internet Computing*, vol. 4(1), pp. 34-43, 2000.

[6] E. Selberg and O. Etzioni, "The MetaCrawler Architecture for Resource Aggregation on the Web," *IEEE Expert*, vol. Jan-Feb, pp. 11-14, 1997.

[7] S. Lawrence and C. L. Giles, "Context and Page Analysis for Improved Web Search," *IEEE Internet Computing*, vol. 2(4), pp. 38-46, 1998.

[8] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning*, vol. 20(3), pp. 273-297, 1995.

[9] T. Joachims, *Making Large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning*: MIT-Press, 1999.

[10] T. Joachims, "Text Categorization with Support Vector Machines: Learning With Many Relevant Features," presented at Proceedings of ECML-98, 10th European Conference on Machine Learning, Heidelberg, Germany, 1998, pp. 137-142.

[11] E. Glover, G. Flake, S. Lawrence, W. Birmingham, A. Kruger, C. L. Giles, and D. Pennock, "Improving Category Specific Web Search by Learning Query Modifications," presented at Symposium on Applications and the Internet, SAINT, San Diego, CA, 2001.

[12] A. Kruger, C. L. Giles, F. Coetzee, E. Glover, G. Flake, S. Lawrence, and C. Omlin, "DEADLINER: Building a New Niche Search Engine," presented at Conference on Information and Knowledge Management, Washington, DC, 2000.

[13] T. Joachims, "Optimizing Search Engines Using Clickthrough Data," presented at ACM Conference on Knowledge Discovery and Data Mining, 2002.

[14] C. Heyer and J. Diederich, "Tibianna: A Learning-Based Search Engine with Query Refinement," (to appear) 7th Annual Australasian Document Computing Symposium, Sydney, Australia, 2002.

Visualisation of Document and Concept Spaces

Sam Holden, Judy Kay, Andrew Lum

School of Information Technologies
University of Sydney, NSW 2006
Australia

E-mail {sholden, judy, alum}@it.usyd.edu.au

Abstract

Collections of documents with conceptual relationships exist in many domains. Teaching systems often contain numerous learning resource documents. University policies are often large collections of related documents. The visualisation of the structure of these collections can be useful as it allows the exploration of the collection.

This paper describes a graphical interface for visualising document spaces. The interface makes it simple for the user to explore the documents and the relationships between them.

Keywords metadata, ITS, ontology extraction, user modelling, visualisation

1 Introduction

There are large numbers of existing document collections which are valuable resources, but are often difficult to explore. From university course materials on the Internet, to policy documents on a corporate intranet.

In the C++ STL domain, for example, there are numerous online resources, from textbooks [3], to short tutorials [5], to programmer reference manuals [8, 7]. The value of these resources to a learner is limited by the difficulty in finding the ones that meet the current learning need, and do not rely on material the learner does not know.

In order to make these documents more useful metadata is required. General metadata such as LOM [2] and Dublin Core [1], provides information about the author, type, subject area, etc.

When documents are used in specialised contexts, such as learning resources in a course, extra metadata is useful. This metadata indicates prerequisite and outcome information for the documents. With the availability of such metadata, a learner can find out which documents teach the concepts they wish to learn. And also determine which documents are available due to the learner understanding all the prerequisites.

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

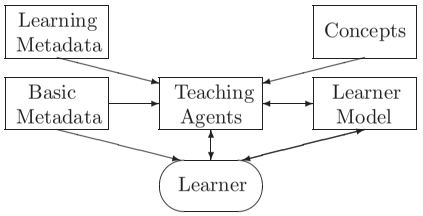


Figure 1: SITS Component Interaction

If the learner does not understand the prerequisites for a document they wish to use, they must learn those prerequisites. This involves viewing documents which teach those prerequisite concepts. If there are yet more prerequisites which are needed then the process will continue in a recursive manner.

Doing this manually would be complicated and tedious. Instead, a teaching system should be able to assist the learner. This paper describes a graphical interface for helping a learner explore the document space.

The teaching system used to host the interface will be briefly described. Then the visualisation interface will be presented, with a report on a user evaluation. Finally the benefits and some possible improvements to the interface will be discussed. We summarise the current state of this work.

2 SITS

Scrutable Intelligent Tutoring System (SITS) is a teaching system designed to reuse existing learning documents.

Figure 1 shows the main components of SITS.

The Concepts are the pieces of knowledge that SITS models in the learner model. The concepts are simply a vocabulary; no relationship structure between concepts is used in SITS. For example, in a C++ STL course some of the concepts might be: sort, copy, vector, and list.

The Basic Metadata is a database of references to documents. A reference to a LOM format metadata source is associated with the document if it is available. These documents are independent of SITS, and are simply reused by SITS.

The Learning Metadata is metadata relating documents in the Basic Metadata to the Concepts. The Learning Metadata is simply a list of Concepts associated with each Document. The concepts whose understanding is required in order to understand the document are listed as *prereq* concepts. The concepts that are taught by the document are listed as *shows* concepts. Finally, the concepts that are referenced by the document, but whose understanding is not necessary in order to make use of the document are listed as *uses* concepts.

The Learner Model uses an evidence-based approach based on um [4] in which evidence that the user knows or does not know concepts is stored. The learner model keeps track of the learner's knowledge of each of the concepts in the Concept vocabulary and also preference information for the learner.

The Teaching Agents are the central components of the system. Multiple Teaching Agents exist, though the learner selects a particular one to use at any given time. The Teaching Agent has the task of determining which concept the learner should learn next. Then it needs to choose which document the learner should view in order to progress toward the goal concepts. SITS does not place limits on how teaching agents go about their tasks. They are free to extend the interface beyond the SITS basics.

3 SV - a visualisation-based teaching agent

SV is the tool we have developed to visualise the documents and concepts in SITS. It is a heavily modified version of VIUM [9], a web-based tool for visualising large user models. The VIUM interface is designed as a Java applet which is loaded in a web browser so that the user can view their complete user model. The applet sits in a frame on the left side of the screen, and page contents are shown on the right.

SV accepts a graph where each node represents either a document or a concept. Each document connects only to concept nodes representing the concepts that the user must know, i.e. prerequisites, in order to be able to understand that particular document. Conversely, concepts have connections to documents that teach the concept. We use this to enable the user to see the relationships between the documents and concepts associated with them. SV displays the graph in two separate columns, with documents on the left hand side and concepts on the right.

Selecting a document will expand the prerequisite concepts. Selecting a concept will expand documents that teach that concept, allowing users to navigate back and forth between the document and concept space. For the document column, the colour represents whether the user is ready to read it or not - red means they have not fulfilled the prerequisite requirements, green means they are ready to read that document. Once a document has been read, its colour is changed to yellow and it is indented slightly to the right. The concept column uses green to mean that the user has learnt enough about this concept, and red to indicate that they have not and need to read the relevant documents.

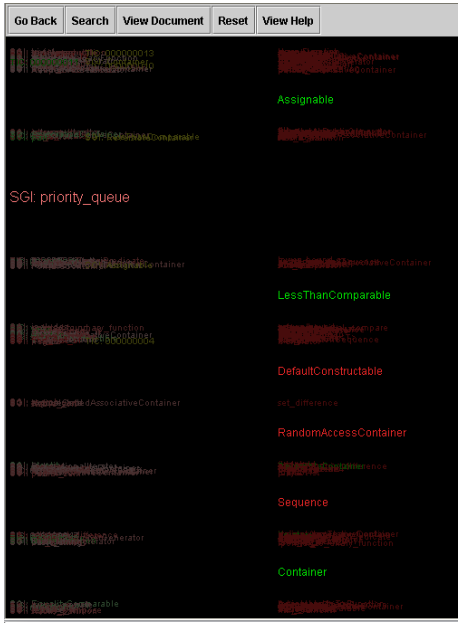


Figure 2: SV with document *SGI: priority_queue* currently selected. The most visible concepts on the right column - *Assignable*, *LessThanComparable*, *DefaultConstructable*, *RandomAccessContainer*, *Sequence* and *Container* - are the prerequisite concepts the user should know before reading *SGI: priority_queue*. The user can see that they still have to learn the concepts displayed in red - *DefaultConstructable*, *RandomAccessContainer* and *Sequence*.

As an example, a student Jane wants to read a particular document *SGI: priority_queue*, but the title appears red meaning she has not fulfilled all the prerequisites. She selects it (Figure 3), expanding the prerequisite concepts on the right hand side. The concepts *Assignable*, *LessThanComparable* and *Container* are green, indicating Jane has learnt these concepts. The remaining concepts *DefaultContstructable*, *RandomAccessContainer* and *Sequence* are shown in red, indicating Jane has not yet learn them. She then clicks on the unknown

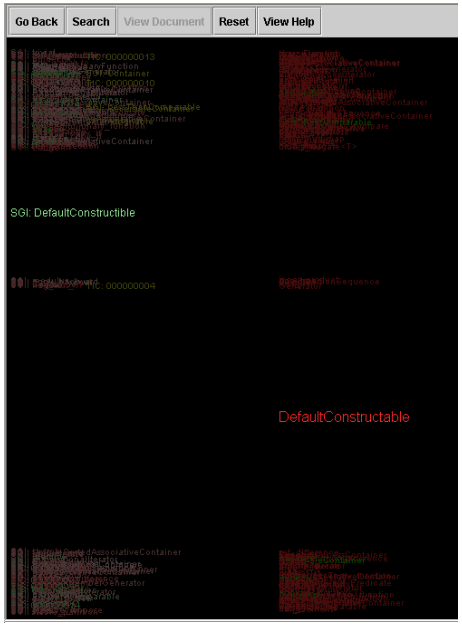


Figure 3: SV with concept *DefaultConstructable* currently selected on the right column. We can see that the document *SGI: DefaultConstructable* teaches this concept. The user is ready to read this document (as it is in green), indicating they already know the prerequisite concepts.

concept *DefaultContructable* to see what documents show this concept, causing the *SGI: DefaultConstructable* document to be displayed (Figure 3 on the preceding page). Jane decides she will read the *SGI: DefaultConstructable* document as it is green. This example shows how users can navigate through their user model and explore the relationship between the documents and the concepts they teach.

A SITS teaching agent has been implemented which used VIUM by generating a VIUM consumable RDF file of the metadata. Additional elements were added to the RDF for typing of the components and the relationships. Each component is designated as either a document or a concept by its own *componentType* element.

4 Evaluation

A think aloud experiment was performed with the recommended five users [6] for a basic qualitative evaluation of usability. All the users were undergraduate students studying computer science.

The users were asked to pretend they were a particular user who wanted to know about the *priority_queue* concept. For the purposes of the experiment they were told to assume they learnt all concepts *shown* by a document when viewing it. All the users managed to complete the task successfully. The colour and positional meanings of the concepts and documents in SV were explained to the users at the beginning.

Table 1: Think Aloud Results Summary

User	# Docs	Notes
User1	5	Found the colours too similar
User2	3	Used back button to observe changes to learner model
User3	5	Found interface easy to use
User4	7	Got confused between documents and concepts
User5	5	Liked the way the green increased as you progressed

A user model was constructed and each user was asked to use the system to 'learn' the *priority_queue* concept. The document, metadata, and user model were designed so that there were two documents that could be viewed to learn the *priority_queue* concept. Both those documents had *prereq* concepts which the user model indicated were not known. The first document had six *prereq* concepts, half of which were not known. Learning all of those required documents for which yet more *prereq* concepts were required because they were not initially known. The second document that taught the *priority_queue* concept had only one *prereq* concept. That concept was not known according to the user model.

It was possible to 'learn' the *priority_queue* by viewing two documents. None of the users tried to find a quick path through the documents. Instead they all headed depth first down the first path.

Table 1 shows the number of documents each user viewed in order to 'learn' the *priority_queue* concepts. As well as a representative comment on the users experience with the interface.

The main difficulty that all the users had was getting lost and losing track of the sub-task they were performing. This occurred after reaching a depth of about two *prereq* concepts (which involved clicking on a document, then a *prereq*, then another document, and then clicking another *prereq*). The users could not remember exactly why they had selected the concept. This was resolved in all cases by clicking the back button a few times and regaining their bearings. The fact that the users not actually using the system to learn, probably meant they weren't paying much attention to the actual concepts which would account for some of this problem. A real learner would probably not forget which concept they were trying to learn.

The users gave many suggestions as to why they got lost. The lack of a visible history indication to show the path they had followed was one complaint.

Another was that when a document is selected, the *prereq* concepts are indicated, but no indication of the *shows* concepts is provided. The lack of that indication caused some users to click back to remind themselves why they have clicked on the document.

One user confused the document and concept columns. They suggested column headers as a simple solution to the problem. Column headers would also be useful in providing the users with an indication of whether the concepts being indicated, are *prereq* concepts of the selected documents, or *show* concepts of the indicated documents.

All the users completed the task in a few minutes, and most of that time was spent waiting for the web server to respond to page requests. The VlUM applet itself was fast enough, with SITS being the speed bottleneck. This is due to the fact that SITS has not been optimised for speed. Simple caching techniques will speed SITS up, in a later test a speed improvement of about 20x was realised by caching just one database query between sessions. The entire graph they were exploring contained 171 documents, 155 concepts, and 620 edges, all of which were accessible from within SV.

The think aloud experiment indicated that the interface is usable, since all the users succeeded at the task without any assistance. SV provides an interface to a complicated graph which is fast to learn and use.

5 Discussion and conclusion

The system enables the user to explore a range of documents that explain the concepts the user is interested in. The user simply selects a concept they are interested in and suitable documents are indicated along with a colour indication of whether the user model indicates the user understands the required prerequisites. The users understood this, and successfully performed these tasks. One user was confused for a short time by the fact that green concepts are known, whereas green documents are ready to be used, and not already used.

The system also enables the user to explore the concepts which are required to understand a document. The user simply selects the document and those concepts are indicated, and coloured to indicate how well the user understands them. All the users has no problems understanding and using this prerequisite indication.

Navigation through the documents and concepts makes it easy to find the documents that teach concepts that are required by the document the user wants to view.

The main limitation of the interface is that it is not suited to long sequences, such as when the user needs to study a chain of five or six documents deep in order to learn a concept required for the wanted document. The interface does not provide a way to find out that a long sequence exists, other than clicking on the concepts and documents in the chain.

The user test showed that the interface is understandable and usable. It also identified some areas for improvement for the future.

The strength of the interface is that it provides a useful view of a set of documents and concepts. It allows the user to explore the document structure without having to understand the entire, possibly complex, graph.

A number of ideas for improvements came out of the user testing. Such as using a different colours for the documents and concepts, to keep the distinction between them more obvious.

The horizontal indentation of the documents could be used to indicate how suitable the documents are based on external metadata preference information, a certain author might be liked by a particular learner, for example. Indentation could replace colour entirely, by having items which are currently green near the center, and items which are currently red near the edges.

References

[1] Dublin core metadata element set, IETF RFC 2413.

[2] IEEE 1484.12.1-2002, learning objects metadata.

[3] Bruce Eckel. Thinking in c++ 2nd edition (volume 2). http://www.mindview.net/Books/TICPP/.

[4] Judy Kay. The um toolkit for cooperative user modelling. *User Modelling and User-Adapted Interaction*, Volume 4, Number 3, pages 149–196, 1995.

[5] Jak Kirman. A modest stl tutorial. http://www.cs.brown.edu/people/jak/proglang/cpp/stltut/tut.html.

[6] A J Nielsen. Estimating the number of subjects needed for a thinking aloud test. *International Journal of Human-Computer Studies*, Volume 41, Number 1–6, pages 385–397, 1994.

[7] P. J. Plauger. Dinkum c++ library reference. http://www.dinkum.com/refcpp.html.

[8] Silicon Graphics Computer Systems. Standard template library programmer's guide. http://www.sgi.com/tech/stl/.

[9] James Uther. *On the Visualisation of Large User Model in Web Based Systems*. Ph.D. thesis, University of Sydney, 2001.

The Nexus information hub for exploring social-informational context

Tim Mansfield, Markus Rittenbruch, Luke Cole, José Siqueira
Information Ecology Project
CRC for Enterprise Distributed Systems Technology (DSTC)
[timbomb, markusr, lrcole, jose]@dstc.edu.au

Abstract

The Nexus system is an "information hub" that helps users collaboratively manage and organise "contextualised social notifications. The purpose of the prototype is to act as a foundation for research into human information-sharing activity. The paper describes contextualised social notifications in more depth, links this prototype to the earlier Scuttlebutt prototype (presented at ADCS-6), and describes the architecture of the system and research questions.

Keywords Information Retrieval, Personalised Documents, Social Information Sharing

1 Introduction

The concerns of the authors are to investigate the connections between information and social structures (groups, communities, social networks etc). Each of us have previously conducted research about information-centred collaboration which assumed the existence of some notion of "group" as an artifact in the world and discovered the limitations of modelling that notion directly as a system object.

Our focus therefore is on trying to conduct research about supporting human information-sharing activity. We hope to investigate and explore how software might account for social structures more adequately. We take an empirical approach by both employing workplace studies to understand what potential users currently do and also by constructing, deploying and testing prototypes in response to what we find. This paper is primarily concerned with our current prototyping effort, more detail on user involvement in the project can be found in [7].

Our current prototype, called *Nexus*, manages what we call "contextualised social notifications" which we discuss in more depth in Section 2.

At ADCS-6 we reported [5] on the development of a prototype called *Scuttlebutt* which we dubbed a "social portal"; Section 3 explains what happened with that work.

The current Nexus prototype initially resembled a "news aggregator"[1], but our feature set extends beyond news aggregation, so we have taken to describing our system as an "information hub". We describe the information hub concept in Section 4 and the system architecture in Section 5.

2 Contextualised Social Notifications

Our prototyping currently focuses on gathering notification-style information, particularly that with a social intent, designed to communicate with others. Some initial examples of this kind of information are: web page update notifications (gathered by polling RDF Site Summary (RSS) files[2]) and notifications generated by the Elvin Tickertape chat tool [2].

Focussing on social notifications has the following advantages:

- They are small fragments of information with minimal internal structure, so they are easier to manage and process (for example RSS items versus the web pages they refer to).
- They appear in comparatively high volume at high frequency, so they provide a great deal of data to perform matching and inference on (for example Tickertape utterances versus emails from the same person).
- They often have an explicit context, a connection to other fragments of information (for example an RSS item is at least linked to a news article and a website) or a person or group. This means that they can lead us to establish broader interconnections between people and the more static kinds of information that they reference.

[1] "software that periodically reads a set of news sources, in one of several XML-based formats, finds the new bits, and displays them in reverse-chronological order on a single page." [10]

[2] RSS is an XML-RDF format for syndicating website updates and news. See [9]

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

- They are not usually critical to completing work tasks. This, coupled with the volume and frequency characteristics, means that users are more likely to be tolerant of experimental approaches.
- This kind of information (particularly RSS) is gradually gaining attention and developer focus in the technical community so tools and software are appearing to handle the basic formats.

We could refer to this kind of information that we intend to gather as *contextualised social notifications* (but we may just say "social notifications" as a shorthand). This current focus incorporates and supplants our earlier focus on "social recommendations" in the Scuttlebutt system.

3 Scuttlebutt: A Social Portal

The initial Scuttlebutt system [5] was designed to let users send recommendations to each other about web pages they found. It assumed that people communicated about a known set of topics to known sets of friends. We called the association between a group of friends and a topic a "channel". Users had a page that let them view all of their channels no matter where they came from and change the way they were laid out.

Around mid-year 2001, we had made enough progress to start demonstrating the system for users and getting feedback. We deployed the system at DSTC and initiated a sequence of field tests with usability studies to gather as much feedback as possible. Towards the end of 2001 we launched a set of more formal usability sessions that tested whether people could find their way around the system. All of this resulted in a flood of user requests for new features, UI redesigns and so on. It also questioned some of our basic assumptions in the system:

Recommendations: A minority of people regularly recommend pages to others, while most users of the system use it only to receive recommendations. This meant that the system suffered from a "bootstrapping" problem. It was hard to get enough useful information into the system for most users to find it valuable.

Channels: People who do recommend often do it on a much more *ad hoc* basis than we thought, to whoever occurs to them at the time, often on topics they had not thought of before. Forcing these users to create channels to cater for their often impulsive decisions to share information was too onerous. People largely stuck to existing media such as Tickertape or email.

Information Sources: While many people appreciated the "portal-style" of presentation, they were frustrated that other sources of information weren't present on the page, such as mailing lists, news channels from public web sites, Tickertape conversations, etc.

These factors reduced the use of the system and made it hard to adequately support people's need to share information. This meant that studying how that sharing occured in order to understand more about "communicative social interations" and their effect on software was fatally compromised. At the same time, it became obvious that we needed to provide more customisable or tailorable options in the system. These facilities were becoming more necessary to support our research goals as well. We were also becoming disenchanted with our development framework: Python web development support had improved markedly in the year since we had last surveyed the field and it seemed sensible to reimplement our codebase using one of the web frameworks (WebWare[3]) instead of using our own custom-built web application code.

All of these factors combined to suggest that rather than continue with Scuttlebutt, both its design and its codebase, we should pause for a moment and consider alternative ways to approach our research goals.

4 Nexus: An Information Hub

Our user community seemed to need a simpler (in the sense of having less *a priori* structure) and broader (in the sense of incorporating more information sources) system that allowed for more passive use. We have attempted to cater for these needs with a new system called, *Nexus* (see Figure 1).

Nexus aims to be an "information hub" that facilitates the gathering, management and presentation of contextualised social notifications by individuals in a coherent manner. The system addresses the common problem of personal information management. As the flow of information grows and the "irrelevant" begins to outweigh "relevant", it becomes more difficult for each of us to manage the information that we needed to work. As we increasingly use computers to collaborate, we need access to commonly known information which is bound to certain social- and work-related contexts.

A range of concepts which have evolved over the last decade have been addressing different aspects of these needs (shared workspaces, portal systems, systems integrating awareness information). An obvious lack so far is the integration and coherent management of information from different sources,

[3] http://webware.sourceforge.net/

Figure 1: Nexus 0.9 Screenshot

as well as the ease of sharing information in a low effort manner.

Our approach is broadly to:

- Gather these social notifications together into the hub where they can be effectively managed.
- Process the gathered information automatically to preserve and create connections between related information (news item to web site, news item to similar news item, etc).
- Give users abstractions to allow them to interactively create and express connections both between information and between information and people where they perceive them (news item relevant to friend, collection of keywords related to project etc).
- Present contextualised social notifications to users in a variety of ways to fit existing work practices.

This basic framework for gathering, managing and presenting social notifications provides a platform which lets us build and test more advanced research ideas. Currently, the following seem to be interesting questions:

- What effect does decoupling the input and output media have on potential use of a system like Nexus?
- How are shared information and shared adaptations of information control and presentation used in an organisational context?
- Is sharing and customisation a basis for supporting negotiation? What kinds of negotiation can be supported and what results from it?
- How do group structures emerge and evolve in a system that doesn't implement a rigid sense of group?
- What are the effects of making information control interdependent between individuals and social groups?
- How can a "vague" specification of shared interest (context) be used to share information within a system? (e.g. What is somebody else interested in? What are people who are interested in X interested in - where interest is defined by usage intensity, disclosure, group rating, etc.?)

5 System Architecture

This section outlines the concept of the Nexus information hub in more depth using a layered approach in which different kinds of contextual connections are introduced as new actors are introduced to the system.

- The fundamental layer of the system is a simple **information transport** that gathers social notifications, marshalls them in an information pool and presents them to the user.
- We add the individual user as an actor in the system by enabling **tailoring** in the fundamental layer. Users can select sources of information, select items using keyword filters and select the type of presentation they receive.
- By introducing **sharing**, the social network of the users effectively becomes an actor in the system. Users can share information by discussing items or even information sources and creating items via public recommendations. They can share aggregational patterns of information by sharing filters and collections. They will ultimately communicate implicitly via their patterns of use (presented with social navigation [1] techniques) and so on. This social action can unintentionally create links between items that were not evident to individuals acting alone.
- By enabling some **automated processing and inference** using Hyperspace Analog to Language (HAL) techniques [3] developed by Peter Bruza and colleagues [8] and Collaborative Filtering techniques [6, 4] the system becomes an actor as well, making connections that might be opaque to both individuals and social networks.

Each of the layers reinforces the other, contributing connections that the others would not uncover. The screenshot in Figure 1 shows an

initial inplementation of the information transport, tailoring and sharing layers.

An item collection called "Syndication" occupies most of the page. This collection is made up of items from a selection of relevant web sites and one or two keyword filters which can be changed by editing the page. A list of other item collections is shown at the top of the main navigation bar at the left. Functions to create new item collections, to manipulate the list of item sources, filters and email bulletins and to view the current list of users are arranged underneath. The search function at the top is a shortcut to creating keyword filters that search the entire item pool.

6 Conclusions and Future Work

In summary, led by a desire to investigate connections between information and social structures, we embarked on a user-centred design project aimed at exploring the notion and use of "contextualised social notifications".

We began the development of a system designed to help manage social notifications using individual, social and automatic techniques. We called this kind of system an "information hub". We plan to release that prototype to a field trial at DSTC and eventually at a participant site.

Our future prototype development and research will focus on two main goals. First, to further explore the possibilities of information sharing on the context of "social notifications" taking into account the complex interplay between the needs of individuals and communities of users. Second, to adapt our understanding of "social notifications" based on our experiences with deploying Nexus in complex "real-world" environments.

6.1 Acknowledgements

First, thanks go to our user-customers, Daniel Johnson, Jacqui Stewart and Andrew Wood for their time and patience in working with us to define the critical, fundamental functions of the Nexus prototype over Jan-Jul 2002.

The design-programming team for Nexus includes both former and current DSTC staff (in alphabetical order): Trond Abelseth, Dominik Bartenstein, Luke Cole, Gregor McEwan, Tim Mansfield, José Siqueira, Markus Rittenbruch, Nigel Ward and Anthony Wilkinson.

The work reported in this paper has been funded in part by the Co-operative Research Centre for Enterprise Distributed Systems Technology (DSTC) through the Australian Federal Government's CRC Programme (Department of Industry, Science & Resources).

References

[1] A. Dieberger, P. Dourish and et al. Social navigation: Techniques for building more usable systems. *interactions*, Volume 7, Number 6, pages 36–45, 2000.

[2] Geraldine Fitzpatrick, Sara Parsowith, Bill Segall and Simon Kaplan. Tickertape: Awareness in a single line. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI '98)*, Los Angeles, CA, April 1998.

[3] K. Lund and C. Burgess. Producing high-dimensional semantic spaces from lexical co-occurence. *Behavior Research Methods, Instruments, & Computers*, Volume 28, Number 2, pages 203–208, 1996.

[4] D. Maltz and K. Ehrlich. Pointing the way: Active collaborative filtering. In *ACM Conference of Human Factors in Computing Systems (CHI'95)*, Denver, CO, 1995. ACM press.

[5] Tim Mansfield, Nigel Ward, Gregor McEwan and Anthony Wilkinson José Siqueira. Designing a social portal. In *Proceedings of the Fourth Australasian Document Computing Symposium (ADCS'01)*, Coffs Harbour, Australia, December 2001.

[6] P. Resnick, N. Iacovou and et al. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 Conference on Computer Supported Cooperative Work (CSCW'94)*. ACM Press, 1994.

[7] Markus Rittenbruch, Gregor McEwan, Nigel Ward, Tim Mansfield and Dominik Bartenstein. Extreme participation - moving extreme programming towards participatory design. In *Proceedings of the Seventh Biennial Participatory Design Conference*. Computer Professionals for Social Responsibility (CPSR), June 2002.

[8] D. Song and P. Bruza. Discovering information flow using a high dimensional conceptual space. In *24th Annual International ACM SIGIR Conference.*, 2001.

[9] Aaron Swartz. RDF site summary (RSS) 1.0. http://www.purl.org/rss/1.0/. Accessed on 29 Nov 2002.

[10] Dave Winer. What is a news aggregator. http://davenet.userland.com/2002/10/08/-whatIsANewsAggregator. Accessed on 29 Nov 2002.

Liquid Mirò: Semantic Softlinking to Support Cooperative Document Exploration

Aaron Quigley *Nathan Lee*

Smart Internet Technology Research Group
School of Information Technologies
Sydney University, Australia

CustomWare
200 George St
Sydney, Australia

aquigley@it.usyd.edu.au *nathan@cutomware.net*

Abstract

In this position paper we present a non-intrusive mechanisim for evolving the overall quality of semantic relationships between elements of information in hypermedia document systems. The evolutionary aspect of this work is an application framework that includes a combination of hard links and temporal soft links between exisitng documents. A hard link completes the binding between two documents in the system upon creation, wheras a soft link delays the binding until some later time. The ablity to monitor, weight, integrate, delay, and then order the soft links is what offers the power in Liquid Mirò document systems. Here we focus on the use of hypermedia document systems which support exisiting online communities, ranging from social to professional groups..

Keywords Personalised Documents, Document Mangement

1 Introduction

On-line distributed hypermedia environments for targeted communities, continue to evolve at an unfettered pace across the web. In support of these growing environments, numerous navigation and exploration mechanisms have been employed. These mechanisms include, structuring the information in a hierarchical manner, site-maps, navigational patterns, link previews, textual and visual search engines and associated expressive query languages, end-user centric structure and review, along with expert manual intervention. Due to the volume of information, and the speed at which it both changes and grows, any structuring mechanism incorporating repeated human intervention becomes less efficient [1,2,7] and hence less useful.

Here we present a novel community based mechanism, to improve the quality of the overall hyper-linking structure, within a given community using hypermedia documents. This mechanism is further encompassed in the *Liquid Mirò* web application framework to couple existing document repositories to emerging client device types, both desktop and mobile. The goal is to allow specific communities to evolve abstract, un-predicted semantic associations, while presenting the information in a fluid form that can be interacted with via any class of device type.

Online community environments range from governments providing logistical information and support, to educational institutions providing distance education support to both educators and students. The typical model is for the few owners of the online environment to publish information as hypermedia documents which are then read by regular users of the environment.

In contrast to this, numerous on-line portals offer features, which more closely mirror the notion of a community as a group with “common rights and privileges”. On-line portals that offer community-building features include Yahoo groups and CNN interactive communities. Due to financial considerations such portals are turning to increasingly sophisticated software and interaction paradigms to reduce the manual effort involved in supporting, maintaining, and growing such on-line communities.

This paper presents an application framework for supporting communities using a variety of desktop and mobile access devices, as shown in Figure 1. This framework incorporates all the navigation and exploration mechanisms noted, along with the evolutionary formation and structuring of semantically relevant soft links. Instead of relying on manual effort to create lists or relationships we propose to let the actual usage of the system create and evolve the inter-connections between elements of information.

The rest of this paper is organised as follows, Section 1.1 describes the background of soft linking and semantic web structuring. Section 1.2 describes related work. Section 2 describes the Liquid Mirò framework. Section 3 outlines four possible Liquid Mirò web applications.

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

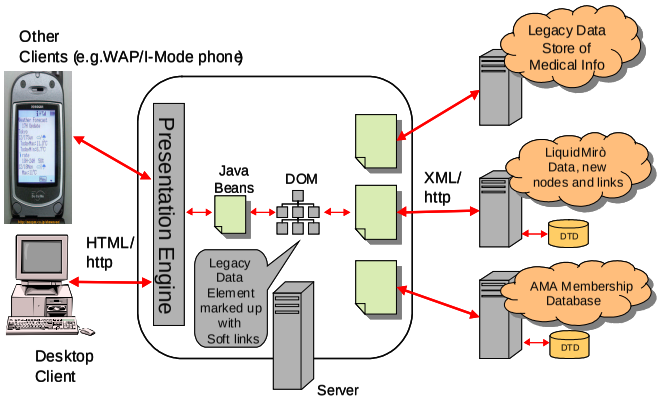


Figure 1: Example Liquid Mirò application overview for coupling a large legacy data source, data from a professional user organization with semantically relevant linking

2 Background

Many online communities are now centred around hypermedia systems, which superimpose an external structure on data [2,6]. Experience has shown that large-scale hypertext systems requiring extensive manual support are costly, error prone, and difficult to maintain [1,2,7]. These problems stem from the fact that even with support tools, maintenance is a laborious and often painstaking task.

The concept of hypertext *soft linking* between elements of information is not new and has been explored in file system research [8], software engineering [1], digital libraries [3], video streams [5], conceptual hypermedia [6], via dedicated services [3,4], and in online services such as Everything2. In the Liquid Mirò application framework we present a more generalised and powerful soft linking mechanism. Liquid Mirò based applications include the ability to monitor, weight, integrate, delay, and order the soft links before they are incorporated into elements of information.

Figure 2 indicates a soft link formation and ordering, between non-connected legacy information elements that is typical of Liquid Mirò applications. Here a registered user starts exploring the information at node X via existing structures, search engines, hard links or other mechanisms. Once the user arrives at another elements of information (a,b,c), Liquid Mirò creates a bi-directional soft link between the now semantically related elements. Based on other users usage patterns, the relative *strengths* of these links change. Depending on the application and user type, if other users don't follow such soft links, they eventually disappear, to be replaced by new soft links.

Dedicated soft link services provide a remote link lookup facility [3]. These services rely on augmenting web documents with soft links prior to delivery to the client. The end point of these links are determined via a dynamic search mechanism when the link is followed. The remote link service provides the end point of the link i.e. it determines the "late-binding" of the link.

Soft link structures such as conceptual indices, are automatically formulated based on an analysis of the overall hypermedia structure. Concept indices provide a temporal cross-linking mechanism, which represent

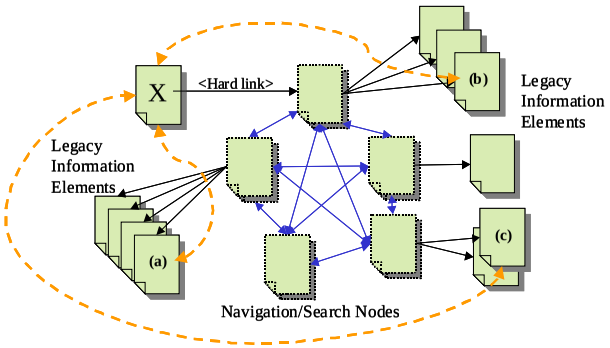


Figure 2: Example semantic soft link creation by user navigation

semantic relationships rather than pre-determined links.

2.1 Related Work

Slashdot is a prototypical online community, which is focussed on "*News for Nerds and Stuff*". This site allows its members to submit stories that are effectively moderated before they are posted on the web site. In the long term, a regular user can become a moderator. A variety of factors go into deciding which users become moderators and for how long. The end result is a pool of eligible moderators that represent the average positive Slashdot contributor. This ad-hoc approach has many limitations and still requires a great deal of manual intervention on behalf of the moderators.

Everything2 is social experiment, which is trying to form a large peer reviewed information web. Everything2 incorporates a soft linking scheme and aims to allow users to read, write, and share quality information. Within Everything2 it would not be possible to create a special purpose e-community to cater for example, to medical professionals or educators with specific needs. The vast quantity of irrelevant, incorrect and ambiguous information would soon render such a system useless. In contrast Liquid Mirò applications incorporate soft linking with an *existing* high quality document repository Systems based on this approach also allow registered users with different levels of access, and makes the information easily accessible across device types.

3 Liquid Mirò

The Liquid Mirò application framework is built around a mechanism of extending existing legacy data to incorporate a dynamic record of usage-based semantic associations within the data. The first step is to evaluate which existing data sources are suitable for integration. The basic criteria are that the legacy information elements can be embedded in a hypermedia system. Often the legacy data is stored in a relational database or flat file system. In the case of legacy data in a special format it can be fully re-engineered into a presentation independent format such as XML, or wrappers can be developed to access the data. Once each element of information can be uniquely distinguished by the system, then the information can be incorporated into a Liquid Mirò application.

Next a soft linking scheme must be decided upon to seed the legacy information units with soft links. Textual information is suitable for such a soft linking treatment, with soft link selection based on partitioning along words or phrases using a simple scheme. For a more sophisticated approach it is possible to integrate with one or more legacy information sources to determine where the seeded soft links may be found.

Development of a mechanism for the presentation of soft links and tracking of their usage requires knowledge of the presentation medium. This part of the Liquid Mirò framework is responsible for altering the community usage information associated with the existing legacy data. The emphasis is on creating a mechanism that enables the users to find the most appropriate links by themselves. The seeding process is merely to enable the most useful semantic associations to be discovered quickly. This mechanism permits a user viewing a particular piece of information to travel to what others or the system regards as the next relevant piece of information. This traversal creates a soft link if not already in existence or strengthens an existing soft link. The strength of the soft link can be represented in a number of ways by simple ordering, colour, size, or the use of glyphs.

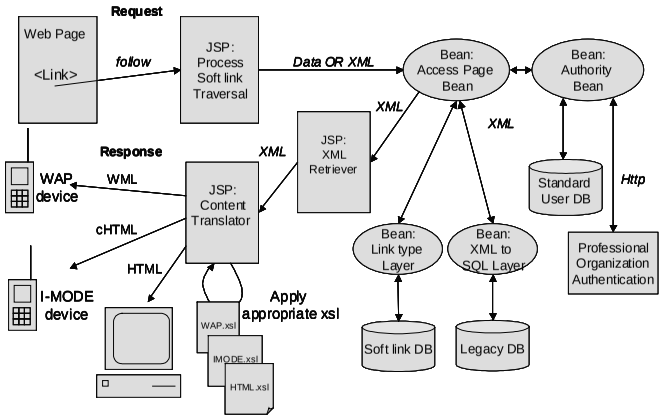
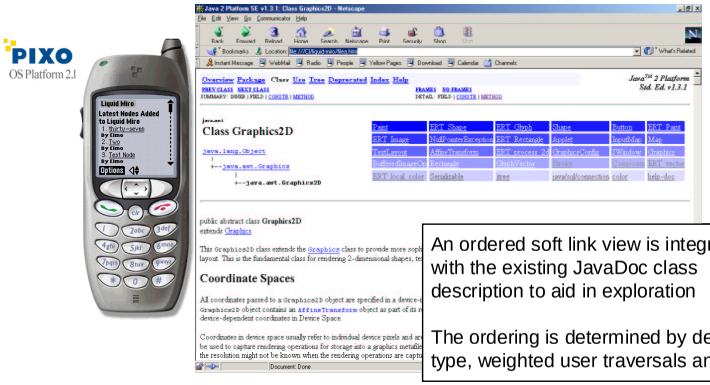


Figure 3: Liquid Mirò implementation (using JSP) supporting multiple client types, with client specific soft linking, internal and external user levels coupled to an existing information web

For more sophisticated implementations of Liquid Mirò, particularly in community based applications, a creating a community determined “most relevant” set of associations.



An ordered soft link view is integrated with the existing JavaDoc class description to aid in exploration

The ordering is determined by device type, weighted user traversals and time

weighting of both the influence a particular community member has and of the relative importance of the Soft Links when generating the presentation format might be integrated into the tracking and presentation mechanism. This can be achieved by integrating a separate user modelling system. For example, senior members of a community or organization might have far greater influence over junior members in terms of the affect their actions have upon the soft linking within the system. In terms of influence over presentation of soft links, weighting of soft links frequently traversed by community members using a similar type of browser might be higher than those used by members viewing with another type of browser (small screen mobile devices versus traditional web browsers). Figure 3 shows an example application permitting assignment of priority via integration with an authentication server and through separation of presentation logic from the main application through the use of XML and XSL to easily tailor the output format. The emphasis is on determination of soft links at request time rather than *a-priori*.

The example software engineering application of Liquid Mirò shown in Figure 4 demonstrates how an existing information service (the automatically generated API documentation of applications developed in the Java programming language) can be enhanced with the addition of soft links. The standard JavaDocs are a set of HTML pages that provide information on the software application. This takes the form of a page for each class in the system with in-line hyperlinks to any associated classes. With a Liquid Mirò framework built around this information source, the users of this system will reveal further relationships between the classes that are then available for use by others. The soft links are created as the software engineers navigate the documentation,

Acknowledgements

Particular thanks to Brett Anderson, Kersten Fernandes, David Hemingway, John Hinton, and Jeffrey Pond all whom allowed us to use screen grabs of their respective Liquid Mirò applications.

References

[1] Oli Kai Paulus, *et. al., Adding Softlinks to the Web*, 8th Joint European Networking Conference, http://www.terena.nl/conf/jenc8/proceedings.html

[2] Monika R. Henzinger. *Hyperlink analysis for the web*. IEEE Internet Computing, 5(1) pp. 45-50, January/February 2001.

[3] Leslie Carr, *et. al., The distributed link service: A tool for publishers, authors, and readers.* In Proc. Fourth International World Wide Web Conference. O'Reilly Associates, 1995.

[4] Eytan Adar and Jeremy Hylton, *On-the-fly Hyperlink Creation for Page Images*, in Proceedings of the Second Annual Conference on the Theory and Practice of Digital Libraries, 1995

[5] Jason W. Smith, *et. al., An orthogonal taxonomy for hyperlink anchor generation in video streams using ovaltine*. in Proceedings of Hypertext 2000, pp. 11-18.

[6] Mark d’Inverno and Michael Hu, *A Z Specification of the Soft link Hypertext model*, In ZUM’97: Proceedings of the Tenth International Conference on Z for Users Bowen, Hinchey and Till (eds.), Lecture Notes in CS, 1212, 225-240, Springer-Verlag, 1997.

[7] Leslie Carr, *et. al., Conceptual Linking: Ontology-based Open Hypermedia*, In Proceedings of WWW10, May 2-5, 2001, Hong Kong.

How to Write a Document in Controlled Natural Language

Rolf Schwitter
Centre for Language Technology
Macquarie University
Sydney, NSW 2109, Australia
schwitt@ics.mq.edu.au

Anna Ljungberg
Centre for Language Technology
Macquarie University
Sydney, NSW 2109, Australia
anna@ics.mq.edu.au

Abstract

This paper shows how a computer-processable document can be written in a controlled natural language (PENG) with the help of a sophisticated look-ahead editor (ECOLE). The editor provides syntactic hints after each word form entered and indicates how the author can continue the text. This way the author does not need to learn or to remember the restrictions of the controlled language. PENG documents are automatically translated into first-order logic via discourse representation structures. These formal entities can be checked by a theorem prover for inconsistency or consistency can be revealed by a model builder.

Keywords Document Processing, Controlled Languages, Authoring Tools.

1 Introduction

Documents such as software specifications or technical manuals are hard to write, as they need to be unambiguous and precise, the opposite to the qualities inherent in full natural language. Moreover, documents written in natural language are often hard to process automatically and inconsistency is difficult to detect. An alternative would be to write, for example, a software specification directly in a formal language. However, formal languages are hard to learn and to understand as they often abstract away from the facts of the application domain [4, 7].

A better strategy is to combine the advantages of a natural language with the advantages of a formal language and to hide the formality as far as possible from the authors. A controlled natural language, which is precisely defined by a restricted grammar and lexicon, can fulfill these requirements [1, 5, 6, 11, 12, 13].

PENG is such a controlled language and allows authors to write their texts in a well-defined subset of English, a language which is familiar to them [13]. To guarantee the painless and efficient usage of this language, ECOLE – a look-ahead editor – has been created and especially designed for PENG. The look-ahead editor guides the writing process and guarantees well-formed syntactic structures that can be translated into a formal language.

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

2 PENG by Example

PENG is a computer-processable controlled language specifically designed to write formal documents. PENG consists of a strict subset of standard English. The restrictions of the language are defined with the help of a controlled grammar and a controlled lexicon [13].

Let us illustrate the coverage of PENG by means of the Dreadsbury Mansion Mystery, a simple puzzle that is used in the literature [10] to test the capacity of automatic theorem provers. The puzzle is usually translated first by hand into a formal notation and the consequence of the puzzle is then proven by a theorem prover such as OTTER [9]. Using PENG, the manual translation becomes unnecessary, since PENG texts can be deterministically translated into first-order logic via discourse representation structures. Below follows the Dreadsbury Mansion Mystery in PENG:

> *A person lives in Dreadsbury Mansion. The person kills Agatha. Agatha is an aunt and lives in Dreadsbury Mansion. Agatha is a person. The butler lives in Dreadsbury Mansion. The butler is a person. Charles lives in Dreadsbury Mansion. Charles is a person. Every person that kills a person hates the person and is not richer than the person. If Agatha hates a person then Charles does not hate the person. If a person is not the butler then Agatha hates the person. If a person is not richer than Agatha then the butler hates the person. If Agatha hates a person then the butler hates the person. No person hates every person. Agatha is not the butler. Who kills Agatha?*

2.1 Controlled Lexicon

The lexicon of PENG consists of predefined function words such as

determiners: *a*, *the*, *every*, *no*, *who*, ...

prepositions: *in*, *than*, ...

copula: *is*

negation: *is not*, *does not*

coordinators: *and*, *or*

subordinators: *if ... then*, *before*, *after*, *while*

that build the structural scaffolding of the controlled language. User-defined content words such as

nouns: *butler*, *Agatha*, *Dreadsbury Mansion*, ...

verbs: *lives*, *kills*, *hates*, ...

adjectives: *rich*, *richer* ...

adverbs: *deeply*, ...

can be incrementally added or modified by the author during the writing process with the help of a lexical editor. Thus, by adding content words, the author creates his own application specific lexicon. In addition, the author can define synonyms for content words and acronyms or abbreviations for nouns. Illegal words (especially intensional words) can be defined in the lexicon by linguists.

2.2 Controlled Grammar

The controlled grammar defines the structure of simple PENG sentences and states how simple sentences can be joined into complex sentences by coordinators and subordinators. The grammar also specifies that simple sentences have by default a linear temporal order and that sentences can be interrelated in a well-defined way to build coherent texts. Simple PENG sentences are:

A person lives.

A person lives in Dreadsbury Mansion.

No person hates every person.

Agatha is a person.

Agatha is not the butler.

Complex PENG sentences are composed of simpler PENG sentences:

Every person that kills a person hates the person and is not richer than the person.

If Agatha hates a person then the butler hates the person.

3 ECOLE User Interface

The most important feature of ECOLE are the syntactic hints. After each word form entered, the author is given a list of choices of how to continue the sentence. These syntactic constraints ensure that the document remains unambiguous and precise. For example, when the author starts typing the sentence *The butler hates a person*, ECOLE displays the following look-ahead categories as subscripts in angle brackets:

The $_{[\ adjective\ |\ noun\ ]}$

The butler $_{[\ relative\ clause\ |\ verb\ |\ negation\ ]}$

The butler hates $_{[\ determiner\ |\ name\ ]}$ ...

As the example shows, the editor makes use of graphical means to display these syntactic hints as a help to the author. Not much linguistic knowledge is required to use this information as signpost. If something is unclear, then the author will be able to click on the displayed syntactic categories to get more information about that particular item.

The editor also handles compound nouns such as *Dreadsbury Mansion*. When the first noun *Dreadsbury* has been entered, the editor will respond with the second part of the compound noun *Mansion* and all other suitable look-ahead categories.

Another feature of ECOLE is the paraphrase that informs the author how the machine interpreted the input. Below follows an example of an input sentence and the paraphrase that is generated:

Input:

Agatha is not the butler.

Paraphrase:

Agatha is not [identical to] the butler.

The paraphrase thus makes it clear to the author that the copula (*is*) followed by a definite noun phrase (*the butler*) is interpreted as identity. If the copula had been followed by an indefinite noun phrase such as (*a butler*), then the machine would interpret this as a property and introduce a state.

PENG allows only well-defined forms of anaphoric references (definite descriptions and names, but no personal pronouns). The paraphrase displays how anaphoric references are resolved during parsing. An anaphoric expression is always replaced by the complete antecedent and the form is put within curly brackets. In PENG an anaphoric expression refers to the most recent accessible noun phrase that is suitable in terms of agreement, gender, and type, with respect to the nominal head and the pre- and postmodifiers.

Input:

A greedy butler lives in Dreadsbury Mansion.

The butler is a person.

Paraphrase:

A greedy butler lives in Dreadsbury Mansion.

{*The greedy butler*} *is a person.*

Another feature of the editor is the discourse representation structure (DRS) [8], which may be of interest to anyone wishing to see how the semantics of the information processed is represented. This feature can be used, for example, to teach students logic and computational semantics. A DRS captures the information in a multi-sentence discourse and shows the relations between the entities, the states, and the events in the application domain. From the following input the user will see the corresponding DRS:

Input:

Agatha kills the butler.

DRS:

```
[A,B,C]

named(A,agatha)

event(B,kill(A,C))

butler(C)
```

The first part of the DRS consists of a list of discourse referents `A, B, C` for the two individuals and the underlying event. These discourse referents are then used in the second part, which consists of conditions for the discourse referents. When a noun phrase is found to be anaphoric during parsing, it is directly resolved and not added to the DRS:

Input:

Agatha kills the butler. The butler is a person.

DRS:

```
[A,B,C,D,E]

named(A,agatha)

event(B,kill(A,C))

butler(C)

state(D,be(C,E))

person(E)
```

The example above shows that *butler* only appears once even though it is entered twice in the input sentence.

At first glance, PENG documents look informal and are easy to read, but they are in fact formal entities with all the nice properties of a formal language. Once translated into first-order logic these documents can be checked for inconsistency by a theorem prover or consistency can be revealed by a model builder [2].

4 How ECOLE Works

When the author types a word form into the editor then the current (partial) sentence is sent to the chart parser via a socket interface. The chart parser processes the input and generates the look-ahead categories. These syntactic categories are then displayed together with the paraphrase, the DRS, and the syntactic tree for the input for the options chosen by the author.

The grammar of PENG is implemented in the definite clause grammar (DCG) format. This unification-based approach allows us to resolve anaphoric references and to build up the DRS, the paraphrase, and the syntactic tree during parsing [3, 13]. Here is a typical grammar rule in DCG format:

```
n2( Agr, Index, Quant, Drs, Scope, ParaIn-
   ParaOut, [np,T1,T2], Gap-Gap, Ana)
   -->
   det( Agr, Index, Quant, Drs, Rest,
     Scope, ParaIn-Para, T1),
   n1( cat:cn, Agr, Index, Quant, Rest,
     Para-ParaOut, T2, Gap-Gap, Ana).
```

The chart parser processes such grammar rules top-down and produces edges according to the rules of chart parsing [5]. The edges have the following general form:

edge(START,END,HEAD,BODY)

Such edges simply tell us, what categories of a grammar rule ($HEAD \rightarrow BODY$) can span the substring of words found between the *START* point and the *END* point. We can distinguish two types of edges: active and inactive edges. An active edge is a hypothesis about a structure and an inactive edge is a result. For example, if the author types the determiner *the* into the editor, then the chart parser produces the following edges (simplified here):

```
edge(0,1,[det],[])

edge(0,0,[s],[n2,v2])

edge(0,0,[n2],[det,a2,n1])
```

```
edge(0,1,[n2],[a2,n1])

edge(1,1,[a2],[a1])

edge(1,1,[a1],[a0])

edge(0,0,[n2],[det,n1])

edge(0,1,[n2],[n1])

edge(1,1,[n1],[n0])
```

The first edge at the beginning of the chart is an inactive edge which contains an empty list `[]`. It represents a confirmed hypothesis and shows that a determiner has been parsed sucessfully between the nodes `0` and `1`. All other edges are active. That means that the chart is maintaining hypothesis about other structures that might follow.

The look-ahead categories are generated in the following way: During chart initialization the length `L` of the intput string is calculated and as soon as active edges are added to the chart that end at `L` then the leftmost category on the right hand side of a grammar rule is collected in a list. This results in two look-ahead trees from which the lexical categories `noun` and `adjective` can be easily derived.

5 Conclusions

This paper demonstrates how an unambiguous and precise document can be written in a computer-processable controlled natural language using a sophisticated look-ahead editor. Writing PENG puts no demands on the author when it comes to learn or to remember the rules of the controlled language as they are efficiently taken care of by ECOLE, the look-ahead editor.

The use of the look-ahead categories guarantees well-formed expressions and provides the necessary structural basis for the semantics of the controlled language in a completely compositional manner. PENG texts are deterministically translated into first-order logic and can be checked for consistency.

Acknowledgments

This research was supported by Macquarie University's New Staff Grant (MUNS 9601/0078). We would like to thank Mitko Razboynkov for developing the first version of the look-ahead editor, and David Hood for integrating the controlled grammar with the look-ahead editor.

References

[1] AECMA. 1988. The European Association of Aerospace Industries. *AECMA Simplified English*, AECMA Document PSC-85-16598. A Guide for the Preparation of Aircraft Maintenance Documentation in the International Aerospace Maintenance Language. Issue 1, Revision 1, January.

[2] J. Bos. 2001. DORIS 2001: Underspecification, Resolution and Inference for Discourse Representation Structures. In Blackburn and Kohlhase (eds): *ICoS-3. Inference in Computational Semantics*. Workshop Proceedings, Siena, Italy, June.

[3] M. A. Covington, D. Nute, N. Schmitz, D. Goodman. 1988. From English to Prolog via Discourse Representation Theory. Research Report 01-0024. Artificial Intelligence Programs, University of Georgia.

[4] N. E. Fuchs, U. Schwertel, and R. Schwitter. 1999. Attempto Controlled English - Not Just Another Logic Specification Language. *Lecture Notes in Computer Science 1559*, Springer.

[5] G. Gazdar, C. Mellish. 1989. Natural Language Processing in PROLOG. An Introduction to Computational Linguistics, Addison-Wesley, Wokingham.

[6] C. Grover, A. Holt, E. Klein, and M. Moens. 2000. Designing a controlled language for interactive model checking. *Proceedings of the Third International Workshop on Controlled Language Applications*. 29-30 April 2000, Seattle, pp. 29–30.

[7] M. Jackson. 1995. *Software Requirements and Specifications, a lexicon of practice, principles and prejudices*. Addison-Wesley, Wokingham.

[8] H. Kamp and U. Reyle. 1993. *From Discourse to Logic*. Kluwer, Dordrecht.

[9] W. W. McCune. 1995. *Otter 3.0 Reference Manual and Guide*, Argonne National Laboratory, ANL-94/6, Revision A, August.

[10] F. J. Pelletier. 1986. Seventy-five Problems for Testing Automatic Theorem Provers, *Journal of Automated Reasoning 2*, pp. 191-216.

[11] S. G. Pulman. 1996. Controlled Language for Knowledge Representation. *Proceedings of the First International Workshop on Controlled Language Applications*, Katholieke Universiteit Leuven, Belgium, pp. 233–242.

[12] R. Schwitter. 1998. *Kontrolliertes Englisch für Anforderungsspezifikationen*. Dissertation, Institut für Informatik, Universität Zürich.

[13] R. Schwitter. 2002. English as a Formal Specification Language. *Proceedings of the Thirteenth International Workshop on Database and Expert Systems Applications (DEXA 2002)*, Aix-en-Provence, France, pp. 228-232.

Studying the Evolution of XML Document Structures

Ivan Sun
School of Computer Science
and Information Technology
RMIT University
GPO Box 2476
Melbourne VIC 3001
Australia
isun@cs.rmit.edu.au

James A. Thom
School of Computer Science
and Information Technology
RMIT University
GPO Box 2476
Melbourne VIC 3001
Australia
jat@cs.rmit.edu.au

Abstract

The structure of XML (eXtensible Markup Language) documents often evolve over time, leading to reformulation and new version releases of the document structure. This occurs independently of the different ways in which XML document structure can be expressed. Major structural changes can cause version incompatibility issues and ensuing resource costs of updating existing documents. Software applications may be required to accommodate to both the old and new document structures. We present the case here for a study on the development process and evolution of XML document structures.

Keywords XML, XML Schema, DTD, Document Management

1 Introduction

Since the XML Recommendation [2] was released in 1998 as a simplified SGML (Standard Generalized Markup Language), it has enjoyed widespread usage as a standard for encoding and structuring documents. Many markup languages have emerged in the fields of industry, academia, government and science. Even the latest version of HTML, XHTML, sees it reformulated with XML rather than SGML.

Certainly one of the chief attractions of XML is its simplicity and flexibility of usage. To create a markup language, it is as simple (or so it would seem) as producing a Document Type Definition (DTD) according to the XML Recommendation. For example, the XHTML DTD is a *schema* for XHTML documents, where a schema refers to an expression of the document structure. As an alternative to the DTD, it is possible to use one of the other *schema types*, which are other ways of expressing XML document structure. These alternatives include XML Schema [6] and RELAX NG [4].

Perhaps because it is so apparently simple to create a markup language, and because there is not a widely recognised methodology for designing XML documents, there is a proliferation of markup languages, of which there are few mechanisms for controlling the quality of design. The worst fate that an ill-designed markup language may suffer is being ignored by the community of users.

However, poor design of document structure also leads to another undesirable outcome. Document structures often evolve over time, resulting in new version releases. Sometimes this is from unavoidable changes to the specifications controlling the usage of the XML documents, or changes to the environment in which the XML documents are used. Poor initial design may result in substantial revisions of the document structure with each version release, instead of minor revisions as the document structure evolves. Frequent major changes to document structure result in significant costs. There are version incompatibility issues and ensuing resource costs of updating existing documents. Software applications may be required to accommodate to both the old and new document structures.

Section 2 compares the DTD to another schema type and discusses how XML document design principles may apply across different schema types. Section 3 looks at different methodologies in the current literature for designing XML schema. In Section 4 we provide a case study of the 'XML Encoding for SMS Messages'. This derives from Koponen *et al.* [10], an IETF (Internet Engineering Task Force) draft for comment. It is an example of the evolution of XML document structure in its early stages. Finally in Section 5 we conclude the arguments for the benefits of further study in this area.

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

```
<?xml version="1.0"
    encoding="ISO-8859-1"?>
<!DOCTYPE message SYSTEM
    "SMSmessage.dtd">
<message cid="1" id="000001">
  <ack status="ok"></ack>
</message>
```

Figure 1: XML document from Koponen *et al.* [10]

```
<!ELEMENT message  ack, to?,  from?)>
<!ATTLIST message cid CDATA #IMPLIED>
<!ATTLIST message id  CDATA #IMPLIED>
<!ELEMENT ack (said?,from?)>
<!ATTLIST ack status
    (in_progress|ok|error) #REQUIRED>
<!ELEMENT said (#PCDATA)>
<!ELEMENT from (#PCDATA)>
```

Figure 2: Simplified extract from the SMS Message DTD from Koponen *et al.* [10]

2 Background

The XML Recommendation [2] defines a class of data objects called *XML documents*. The distinguishing feature of these documents is the use of named tags (for example, `<year>2002</year>`) as markup that conveys information about content of the element `year`. The start tag `<year>` also has a corresponding end tag `</year>` and may contain text data or other element(s).

Also defined in the XML Recommendation are the conditions for which a document may be *well-formed* or *valid*. In summary, when a document is well formed, it means that the document obeys the XML specification including the proper nesting of markup tags and the appropriate use of characters within the document. When a document is well-formed and valid, it means that as well as being well-formed, the document conforms to a Document Type Definition (DTD), which is a prescription of the structure that a document must adhere to.

2.1 DTDs and other schema types

A DTD prescribes the elements, attributes and entities that XML documents must conform to. The DTD may also be embedded within the XML document.

Figure 1 is an example of an XML document that can be validated by the DTD in Figure 2. These examples originate from the markup language for SMS Messages that is a case study in Section 4 of this paper.

The DTD is the only schema type specified in the XML Recommendation [2]. However it has several limitations as a means of defining document structure. First, elements have limited expression of cardinality. An element may occur once, or any one of the following three cardinalities: one or none (?), none to many (*), and one to many (+). There is no straightforward means of expressing certain integer occurrences, such as 2 to 6 occurring elements, within a single element declaration. Second, there is no data type constraint on the data that may be held by each element in the structure.

In XML Schema, a number of predefined data types are supported. In terms of cardinality of elements, specific minimum and maximum integer values are also supported. Schemas are expressed as well-formed XML documents, unlike the DTD. Furthermore hierarchical relationships of elements are expressed via nested elements. However, this results in more verbose declarations and is contrasted to the more compact DTD schema.

From the above comparison of the DTD and XML Schema, it can be seen that due to the different features of each schema type, it is not always possible to have a complete equivalence between two schemas of different types. For complex documents, this is even less likely.

This suggests that in designing XML documents, the intended document structure should be considered separately to what can be expressed in the different schema types. Schemas of these different types are merely textual representations of the intended document structure.

A study of the evolution of XML document structures may lead to the identification of design principles that can be applied to the various schema types. Furthermore, it may also be advantageous to model and to design XML document structure independent of the schema type. The model can then be translated to a schema of the schema type that best suits the application.

3 Existing Methodologies for Designing XML Documents

In this section we review methodologies based on Entity Relationship (ER) modeling, on information modeling via UML, and some other approaches.

3.1 Information Modeling from Database Modeling Theory

Information models, such as relational database information modeling, have well developed principles and processes ensuring good schema design and fault-avoidance. A translation from a relational database schema to an XML schema is a valid strategy of ensuring sound design of XML document structure.

Thom [11] proposes that the structure of XML documents may be derived from information models, such as those used for relational databases.

This is supported by an illustrative example of a conversion from an ER Schema to a DTD. Other information models such as the Relationship Management Methodology, the Unified Modeling Language (UML) and the Object-Oriented Hypermedia Design method (OOHDM) may be similarly extended to derive XML DTDs.

In a similar but more restrictive vein, Fong *et al.* [7] propose a methodology where a DTD is derived from an Extended Entity Relationship (EER) model. Relational data is used to build an EER model, which is then used to derive an XML structure. The purported advantage of this is that the EER model would be a stable, static representation of both XML and relational data.

3.2 Information Modeling via UML

Carlson [3] and Birbeck *et al.* [1] both propose similar methodologies for designing XML documents. There is an initial stage of modeling the information using UML diagrams, followed by a process of translating the Class diagrams into DTDs or XML Schema documents. Such a methodology uses the substantial body of literature in object oriented analysis and design as leverage in the process of designing XML document structure.

3.3 Other ways of designing XML Documents

Other ways of designing XML documents include Erdmann *et al.* [5] which describe an approach of using *ontologies* to derive DTDs to structure XML documents, and to query XML documents using a conceptual level of understanding.

An ontology "provides a formal specification of concepts, their relationship, and other realities of some domain [5]". DTDs are considered limited because they do not have a standard means of providing information about what the markup tags mean, and how they relate to the document structure. Erdmann *et al.* propose a software tool called ONTOBROKER, which outputs a DTD for a class of XML documents given. The input required is ontological information expressed in 'Frame Logic', a formal notation for representing ontology.

Erdmann *et al.* [5] recognise that their methods are not meant to completely replace other ways of modeling XML data. However it does provide an interesting alternative to ensuring a purposeful and strategic design of DTDs for structuring XML documents.

Other authors, such as Harold [8], advocate a more 'intuitive' approach. This involves listing the necessary data information for a particular class of XML documents, before culling unnecessary data components, and translating this to a DTD.

4 Case Study: XML Encoding for SMS Messages

Koponen *et al.* [10] is an IETF draft for comment that specifies a simple protocol for submitting SMS messages to mobile phones and other mobile terminals through the Internet. This draft for comment also provides a DTD for encoding the SMS encoded message to be transferred between the SMS Service Provider (SP) and the SMS Gateway. The SP hosts the SMS service, and provides content for the SMS service. The SMS Gateway acts as a proxy for interfacing with the hardware device (SMS Centre) that submits and receives SMS message via complex telecommunication protocols. The purported advantage of using an XML encoded message for the transfer between the SP and the SMS Gateway is that it replaces five existing protocols for communicating with the SMS Centre.

Koponen *et al.* [10] describe two of the SMS Message DTD versions: draft version 00 [9] and draft version 3 [10]. In the evolution of the SMS Message DTD between the two versions, the change items relating to the `messageID` and the `mref` elements stand out as indicative of a possible XML document design principle.

In version 00, we have the `message` element containing a variety of different child elements representing the different types of SMS messages that may be carried by the XML document:

```
<!ELEMENT message (submit*   |
            deliverstatusreport* |
            deliver*  |
            statusreportrequest* |
            statusreport* |
            delete*  |
            deletereport* |
            ack*  | nack* )>
```

Many of these child elements contain the `messageID` and `mref` elements, which represent character data identifying particular SMS messages sent. The following DTD extract illustrates this.

```
<!ELEMENT submit (da+, oa?, ud?, udh?,
       dcs?, pid?, vp?, timing?
       QoS?, messageID, cost?,
       rsr?)>
<!ELEMENT deliverystatusreport
       (mref, statuslist*)>
<!ELEMENT deliver (oa, da?, ud?, udh?,
       dcs?, pid?, scts?,
       location?, QoS?, mref)>
<!ELEMENT statusreportrequest
       (mref, messageID, da*)>
<!ELEMENT statusreport
       (mref, messageID, statuslist*)>
<!ELEMENT delete (mref, messageID, da*)>
<!ELEMENT deletereport (mref, messageID,
```

```
      deleteOK, deleteNOTOK)>
<!ELEMENT nack
      ((messageID | mref), errormessage)>
```

Here are the contents declared for the `messageID` and `mref` elements:

```
<!ELEMENT messageID (#PCDATA)>
<!ELEMENT mref (#PCDATA)>
```

In version 3, the `messageID` and `mref` elements have been removed from the different child elements of `message`. Instead, the information they represent are captured in the addition of two new attributes `id` and `cid` in the `message` element.

The justification of these changes to the DTD is that the "messageID and mref elements [are] used extensively in version 00"[10]. The above changes are illustrated in the following DTD extract from version 3.

```
<!ELEMENT message ((submit* | deliver* |
      ack* | deliverystatus* |
      statusreportrequest* |
      delete*  | deletereport*),
      to?,  from?)>
<!ATTLIST message cid CDATA #IMPLIED>
<!ATTLIST message id  CDATA #IMPLIED>
```

However an evaluation of whether this evolution represents a move to a better DTD design may include further consideration of whether:

- `messageID` and `mref` elements are required in all various types of SMS messages,
- the new DTD provides more visual clarity for the end-users, and
- the new DTD represents a closer match to the protocol specification.

The above example from the case study demonstrates the possibility that patterns may exist in the causes of change to XML document structure and that extensive case studies may unearth design principles that have not been previously identified.

5 Conclusion

We propose that further studies into the evolution of XML documents will contribute to:

- the formulation of design principles to inform future design of XML documents;
- a better methodology for evaluating the quality of design of document structure, such as fault categorisation; and
- a standard approach to designing XML documents independent of schema type.

We propose that such studies will incorporate an evaluation and synthesis of methodologies for the design of XML document structure in current literature. In formulating design principles, we advocate case studies involving real-life examples of markup languages characterised by significant evolution of document structure.

References

[1] M. Birbeck, J. Diamond, J. Duckett, O. G. Gudmundsson, P. Kobak, E. Lenz, S. Livingstone, D. Marcus, S. Mohr, N. Ozu, J. Pinnock, K. Visco, A. Watt, K. Williams and Z. Zaev. *Professional XML*. Wrox Press Ltd, Birmingham, UK, second edition, 2001.

[2] T. Bray, J. Paoli, C. M. Sperberg-McQueen and E. Maler (editors). Extensible Markup Language (XML) 1.0 (Second Edition) , October 1998. [Online] Available: `http://www.w3.org/TR/2000/REC-xml-20001006`.

[3] D. Carlson. *Modelling XML Applications with UML - Practical E-business Applications*. Addison-Wesley, New Jersey, USA, 2001.

[4] J. Clark and Makoto (editors). RELAX NG Specification. [Online] Available: `http://www.oasis-open.org/committees/relax-ng/spec-20011203.html`, December 2001.

[5] M. Erdmann and R. Studer. How to structure and access XML documents with ontologies. *Data and Knowledge Engineering*, Volume 36, Number 3, pages 317–335, March 2001.

[6] D. C. Fallside. XML Schema Part 0: Primer. [Online] Available: `http://www.w3.org/TR/xmlschema-0`, May 2001.

[7] J. Fong and F. Pang. Converting relational database into XML document. In *Database and Expert Systems Applications, 2001. Proceedings, 12th International Workshop on*, pages 61–65, September 2001.

[8] E. R. Harold. *XML Bible*. IDG Books Worldwide, Foster City, California, 1999.

[9] J. P. T. Koponen, T. Ikonen and L. Ziegler. XML Encoding for SMS messages. [Online] Work in Progress. Available: `http://www.xml.coverpages.org/draft-koponen-sms-xml-00.txt`, May 2001.

[10] J. P. T. Koponen, T. Ikonen and L. Ziegler. XML Encoding for SMS messages. [Online] Work in Progress. Available: `http://www.ietf.org/internet-drafts/draft-koponen-sms-xml-03.txt`, April 2002.

[11] J. A. Thom. Information models for document engineering. In M. Rossi and K. Siau (editors), *Information Modeling in the New Millennium*, pages 259–267. Idea Group Publishing, 2001.

Towards a System to Improve Administrative Processes for Front-Line Academic Staff

Roger Tagg and Madan Kumar Narayana Murthy

School of Computer & Information Science
University of South Australia
Mawson Lakes,5095, Australia

Roger.Tagg@unisa.edu.au Madan.Murthy@unisa.edu.au

Abstract

Current economic pressures are causing severe problems for many enterprises in maintaining service standards with shrinking headcounts. Groupware, Workflow and Agent technologies have been widely advocated as a solution, but there are few reported success stories. The project described in this paper addresses the case of running large undergraduate courses. A preliminary vision of a possible integrated administrative support system is presented, and the future activities necessary to advance such a vision are outlined.

Keywords Administrative Applications, User Interface Design, Software Agents

1 Introduction

Severe financial constraints in recent years have forced many enterprises, including universities, to try to achieve more and more with fewer and fewer resources, including staff. Staff members have become progressively less able to cope with the increased workloads, while consumers have experienced a gradual deterioration in the quality of the actual service for which the enterprise exists (e.g. teaching and supervision, health care etc).

A few years ago it was possible for a single academic to run a course with the help of the university's administration. Nowadays a team approach is necessary, including senior, junior, guest and contract lecturers, moderators and technical and administrative support staff. This team has to share both data and process knowledge.

Regarding data, the student's record is the primary document. But there are also assignment mark sheets, spreadsheets, lecture notes, study guides, documents on the course website, documents for tutors, computer files for practicals (e.g. provided data and source programs).

From the process viewpoint, teams – especially where staff turnover is high - need more help in carrying out the processes than when the course is run by a single lecturer.

The authors of this paper are a senior lecturer and a recent Masterate graduate in a school of computing and information science. It is therefore appropriate to consider how we can take some of our own medicine. Our school is involved in research into groupware, workflow and intelligent agents, which have been proposed as a means of improving the effectiveness of workers in teams and enterprises, for example [1]. We develop theoretical models, and talk about potential benefits, but have yet to apply these in practice.

At the same time, university computer developments such as ERP systems for general student administration, together with web interfaces, have improved both general administration and student accessibility. But they have not helped improve the processes carried out by front-line academics. A recent project at the University of Queensland has developed a system, FlexEl [2], that uses a workflow engine to coordinate a self-paced course; however this does not address the wider aspects of team teaching administration support.

2 Case study: running a large undergraduate course

Our school currently runs 17 core undergraduate courses that the majority of students must take. Third year numbers at the main campus are over 200, with double this in the second year and triple in year one. Teaching is offered at 2 or 3 campuses in South

Proceedings of the 7th Australian Document Computing Symposium,
Sydney, Australia, December 16, 2002

Australia, and there are sometimes external students as well. Courses following the same syllabus and taking equivalent assessments are offered at an increasing number of sites in South East Asia. Some courses additionally run evening streams.

Our current project has been started with the dual intention of improving support for transient staff working in team teaching and providing a test bed for workflow and agent technologies. The authors have started to create models describing the current data, role and process structures.

The *organization model* describes the reporting structure of the various roles involved in team teaching, including course coordinator, moderator, lecturers, tutors etc and also administrative staff.

The *data model* describes the artifacts involved in team teaching, including lectures, tutorials, practicals, self-study modules, assignments, tests and the documents that relate to these artifacts. Data can be related to the whole course offering, or to individual students or project groups.

The *process models* are divided into three main groups: running a course, running a degree program and running a project or thesis. In each process model there is a 4-level hierarchy:

1. run the course, program or thesis over its lifetime
2. run it in any one academic year
3. run one offering (i.e. with a given start date, at a given campus)
4. run one activity within an offering (e.g. assessment, on-campus module).

3 Preliminary design of a prototype system

As the main motivation for this project is to improve the efficiency and effectiveness of staff, the user interface has been an early part of the design. Figure 1 shows a large window that would be offered to the user on startup. This is assumed to occupy all except the outermost edges of the user's screen, where essential desktop icons could remain visible.

The center of this workspace window would be available to whatever applications the user starts up, either explicitly through the Operating System or implicitly by clicking the buttons round the edge of the window. Some of these buttons support a "drop box" functionality; the actual semantics of dragging and dropping an object onto such a button will be different for each button.

The buttons are arranged in groups, some of which are common to all users, but others of which only apply to those users who have responsibilities in specific courses and degree programs.

A key concept in this system is *context*. This can be established by one of the following means:

- the area of responsibility the user is currently working on (see the highlight on course 123456)
- the function the user is currently carrying out (e.g. the topmost window)
- explicitly by the user dragging and dropping the window he/she is working with into one of the drop boxes.

The W*orkflow design* tool allows the user to tailor the workflow, either for the current case or "from now on". A currently live work item can be dropped into either the *New case* or *Action this case* box. The user can review the progress of all cases within a workflow type, with an optional selection criterion (e.g. all students doing the BInfTech(SwEng) program).

The *Tasks* and *Appointments* areas have drop boxes where e-mails and other documents can be dropped. An agent will scan the dropped document and deduce any tasks or appointments, not just for the current user, but for other members of the teaching team.

The main *E-mail* drop box is an intelligent outbox agent. The agent should be able to format the e-mail for the mailer software being used, and to deduce the email address if possible.

Other general buttons include *Upload* and *Download* for a mobile computing device; *General trash* (an intelligent archive) and *General filing*. This is an intelligent drop box that works out where the document should be filed, either from context (e.g. text within the document itself, current live task) or by prompting the user.

The *University student records* system drop box is an intelligent agent that sends valid transactions to this server system. One common use would be to drop the course marks into it at the end of each teaching period. Students would be matched by keys. Data columns could be matched by name, or the columns highlighted in the source spreadsheet.

In the bottom left corner *My avatar/avataress* allows for possible future use of generated voice prompts and responses.

The remaining groups each represent an area of this user's main responsibility, such as a course. The range of buttons might vary, but for most courses there would be buttons for *Teaching team*, *Planning* and *Teaching resources*; and drop boxes for the *Course website*, *Course filing*, *Classes and Groups* and *Course marks*.

The *Course website* drop box would work out a suitable URL (Universal Resource Locator) for a page where the dropped document would be stored, ensure that valid links were built to this URL, and arrange file transfer to the live web server. Also, a planned course offering could be dropped in the *Course website* drop box to invoke creation of a new subdirectory for the new offering.

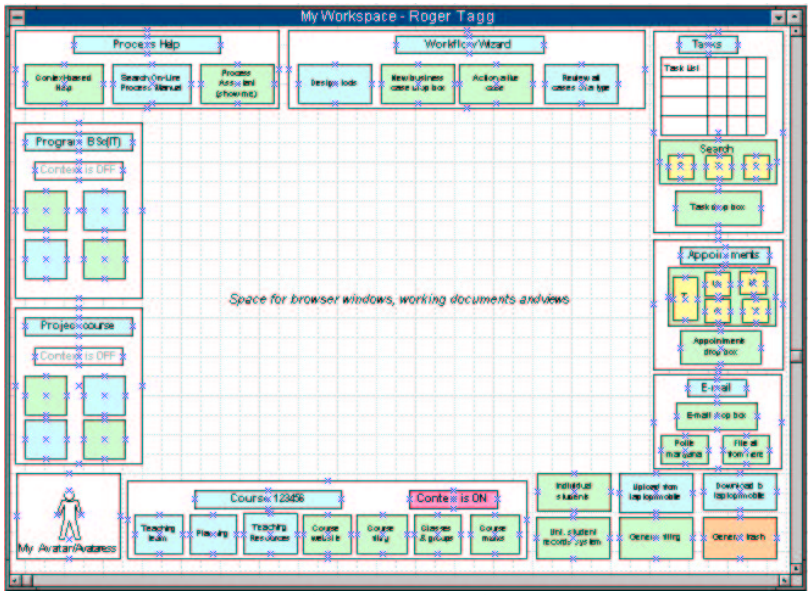


Figure 1: Preliminary Design Layout for the User Interface for a Team Teaching Support System.

The *Course filing* drop box would behave similarly to the general filing one, but would attempt to store documents in their correct place within the folder structure being maintained for the course.

Classes and Groups is used to maintain student allocation to tutorials, practicals and project groups. If the *University student records system* supports class level enrolment, initial allocation can be set up by drag and drop. Formation of project groups - and entering of subsequent changes - can be assisted by dropping student details from *Individual students*

The *Course marks* drop box supports actions relating to the spreadsheets that contain the history and marks of each student on each offering of the course. The main actions would be as follows:

- Drag and drop the *University student records system* onto *Course marks* to download student enrolments, both initially and to refresh the class list
- Generate marking spreadsheets (or other forms) for the markers of assessments, and send these out as attachments to emails
- Drop in the markers' completed spreadsheets when they are emailed back
- Finally, drop the *Course marks* drop box itself onto the *University student records system* at the end of the semester to register the final grades.

4 A planned software architecture

Figure 2 shows the client and server components, together with various other systems with which the user works (either by choice or by decree). Some specific technologies planned for the system are discussed below.

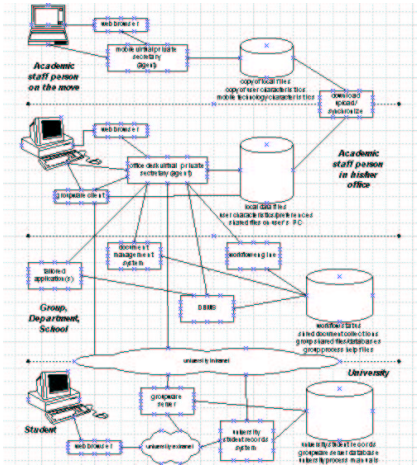


Figure 2: A planned software architecture

We plan to make use of a collection of agents (or components) of various degrees of intelligence. Some will capture and utilize the knowledge of what the users do frequently, and will fit in with their preferred formats and ways of working.

Others will service the drop boxes and buttons. In some cases these agents will provide intelligent front-ends for typical groupware functions such as task lists and calendars, which may reside in a package such as Microsoft Exchange or Lotus Notes. We have christened the aggregation of all our client-side agents as the "virtual private secretary".

On the server side we plan to include a workflow server and a shared process help system.

With regard to the workflow component, totally predefined processes are neither practicable nor acceptable in this application. Therefore the total process support system has to be a mixture of services, ranging from ad hoc requests for process help, through workflow control applied at a user's request, to predefined workflow with ad hoc exceptions and routing changes. A flexible workflow software approach is therefore a requirement.

We plan to adopt, where possible, techniques that reduce keyboard data entry by teaching team staff. Examples are bar coding, character recognition, telephone enrolment by students – and in the future, voice input. Wherever we can, we want to use data that already exists somewhere on the university's network. We have already mentioned the re-use of course mark spreadsheets instead of re-inputting marks into the student record system. Another opportunity arises when entering information about persons. Many people can be identified by data held in the staff or student records systems, and by such things as personal web page URLs.

Because of the need to link the proposed system with other existing or planned IT systems, there will also be a need for intelligent agents to resolve unclear matching with data from other systems, databases and files. These agents would make use of database schemas, XML(Extensible Markup Language) DTDs (Data Type Definitions) and ontologies.

It is envisaged that the proposed system will be made up of a number of autonomous agents, rather than a monolithic client or server. Components (and possibly third-party web services) will be capable of being plugged out and in when justified.

5 Current and future work, and conclusions

We have started modelling the existing processes – most of which are not documented at all - and are building up a process inventory. As the team teaching approach is still being implemented, these processes are not settled. We will next be seeking feedback from our colleagues on the correctness of our models and usage scenarios.

We have also started building a simple system to record receipt of assignments, based on the use of bar code readers to scan assignment cover pages that contain the student or project group's identity, as well as the course and assignment number, in a bar code font. This system is due to operate live from the beginning of the year 2003.

We are about to start an evaluation of the Chameleon flexible workflow engine from DSTC (Distributed Systems Technology Centre, based in Brisbane, Queensland) [3]. Chameleon has been developed from the workflow engine that was developed for FlexEl [2]. The corresponding modelling tool for Chameleon is FlowMake. We are already using the Adonis [4] business process management tool to chart some of our processes, and are building a translator from Adonis to FlowMake.

In 2003. we have scheduled a project to investigate the use of software agents to support our proposed system. At this stage we will carry out a full literature search and evaluate what existing agents we can utilize. We will then prototype parts of the proposed system and seek further feedback.

In conclusion, groupware, workflow and intelligent agents are important research areas in document management and computer science. But theory has got a long way ahead of practice. This project aims to utilize these technologies in a pragmatic way on our own working environment. It is significant that our Vice Chancellor has recently nominated efficiency and workloads as two of the most important issues to be tackled in 2003!

References

[1] Chaffey, D., Groupware, Workflow and Internets, Digital Press, 1998

[2] Distributed Systems Technology Centre, Annual Report, 2000. Available from http://www.dstc.edu.au/Publications/Annual_Report/AR00.pdf

[3] Distributed Systems Technology Centre, Praxis Project Website, http://www.dstc.edu.au/praxis/

[4] BOC Ltd. Website. http://www.boc-eu.com/english/index.shtml

Buying bestsellers online: A case study in Search & Searchability

Trystan Upstill
Deparment of Computer Science
Australian National University
Canberra ACT 0200
Trystan.Upstill@cs.anu.edu.au

Nick Craswell
Enterprise Search Group
CSIRO Mathematical and Information Sciences
GPO Box 664, Canberra, ACT 2601
Nick.Craswell@csiro.au

David Hawking
Enterprise Search Group
CSIRO Mathematical and Information Sciences
GPO Box 664, Canberra, ACT 2601
David.Hawking@csiro.au

Abstract

A website's design directly affects how well search engines can crawl, match and rank its pages. For this reason, searchability is an important concern in site design. We study the interaction between search engines and Web sites by means of a case study of online bookstores and general-purpose search engines. The task modelled is that of finding web pages from which a book, described by its title, may be purchased.

We first compared the relative effectiveness of search engines in finding pages matching the criterion, regardless of bookstore. Then we compared the relative searchability of the bookstore websites by observing how many times each bookstore contributed useful answers to the search results.

Large differences in the performance of both search engines and bookstores were observed. Two of the search engines performed better than their peers, and one bookstore was far more searchable than all others. To further explore these differences we tabulate the total number of pages from each bookshop which are included in the search engine indexes.

We conclude with recommendations both to bookstores on how they may improve their Web presence, and to search engines on how they may improve their performance for product searches.

Keywords Information Retrieval

Additional Keywords Web search, evaluation, transactional search

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

1 Introduction

The World-Wide Web is commonly used by consumers to research and purchase goods [7]. With this demand for goods and services online, product finding is becoming an increasingly important consideration for Web search engines. However, despite the prevalence of such tasks, the general problem of product, or *transactional* [11] search has not been thoroughly examined.

To begin to understand the difficulties involved in processing transactional searches we have conducted a study of the most popular Web bookstores and search engines. Our study examines product finding both from a search engine and from a bookstore point of view. This is necessary because any difficulties involved in Web product search may well be due to a mismatch between search engines and product providers.

Previous work has evaluated the service finding ability of TREC search systems [5] and Web search engines [6] on a set of apparently transactional queries extracted from natural language Web logs. In these studies the sole aim was to compare search engines on early precision; no information was available (or needed) about what resources were available to find and there was no opportunity to compare the *searchability* of online vendor sites. Searchability is concerned with *crawlability*, that is how easy it is to retrieve pages, and how well a site is matched and ranked by search engines.

Here, we study both search effectiveness and searchability with respect to a particular type of commodity (books) which is frequently sold over the Web. We measure the relative effectiveness of a selection of search engines in finding pages from which a book, specified only by its title, may be purchased. We also compare the relative searchability of a selection of online bookstores. This

is performed through an examination of the proportion of best-seller books for which a "buy-here" page from a bookseller appears in search engine listings.

2 Method

The process we followed to gather and evaluate our data is outlined in this section. We began by identifying our book query set. Following this we identified a set of search engines for querying. We sent our queries to the search engines and retrieved and recorded the first 1000 results. We then used popular directory listings to determine which bookstores would be included in the study. Finally we determined the correct books for our queries (by ISBN) and created a listing of correct bookstore URLs at which they could be found.

These steps are detailed below:

2.1 Query set

The query set was derived from the titles of the New York Times best-sellers for September 2002 [14]. We found 206 unique book titles in 9 categories. The book/category breakdown is outlined in the Appendix. We sent the titles to search engines as phrases (i.e. inside ‘ ‘ or marked as phrases in advanced searches). Since the book titles on the best-seller lists were fully capitalized, we changed their case to combinations of lower and upper-case letters. All words, apart from join words (such as "the", "and" and "or"), begin with a capital.

2.2 Search engine set

Table 1: **Search Engines Properties.** The column labelled "Abr." contains abbreviations used through the paper. "Used by" reports search services that use the search system. "Rank" reports the position in the Nielsen/NetRatings Search Engine Ratings for September 2002 [9].

S.Engine	Abr.	Used by [10]	Rank
AltaVista [1]	AV	AltaVista	8
Fast [2]	FA	AllTheWeb	-
Google [4]	GO	Google	3
		AOL	4
		Netscape	9
		Yahoo	1
MSN Search [8] (based on Inktomi)	MS	MSN Search	2
		Looksmart	10
		HotBot	-
		Overture	6

Table 1 outlines the characteristics of the search engines examined in our study. Our studies span four popular Web search systems. These systems provide core search services for the 4 most popular search engines, and for 8 of the top 10 search services [9].

The query syntax submitted to each search engine is reported in the Appendix.

2.3 Bookstore set

The bookstore set was derived from the Google *"Shopping > Publications > Books > General"* [3] and Yahoo! *"Business and Economy > Shopping and Services > Books > Booksellers"* [13] directories. Bookstores were included if they sold the top bestseller in at least three of the nine categories. The books were found using internal search engines to search for both the title and the author of each book. The bookstores were only judged on the categories for which they stocked (or listed) the bestseller. The intuitive justification for this was that there may have been some specialised (e.g. fiction only) bookstores and we did not want to exclude them from our study. The full list of the 38 eligible bookstores and their salient properties is provided in Table 7 in the Appendix.

2.4 Correct answers

In our study correct answers must fulfill two criteria: the page must relate to the book whose title is given as the query, and the retrieved page must be transactional in nature.

When identifying book identities we deemed correct answers to be either hardcover or paperback editions of the book[1]. A list of the queries and the ISBNs of the books judged as correct answers is available [12].

We define a *transactional page* to be a bookstore page from where you can buy a book. Browse pages, search results or genre listings are not judged as correct results. For many bookstores the correct answers were observed to have the hardcover or paperback ISBN in the URL (in many cases there were many correct URLs which were all observed to contain the ISBN). To cut down on manual judging for these bookstores, we performed automatic judging based on the presence or absence of the ISBN in the URL. For other bookstores we located unique product identifiers for each book and checked URLs for their presence.

3 Results

We begin by outlining our findings from a book retailers' perspective - detailing which bookstore was the most searchable. We then report which search engine was best at finding books. Finally we examine the size of the index and link graph for each bookstore reported by each search engine.

[1] Large print books and audio books were deemed to be incorrect answers.

Table 2: **Bookstore comparison.** This table includes all bookstores which had at least one success at 1000 for any search engine. *Powells* is included in the table due to the high number of results matched in search listings. The S@1000 breakdown column gives an indication of which search engine returned what correct books from which bookstores in its top 1000. The "Hostname Results" column reports the number of pages found for each bookstore by all search engines.

Bookstore	S@1	S@5	S@10	S@100	S@1000	S@1000 breakdown (AV:FA:GO:MS)	Hostname Results
Amazon	0.124	0.325	0.402	0.492	0.584	104:83:162:132	3903
Barnes and Noble	0.028	0.096	0.140	0.225	0.316	0:87:170:3	3603
Walmart	0.010	0.030	0.045	0.070	0.075	2:0:0:60	277
BookSite	0.000	0.004	0.005	0.013	0.013	0:0:0:11	52
ecampus	0.0	0.0	0.0	0.005	0.012	0:7:0:3	290
AllDirect	0.0	0.0	0.0	0.002	0.005	0:4:0:0	52
NetstoreUSA	0.0	0.0	0.0	0.001	0.010	0:8:0:0	261
Sam Weller's Books	0.0	0.0	0.0	0.001	0.006	0:5:0:0	22
Books-A-Million	0.0	0.0	0.0	0.0	0.008	0:4:0:3	775
1BookStreet	0.0	0.0	0.0	0.0	0.006	0:5:0:0	17
Wordsworth.com	0.0	0.0	0.0	0.0	0.004	1:0:1:1	92
TextbookX.com	0.0	0.0	0.0	0.0	0.002	0:2:0:0	22
CodysBooks.com	0.0	0.0	0.0	0.0	0.002	0:2:0:0	78
Arthurs Books	0.0	0.0	0.0	0.0	0.003	0:1:0:0	3
Powells Bookstore	0.0	0.0	0.0	0.0	0.0	0:0:0:0	1031

3.1 Bookstore comparison

We first measured the book finding success rates at several cutoffs. The success rate measure is indicated by S@n where n is the cutoff rank. S@10 measures how often at least one correct page is returned within the first 10 results, and S@1 corresponds to the probability that a right answer appears at rank 1 (cf. the "I'm feeling lucky" button on **Google**). We also evaluated S@5, S@100 and S@1000.

Table 2 contains the results for this experiment. Our observations are that:

- Of the 38 bookstores evaluated only 14 returned any correct answers within the top 1000 results by any of the search engines.
- Only 4 bookstores ever made it into the top 10 in any search engine: *Amazon*, *Barnes and Noble*, *Booksite* and *Walmart*
- *Amazon* was the most searchable bookstore in our evaluation, achieving the highest success rates.
- Only *Amazon* had correct results returned by every search engine.
- *Barnes and Noble* performed well on **Google** and **Fast**.
- *Walmart* performed well on the **MSN Search** search engine.
- The only search engine which returned results for many of the smaller bookstores was **Fast**.

3.2 Search engine comparison

Table 3: **Search Engine Success Rates.** The best result at each cutoff is highlighted.

search engine	Success Rates @1	@5	@10	@100	@1000
AV	0.14	0.39	0.45	0.50	0.52
FA	0.00	0.02	0.05	0.18	0.52
GO	0.15	0.56	**0.67**	**0.83**	**0.89**
MS	**0.36**	**0.57**	0.65	0.72	0.73

Table 4: **Search Engine Precision.** Note that precision at 1 is equivalent to the success rate at 1. The precision at measures below 100 is less than 1/100 in all cases. The best result at each measure is highlighted.

search engine	Precision @1	@5	@10	@ 100
AV	0.14	0.08	0.05	0.01
FA	0.00	0.00	0.01	0.00
GO	0.15	**0.20**	**0.15**	**0.03**
MS	**0.36**	0.13	**0.08**	0.01

Our observations on the success rates and precision reported in Table 3 and Table 4 were:

- **AltaVista**'s (AV) performance was inferior to that of both **Google** and **MSN Search** at all cutoffs. **AltaVista** demonstrated around half the precision of **MSN Search**.
- **Fast** (FA) trailed well behind all other search engines, but provided a large number of correct answers between the 100th and 1000th position (success rate jumps from 0.18 to 0.52). The precision for **Fast** was very low.
- **Google** (GO) trailed **MSN Search** at S@1 but surpassed its performance from S@10 onwards. **Google** returned more correct answers in their top 5, 10 and 100 results than did **MSN Search**.
- **MSN Search** (MS) produced the strongest results at S@1 and S@5, but did not improve as much as other search engines when cutoffs were extended.

3.3 Search engine/bookstore URL coverage

The transactional pages for some bookstores may not be returned because they have never been crawled by a search engine. Table 5 lists the number of pages from each bookstore reported to be contained within each search engines' index.

From these results we observed that:

- *Amazon* had a consistently large search engine coverage (around three million on three out of four search engines). **Fast** covered an order of magnitude less documents from *Amazon* than did the other search engines, but still more than it did for any other bookstore.
- The coverage of *Barnes and Noble* varied widely. While the **MSN Search** coverage of *Barnes and Noble* was small it appeared to contain product pages, with three correct answers retrieved. Only 500 *Barnes and Noble* pages were covered by **AltaVista**. Over a million pages were covered by **Google**.
- A large number of *Walmart* pages were covered by **MSN Search**, whereas **Fast** and **Google** covered a relatively small number of pages.
- **Fast** did not have very large coverage of any one bookstore (maximum was 360,000). They tended to have a larger breadth of results, with larger crawls of lesser known bookstores.
- **AltaVista** had large coverage only of *Amazon*, *Walmart* and *Powells*. We could not expect to find book results in their small (sub 1000 page) crawls of other bookstores.
- *Powells* had large coverage (with three out of four search engines indexing over 40,000 pages) but never had any of their books returned in the top 1000 results for these search engines.

3.4 Search engine/bookstore link coverage

Links are used by search engines when discovering pages to crawl and when ranking pages. Only two of the evaluated search engines supported domain name link counts: **AltaVista** and **Fast**. Domain name link counts gave us the number of links to an entire domain name rather than just links to a single page. Using this information we could determine the link popularity of an entire bookstore.

Table 6 contains the results for this study. From these results we observed that:

- **Fast** observed a large number of links to *Amazon* but did not cover *Amazon* as comprehensively as other search engines.
- *Powells* bookstore had a large number of incoming links, but still performed poorly.
- **Fast** discovered more links to diverse hosts than did **AltaVista**. This could be attributed to the fact that **Fast** performed a deeper crawl and encountered a larger number of internal links.

4 Discussion

The best book finding search engines were **Google** and **MSN Search** and the most successful bookstore was *Amazon*. **MSN Search** provides the most correct answers at position 1. However, **Google** provides more correct answers in the top 5; potentially giving users more book buying options. The following sections discuss why we believe **Google** and **MSN Search** perform better than the other two search engines, and why *Amazon* performance is better than that of other bookstores.

While we believe the concepts from this experiment are relevant to general Web product searches, it is important to note that these experiments are not necessarily reproducible. The algorithms and indexes used in the evaluated search engines and bookstores are not known and may well change over time.

4.1 Search engines

In order to maximize the book finding ability of a Web search engine it seems that the crawler must perform deep crawls (i.e. crawl long URLs) of at least one bookstore. This is because all the examined bookstores bury product pages deep within their URL directory tree (generally as leaf nodes).

Table 5: **Search Engines URL Coverage.** Note that the totals in the right hand side column may contain duplicate URLs. (this occurs when the same URL is found by different search engines.) Also the totals include figures for unlisted bookstores.

Bookstore	AV	FA	GO	MS	TOTAL
amazon.com	3,675,723	358,376	3,620,000	2,838,819	10,492,918
barnesandnoble.com	521	192,792	1,240,000	2,822	1,436,135
walmart.com	89,243	1,076	10,500	916,162	1,016,981
netstoreusa.com	1,171	315,002	93,000	42,052	451,225
powells.com	39,397	111,977	65,900	6,204	223,478
textbookx.com	18	23,157	38,600	150	61,925
alldirect.com	24	26,278	7	27	26,336
ecampus.com	300	7,763	2,010	240	10,313
planetgold.com	18	8,361	774	18	9,171
booksamillion.com	22	5,860	54	865	6,801
cornwalldiscountbooks.com	1	5,423	2	1	5,427
wordsworth.com	735	228	2,290	1,271	4,524
booksite.com	93	169	1,190	290	1,742
codysbooks.com	74	1,308	238	57	1,677
arthursbooks.com	7	1,221	8	384	1,620
samwellers.com	7	278	5	8	298
total	3,809,257	1,087,000	5,081,161	3,810,094	13,787,512

Table 6: **Search Engines Link Coverage.** Note that the totals in the right hand side column may contain several links to the same URL.

Bookstore	AV	FA	TOTAL
amazon.com	12,408,441	25,955,858	38,364,299
powells.com	5,197,526	316,989	5,514,515
textbookx.com	3,456,068	28,453	3,484,521
barnesandnoble.com	234,137	784,088	1,018,225
walmart.com	14,783	267,008	281,791
booksite.com	4,927	113,729	118,656
booksamillion.com	34,137	79,351	113,488
ecampus.com	2,170	102,047	104,217
netstoreusa.com	10,548	91,867	102,415
1bookstreet.com	25,229	50,064	75,293
wordsworth.com	2,750	21,694	24,444
a1books.com	4,545	16,270	20,815
codysbooks.com	1,062	9,512	10,574
alldirect.com	614	6,508	7,122
arthursbooks.com	109	1,700	1,809
samwellers.com	106	208	314
total	21,463,332	28,160,545	49,623,877

While **Fast** appeared to index a much larger selection of bookstores, they appeared to not crawl as much of the *Amazon* bookstore as the other search engines. Considering the majority of correct hits for all search engines came from the *Amazon* bookstore this could have been one of the main reasons for the observed low effectiveness of **Fast** on this task.

To improve crawler coverage of popular bookstores it is also necessary to crawl dynamic URLs with `?` or `cgi-bin`, or even crawling many pages generated from a single script with different parameters. On the *Powells*, *Walmart* and *Barnes and Noble* bookstores, all product pages are created from a single script, with the book's ISBN as a parameter. As many slightly different URLs frequently contain exactly the same information it is also necessary to perform advanced equivalence (duplicate) detection. This is the case with product pages from the *Amazon* bookstore which includes referral identifiers in their URLs.

Finally, **MSN Search** and **AltaVista** use referral information when directing traffic to bookstores but **Google** and **Fast** do not. While not wanting to encourage preferential treatment of search engine partnered bookstores, using a referral id to gain revenue when people purchase books found through your search engine is a potential source of income for a search engine.

4.2 Bookstores

We split up our examination of bookstore performance into two parts: Coverage and Matching/Ranking. Coverage discusses how bookstores may maximise their crawlability. Matching/Ranking discusses reason why some bookstores' books may be returned in preference to others.

4.2.1 Coverage

When consulting Tables 5 and 6 we observed that the top three bookstores by URL coverage were also the top three bookstores by success rate. Having your bookstore featured extensively in search engine indexes appears to be essential for success. *Amazon* achieves high coverage in the indexes of all evaluated search engines.

It is important for a bookstore to have deep crawls indexed in as many search engines as possible. We found three potential reasons why bookstores were not crawled deeply.

1. The bookstores did not effectively convert incoming links into crawled pages.
2. The bookstores did not have sufficient external deep links directly to product pages.
3. The product pages were too deep in the bookstores site hierarchy.

Converting links into crawled product pages is an important consideration. Many bookstores that have a high link count are unable to achieve wide URL coverage. This may be due to bookstore sites appearing to search engines as being dynamically generated. When crawling a dynamic site a crawler may be confused by the site structure. This could be caused by dynamic information in the URL (such as question marks), or by the generation of a series of pages by a single script. In many of these cases a crawler will either simply ignore the link, or be unable to retrieve any meaningful information from it. This appears to occur on *Powells*, which has a large number of incoming links but less indexed pages than other well linked bookstores. The use of a dynamic-to-static web site convertor may go some way to addressing these coverage problems [15]. The site which managed to best convert incoming links to crawled pages was *NetstoreUSA*. In contrast to all other evaluated bookstores, *NetstoreUSA* had more pages indexed by the search engines than they had links. Upon further examination we found that *NetstoreUSA* have static-looking URLs, with a simple hierarchy of `shtml` pages.

To encourage a deep crawl that will cover all product pages it is necessary for bookstores to ensure they have both internal and external links directly to their product pages. To encourage user linking it is important to use meaningful, sensible and consistent URLs for products. While one can envisage a Web developer linking to a URL which has the form `foo.com/ISBN/` it may be less likely that they link directly to `foo.com/prod/prod.asp?prod=9283&source=09834`. Further deep linking may be encouraged through the use of incentive or partnership programs. If such a program is in place it is important to ensure partners are able to point directly to products and that all partners point to the same consistent URL for each product.

4.2.2 Matching/ranking performance

All of the evaluated search engines use some form of link and page content information when matching and ranking pages. It is important, from a bookstore perspective, that a search engine return relevant product pages in response to a query. However, many of the observed search results were browse and search pages. The *Powells* bookstore is a case in point. Despite having many links, reasonable coverage in search engine indexes and having results matched frequently, *Powells* transactional pages were never returned. This may indicate poor page content, site organisation and/or a lack of deep linking directly to products (as their referral program appears to be processed through their front page).

A potential method to alleviate these problems is to use a "robots.txt" to direct crawlers to ignore search and browse pages and only index product pages.

4.2.3 External factors and limitations

We encountered several interesting phenomena we could not explain with certainty.

Some search engines appear to favour certain bookstores over others. (e.g. **Google** and **Fast** have large indexes of *Barnes and Noble* while **MSN Search** and **AltaVista** have large indexes of *Walmart*.) The reasons for these preferences cannot be fully accounted for in this paper as we do not have access to the algorithms employed by the Web search engines. An example of this is the good performance of the *Walmart* bookstore in **MSN Search**. The results suggests that **MSN Search** has access to extra information for *Walmart* that is not available to the other search engines.

Further, we rely on each search engine returning fairly accurate size estimates. We assume that the figures reported are at the very least indicative of the relative coverage.

5 Conclusions

Throughout our experiments we have observed large differences in the performance of bookstores and search engines. The search engines' precision at 10 varied from 0.00 to 0.20, while 24 of the evaluated bookstores did not appear in the top 1000 results for any of the evaluated search engines for any of the books.

Our results have illustrated the importance of a combined approach to improving product search. While search engines should endeavour to discover more product pages it is equally important for bookstores to build a suitable site structure that allows search engines to perform deep, thorough crawls. To improve transactional search effectiveness the search engines should perform deep crawls of provider sites and crawl their dynamic pages (especially those that are linked to by other sites). To improve searchability, bookstores should use short non-changing URLs and encourage deep linking directly to their product pages.

5.1 Further work

In our experiments we only included results that were bookstore book purchase pages. An extension to our study would be to include other types of pages in search effectiveness measures, such as price comparison pages.

Another extension would be to examine how searchability changes according to book properties. It would be interesting to correlate the age of a book with how well it is retrieved by the search engines. One would expect newer books to be less effectively retrieved, as there would be little anchor-text. In this way the current study may set a difficult task since many bestsellers may have only recently been published. However, approximately half the books are paperback versions for which hardcover versions, which have generally been released some time ago, are also considered relevant. Differences in searchability across book genres may also be of interest, we may expect computing books to be more effectively retrieved due to a larger amount of anchor-text.

References

[1] AltaVista. AltaVista search engine, 2002. www.av.com.

[2] FAST Search and Transfer, ASA. Personal communication, 2002. www.alltheweb.com.

[3] Google. Google directory > shopping publications > books > general, September 2002. directory.google.com/Top/Shopping/Publications/Books/General.

[4] Google. Google search engine, 2002. www.google.com.

[5] David Hawking. Overview of the TREC-9 Web Track. In *Proceedings of TREC-9*, 2000. trec.nist.gov/pubs/trec9/.

[6] David Hawking, Nick Craswell and Kathleen Griffiths. Which search engine is best at finding online services? In *WWW10 Poster Proceedings*, Hong Kong, 2001. www10.org/cdrom/posters/1089.pdf.

[7] John B. Horrigan and Lee Rainie. PEW Internet & American life project: Getting serious online, March 2002. www.pewinternet.org/reports/reports.asp?Report=55&Section=ReportLevel1&%Field=Level1ID&ID=241.

[8] Inktomi. MSN search engine, 2002. search.msn.com.

[9] Danny Sullivan. Nielsen//NetRatings search engine ratings. Web Site, September 2002. www.searchenginewatch.com/reports/netratings.html.

[10] Danny Sullivan. Who powers whom? search providers chart. Web Site, September 2002. www.searchenginewatch.com/reports/alliances.html.

[11] Bob Travis and Andrei Broder. Web search quality vs. informational relevance, www.infonortics.com/searchengines/sh01/slides-01/travis.html 2001.

[12] Trystan Upstill. Full bookstore/ISBN query/result list. Web Site. cs.anu.edu.au/~Trystan.Upstill/pubs/results/bstore-rels-02.html.

[13] Yahoo! Yahoo! business and economy > shopping and services > books > booksellers, September 2002. www.yahoo.com/Business_and_Economy/Shopping_and_Services/Books/Booksell%ers/.

[14] New York Times. Bestsellers. Web Site, September 2002. www.nytimes.com/2002/09/01/books/bestseller/.

[15] YourAmigo. Youramigo spider linker, 2002. www.youramigo.com/downloads/documents/SLWhite.pdf.

6 Appendix

6.1 Book Query Breakdown

- (27) Children's
- (15) Hardcover Advice
- (11) Hardcover Business
- (35) Hardcover Fiction
- (29) Hardcover Non-Fiction
- (15) Paperback Advice
- (07) Paperback Business
- (35) Paperback Fiction
- (32) Paperback Non-Fiction
- **(206) Total**

Note: duplicates were removed from the query set. (e.g. Stupid White Men was in both the Hardcover Business and Hardcover Non-Fiction sections, and so was only considered in the Hardcover Business category.)

6.2 Bookstores

Table 7 contains the list of evaluated bookstores.

6.3 Query Submission

- **AltaVista**
  - *General Queries:* Book title surrounded by quotation (") marks.
  - *URL Coverage:* canonical domain name with "url:" parameter.
  - *Link Coverage:* canonical domain name with "link:" parameter.
  - *Timeframe:* General and Domain Restricted Queries submitted between 20/09/02 and 02/10/02 Link Coverage and URL Coverage experiments performed on the 09/10/02.
- **AllTheWeb (Fast)**
  - *General Queries:* Book title with exact phrase box ticked.
  - *URL Coverage:* Advanced search restricting to domain using "domain" textbox with canonical domain name.

Table 7: **Bookstores included in our evaluation.** This table reports whether the bookstore contained ISBNs in its internal URLs (URL), whether they used question marks in internal URLs (Dyn.), whether they were a derivitive of another site (Deriv.) and how many of the 9 book categories they matched (Cat.). A "*" next to the "Deriv." column indicates that the site was a *booksense.com* derivative, while a "+" indicates that the bookstore was a *booksite.com* derivative. A partial in the "Dyn" column indicates that the site was dynamic but didn't "look" dynamic (it didn't have a "?" with parameters following the URL). Note that this table includes all 38 evaluated bookstores. All other tables only consider *Powells* and the 14 bookstores that were returned by search engines at least once.

Bookstore	Core URL	Deriv.	Dyn.	URL	Cat.
1BookStreet	1bookstreet.com	N	Y	ISBN	9
A1Books	a1books.com	N	Y	ISBN	9
AllDirect	alldirect.com	N	Y	ISBN	9
Amazon	amazon.com	N	Partial	ISBN	9
Americana Books	americanabooks.com	N	Y	-	7
Arthurs Books	arthursbooks.com	N	Y	ISBN	4
Barnes and Noble	barnesandnoble.com	N	Y	ISBN	9
BookWorks	bookworksaptos.com	Y*	Y	ISBN	9
BookSite	booksite.com	Y+	Y	ISBN	9
Changing Hands	changinghands.com	Y*	Y	ISBN	9
ecampus	ecampus.com	N	Y	ISBN	9
NetstoreUSA	netstoreusa.com	N	Partial	ISBN	9
Planet Gold	planetgold.com	N	Y	-	9
TextbookX.com	textbookx.com	N	Y	ISBN	9
VStore	vstore.com	N	Y	ISBN	3
Sam Weller's Books	samwellers.com	N	Y	ISBN	9
All Textbooks 4 Less	alltextbooks4less.com	N	Y	ISBN	9
The Book Shop	bookshopmorris.com	Y*	Y	ISBN	9
Cornwall Discount Books	cornwalldiscountbooks.com	N	Y	-	8
A Lot of Books	alotofbooks.com	N	Y	-	3
HearthFire Books	hearthfirebooks.com	Y*	Y	ISBN	9
Walmart	walmart.com	N	Y	-	9
Wordsworth.com	wordsworth.com	N	Y	ISBN	9
Powells	powells.com	N	Y	-	9
BiggerBooks.com	biggerbooks.com	N	Y	ISBN	9
That Bookstore in Blytheville	tbib.com	Y*	Y	ISBN	9
StrandBooks.com	strandbooks.com	N	Y	ISBN	7
St. Marks Bookshop	stmarksbookshop.com	Y*	Y	ISBN	9
RJ Julia	rjjulia.com	N	Y	ISBN	9
Paulina Springs Book Company	paulinasprings.com	Y*	Y	ISBN	9
Books-A-Million	booksamillion.com	N	Y	ISBN	9
CodysBooks.com	codysbooks.com	Y*	Y	ISBN	9
The Concord Bookshop	concordbookshop.com	Y*	Y	ISBN	9
Dartmouth Bookshop	dartbook.com	Y*	Y	ISBN	9
GoodEnough Books	goodenoughbooks.com	Y*	Y	ISBN	9
MediaPlay.com	mediaplay.com	N	Y	-	9
Northshire Bookstore	northshire.com	N	Y	ISBN	8

  - *Link Coverage:* Advanced search using Word Filter with "Must Include" in the preceeding drop down box, canonical domain name in middle text box and "in the link to URL" in the final drop down box.
  - *Timeframe:* General and Domain Restricted Queries submitted between 20/09/02 and 02/10/02 Link Coverage and URL Coverage experiments performed on the 09/10/02.

- **Google**
  - *General Queries:* Book title surrounded by quotation (") marks.
  - *URL Coverage:* Search for the non-presence of a non-existing word (-adsljflkjlkjdflkjasdlfj0982739547asdhkas) and using canonical domain name with "host:" parameter.
  - *Link Coverage:* Not available.
  - *Timeframe:* General and Domain Restricted Queries submitted between 20/09/02 and 02/10/02 Link Coverage and URL Coverage experiments performed on the 09/10/02.

- **MSN Search (Inktomi)**
  - *General Queries:* Advanced search with book title as an "exact phrase box".
  - *URL Coverage:* Advanced search using the domain name as the query, and restricting domain using "domain" textbox with canonical domain name.
  - *Link Coverage:* Not available.
  - *Timeframe:* General and Domain Restricted Queries submitted between 20/09/02 and 02/10/02 Link Coverage and URL Coverage experiments performed on the 09/10/02.

Vector Space Ranking: Can We Keep it Simple?

Vo Ngoc Anh *Alistair Moffat*

Department of Computer Science and Software Engineering
The University of Melbourne
Victoria 3010, Australia

{*vo,alistair*}*@cs.mu.oz.au*

Abstract: *The vector-space model is used widely for document retrieval, based upon the TF-IDF rule for calculating similarity scores between a set of documents and a query. One of the drawbacks of this approach is the need to select a specific formulation for the similarity computation. Here we present an initial attempt to simplify the heuristic, by hiding the various detailed calculations, and evaluating the term importance qualitatively rather than quantitatively. A new technique, called local reordering is introduced. Local reordering still relies on the vector-space model, as it employs a scalar vector product for calculating similarity scores. But there is no longer a requirement for precise values of the document or query vectors to be determined. Initial experiments on two data sets shows that it is highly competitive in terms of retrieval effectiveness. As a useful side effect, the method allows extremely fast query processing.*

Keywords Information retrieval, text indexing, vector-space ranking, similarity heuristic.

1 Introduction

Given a collection of documents $D = \{d\}$ and a query q, we wish to determine the documents in D that are most relevant to q, where "relevance" is judged by human assessors. Statistical methods for approximating human relevance use the TF-IDF rule for firstly calculating similarity scores between each document and the query and then choosing the documents with the highest similarity scores as the answers. The TF-IDF rule [Salton, 1989, Witten et al., 1999] asserts that, for each term t in q, the similarity score must vary as a positively correlated function of $f_{d,t}$, and as a negatively correlated function of f_t, where $f_{d,t}$ is the *within-document frequency* of t in d, and f_t is the *document frequency* of term t in the collection D.

Suppose that N and n are number of documents and number of distinct terms, respectively, in D. The vector space model deploys the TF-IDF rule in the following way. Conceptually, a n-dimensional vector space is constructed, with each dimension representing a term that appears in the collection. In the space, a document d is represented as

$$d = (w_{d,t_1}, w_{d,t_2}, \ldots, w_{d,t_n}),$$

and the query q as

$$q = (w_{q,t_1}, w_{q,t_2}, \ldots, w_{q,t_n}).$$

In this framework, the t_i are the distinct terms of the collection, and $w_{x,t}$ is the projection of document or query x in dimension t. That is, $w_{x,t}$ is the "importance" of t in x, and can be calculated by any formulation obeying the TF-IDF requirement. A similarity score $S(d,q)$ between d and q calculated by the cosine measure is of the form:

$$S(d,q) = \frac{\sum(w_{d,t} \cdot w_{q,t})}{\sqrt{\sum w_{d,t}^2} \cdot \sqrt{\sum w_{q,t}^2}}, \qquad (1)$$

where the three summations are over all n terms.

In mathematical terms, $S(d,q)$ represents the cosine of the angle between the vectors d and q. The greater the value of $S(d,q)$, the smaller the angle between the vectors d and q, and the more "similar" they can be claimed to be. Note, however, that the calculation $\sum(w_{d,t} \cdot w_{q,t})$ alone can be considered to meet the gross requirements of the TF-IDF rule, and the denominator in equation 1 represents a modification of the TF-IDF value that penalizes long documents or queries, and should be thought of as a component of the cosine rule, but not necessarily of the TF-IDF approach. Note also that $W_d = (\sum w_{d,t}^2)^{0.5}$ is usually referred to as the *document length*, and that the corresponding query length $(\sum w_{q,t}^2)^{0.5}$ is constant for a given query and can be ignored.

Zobel and Moffat [1998] explored a range of similarity score variants. They showed that the number of possible variants is large, and that none of them seems to be an absolute winner when retrieval effectiveness is taken as the criteria. They do, however, suggest the use of the mechanism denoted BD-ACI-BCA, which performed well in their experiments. In this formulation, $w_{d,t} = 1+\log_e f_{d,t}$, and $w_{q,t} = (\log_e(1+f^m/f_t)) \cdot (1+\log f_{q,t})$, where f^m is the maximum value of f_t in the collection, and the document length is a normalized function of W_d [Singhal et al., 1996].

The work reported in this paper represents an attempt to escape the need to pick a particular formulation. We introduce a method called *local reordering*, which takes each document in the collection as an (almost) independent scope, and assesses the importance of each term appearing in it without reference to other documents.

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

Our presentation begins in Section 2 with a examination of the use of document lengths in the cosine measure. That discussion leads to our principal hypothesis: that document length is used as a quantitative surrogate for a more direct requirement, that of qualitatively estimating the importance of each term that appears in a document to the "meaning" of that document.

The idea is developed further in Section 3 where we report our initial attempts to avoid the detailed calculation involved in the cosine computation, and return to the simpler estimation implied by the TF-IDF rule. Section 4 then provides experiment results that show the simple approximation to be perfectly adequate in terms of retrieval effectiveness (hence the title of this paper). Implementation issues are discussed briefly in Section 5. Section 6 then finalizes our presentation with some conclusions and directions for future work.

A brief word on notation is necessary. In this work we concentrate on individual documents d rather than the whole collection D, and it is useful to slightly abuse some of the usual notation. In particular, a document d is supposed to have n_d distinct terms $t_1, t_2, \ldots, t_{n_d}$, and $T_d = (t_1, t_2, \ldots, t_{n_d})$ is referred to as the *term list* of d. In any formulation of $S_{d,q}$, the value derived solely from a certain term t and the document d is called *retrieval contribution*, or *contribution* of term t in d, and is denoted by $\omega_{d,t}$. Thus, in equation 1 we have $\omega_{d,t} = w_{d,t}/W_d$. We also refer to $\Omega_d = (\omega_{d,t_1}, \omega_{d,t_2}, \ldots, \omega_{d,t_{n_d}})$ as the *retrieval contribution list* for d.

2 Document length: Keeping it simple

In this section we examine the possibility of avoiding the document length factor in the cosine similarity computation. The key idea is to define term contribution locally within each document, instead of globally in whole document collection.

Consider equation 1. The document length division is intended to penalize long documents, but it is a rather blunt instrument, and also creates a bias in favor of short documents. That phenomenon was noticed by a range of investigators, and there have been several attempts to ameliorate it. Most recently, Singhal et al. [1996] and Chowdhury et al. [2002] normalized the value of document lengths and were able to significantly improve retrieval effectiveness. Anh and Moffat [2002] obtained further improvement by normalizing the global set of term contributions.

It is possible that further tweaking with the document length might lead to continued incremental gains in retrieval effectiveness. But it is also attractive to think about completely removing it as a factor. In their study Zobel and Moffat [1998] explored mechanisms using a unit document length, but without obtaining competitive effectiveness.

Here we propose a partial removal. Instead of normalizing each term weight by the document length, the term contribution list Ω_d of each document d is re-evaluated in the context of that document so that occurrences of a term may well be treated differently in short documents compared to long ones. For any d, the re-evaluation is done locally in d, without consulting any term contribution values in any other document. This re-evaluation is thus characterized as *local*. By way of contrast, the impact transformation scheme given by Anh and Moffat [2002] is also a re-evaluation, but global.

Now consider Ω_d again. The largest value of Ω_{d_1} in a document d_1 might still be small compared to the largest value of Ω_{d_2} in document d_2. The intention of the local re-evaluation process is to adjust the values in Ω_d so that, over the set of documents D, the largest and smallest values of Ω_d for each document are roughly the same. As in our previous work [Anh et al., 2001, Anh and Moffat, 2002], the contributions are also quantized, so that each *surrogate weight* is used as the retrieval contribution $\omega_{d,t}$ for a whole set of terms. In essence, in each document d the largest component in Ω_d is mapped to a pre-determined integer k, and each other $\omega_{d,t}$ value is represented by an integral surrogate weight in the range 1 to k. Suitable values for k are discussed shortly.

Once it is accepted that it is the relative ordering of $\omega_{d,t}$ values that is important, rather than their actual numeric values, there is no need to persist with a formulation for document length. Instead, document-term weights are calculated as $\omega_{d,t} = w_{d,t}$, and mapped onto the set of surrogates. Requiring that each document d use the full range of k surrogate values guarantees a kind of length-based normalization process, but one in which there is no cross-talk between documents.

Four mapping methods are considered here.

- *By-Value*: The first method is based on the value of the mapped elements. Initial experiments showed that the distribution of $w_{d,t}$ for a single document d is similar to that of the impacts over the whole collection (see [Anh et al., 2001]). That is, for a certain d, the number of high values is much smaller than the number of low values. Following the lead of [Anh and Moffat, 2002], we map the value w to

$$\left\lfloor k \cdot \frac{\log w - \log U_d}{\log U_d - \log L_d + \epsilon} \right\rfloor + 1 \,, \qquad (2)$$

  where U_d and L_d are the maximum and minimum, respectively, of the values $w_{d,t}$ in document d. Note that the use of ϵ ensures that none of the quantized values is higher than k. Note also that the mapping is local, and ensures that surrogate weights of both 1 and k are generated on non-trivial documents.

The other three mappings are based on a strategy called *By-Rank*, in which the fraction of the elements in Ω_d that share each surrogate weight is determined in advance. It seems clear that surrogate weight k should not be used with a greater frequency than surrogate weight 1. Taking x_i to be the number of elements assigned a surrogate weight of i (where i is between 1 and k inclusive), three variants have been considered:

- *By-Rank, Geometric*: The number of elements in a document corresponding to each surrogate weight, in decreasing order of the weights, forms a geometric subsequence. That is, $x_i = x_{i+1} \cdot B$, with B determined in similar way given in equation 2 with L_d and U_d replaced by 1 and n_d respectively.

- *By-Rank, Arithmetic*: The number of elements in a document corresponding to each surrogate weight, in decreasing order of weights, forms a arithmetic subsequence. In particular, $x_i = x_{i+1} + B$, where $B = (n-1)/(k \cdot (k-1))$.

- *By-Rank, Uniform*: Within a document each surrogate weight corresponds to approximately the same number of elements.

For example, when *By-Rank* is combined with the *Uniform* method, approximately n_d/k of the n_d distinct terms in document d are assigned to each of the surrogate ranks from 1 to k.

3 TF-IDF: Keeping it simple

The approach described in the previous section eliminates one of the three parameters in the cosine measure, by removing the need to choose a document length normalization regime. Nevertheless, there are still two factors remaining. One way of trying to establish these would be to repeat a suite of experiments of the kind performed by Zobel and Moffat [1998].

Instead, we prefer to again just make a simplifying assumption, and revert to the underlying basis espoused in the TF-IDF rule.

In its original form, TF-IDF is not a precise definition – it is a philosophy. It is our human desire for precision in computation that has led to overly precise formulations, with their various tuning factors. So, in keeping with the spirit of the "rank is more important than value" claim of the previous section, this section explores simple rules that order the term contributions in way that is consistent with the TF-IDF philosophy. Ideally, we would like to create a sorted list of terms corresponding to each document without doing any detailed computation. The surrogate weights of the n_d terms in document d are then computed for use in the query processing regime, without any further recourse to collection or document statistics.

Again, we focus on one document at a time, and divide the process into two phases. The first *sorting* phase orders the term list T_d of document d in decreasing order of term contribution. In the second *mapping* phase, each value in the term list T_d is converted to an integer in the range 1 to k. As the end of the process, these integer values serve as term contributions, and form the list Ω_d.

There are many ways of ordering the term contributions in the sorting phase. For example, any cosine formulation could be used to calculate a numeric sort key. But our overriding desire in this work is to "keep it simple", and we eschew formula-driven orderings in favor of more primitive methods. In particular, the following variants are explored in the experiments described in Section 4:

- (*IDF*, *TF*): The term list is sorted in increasing order of f_t, with ties on f_t broken by using decreasing order on $f_{d,t}$ as a secondary key. In this arrangement, the *IDF* component is presumed to dominate the *TF* component; the nomenclature reflects the lexicographic sort ordering.

- (*TF*, *IDF*): The *IDF* component is a statistic of the collection, rather than of a particular document. The (*IDF*, *TF*) method, in that sense, does not reflect our "keep it simple" theme. From the point of view of a single document, the *TF* component should dominate. In this arrangement, the term list is sorted in decreasing order of $f_{d,t}$, with ties broken by using increasing order on f_t as a secondary key.

- (*TF*, *IDF*, *stopped*): One possible drawback of the (*TF*, *IDF*) proposal is that it emphasizes common words in English that are devoid of meaning, such as "the", "and", and "at". These words tend to have high $f_{d,t}$ values, and get mapped by the (*TF*, *IDF*) ordering onto the highest surrogate weights. To correct for this anomaly, the (*TF*, *IDF*, *stopped*) mechanism resets the $f_{d,t}$ value of a set of *stop words* to 1, to guarantee their positions at the tail of each sorted list. There is, however, a consequent issue as to how to decide whether a word should be stopped [Fox, 1992]. To keep it simple, here we classify words appearing in 20% or more of the documents (that is, $f_t \geq N/5$) as being stop words.

The number of different f_t values is high in any non-trivial document collection, so in any of these three sorting rules there is only a small chance for two different terms in the same document have the same sort key value.

A fourth method is also included in the experiments described shortly:

- (*TF* $\times$ *IDF*): The value $w_{d,t}$ calculated by the BD-ACI-BCA similarity computation is used as the sort key, with the term list sorted into decreasing order.

Once the term list has been sorted, it is partitioned using one of the four mechanisms described in Section 2, except that the *By-Value* partitioning process can only be coupled with the (*TF* $\times$ *IDF*) ordering. The resulting surrogate weights are used as the basis of a similarity computation.

4 Experiments

The aim of the first experiments is to ensure that the reported approaches result in acceptable retrieval effectiveness, and to choose appropriate parameters or methods. The dataset *WSJ2* – a subset of *Disk2* of the *TREC*

corpus [Harman, 1995], is employed here for that purpose. The collection *WSJ2* is a homogeneous text collection, containing the text of the *Wall Street Journal* for the period 1990–1992. The collection has around 75,000 documents totaling approximately 240 MB. To measure effectiveness, 150 queries were formed by taking the "title" fields of *TREC* topics 051–200.

To control the complexity of the experimental process, a set of initial experiments was undertaken with the *By-Rank* process to determine the best of the three alternative partitioning regimes. As discussed in Section 2, the three mechanisms are *Geometric*, *Arithmetic*, and *Uniform*. They are similar in many ways to the methods discussed by Anh and Moffat [2002], and the preliminary experiments led to the expected results – *Geometric* outperforms *Arithmetic*, and *Arithmetic* outperforms the *Uniform* method. Those results will appear in a more detailed version of this paper. In the remainder of this presentation, whenever a mapping is used to convert a sorted list of objects into a set of integers, the *Geometric* method is employed.

A second set of experiments was then undertaken using the various proposed methods to sort the term list of a document. The BD-ACI-BCA cosine mechanism, which includes document-length pivoting, was used as a baseline in these experiments, and is denoted in the results by "cosine, baseline". The same similarity formulation, but without document-length normalization, and thus without pivoting, was used as the basis of the $w_{d,t}$ values required by the *By-Rank* and *By-Value* surrogate assignment mechanisms. In the results in Figure 1 these two are denoted as "(*TF* $\times$ *IDF*), *By-Rank*" and "(*TF* $\times$ *IDF*), *By-Value*" respectively.

Note that the decision to continue only with the *Geometric* partitioning means that the value of k (the number of distinct surrogate weights) is the only tunable parameter that remains; and that the retrieval achieved by the "baseline" mechanism is independent of k.

Figure 1 plots retrieval effectiveness as a function of k for the five methods, and compares them to the BD-ACI-BCA baseline. Three different metrics for assessing retrieval effectiveness are shown. The most reliable metric is average precision [Buckley and Voorhees, 2000]. However, precision at 10 documents retrieved and reciprocal rank are also included, as they are important for many people, especially in situations in which number of identified-as-relevant documents is unknown – when web searching, for example.

To rank the documents with respect to a query, the same process was applied to each query as was applied to each document – namely, the terms in the query were ordered using the same rule as the terms in the documents; and then the ordered terms were mapped to surrogate weight values using the same mapping as was used for the documents. The inner-product of the two resulting integer vectors was then calculated, and used as a similarity score without any further adjustment.

There are a number of interesting points that can be drawn from the results.

First, except for the (*IDF*, *TF*) term ordering regime, all of the methods using surrogate weights are capable of giving excellent retrieval performance and of outperforming the BD-ACI-BCA baseline by a compelling margin. Indeed, the good methods have surprisingly similar effectiveness curves when plotted as a function of k, and all are relatively stable when k is greater than about 6. Of the four good methods, (*TF*, *IDF*, *stopped*) has a slight edge.

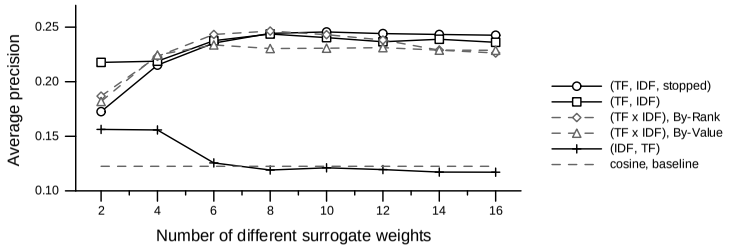
Second, it should be noted that even with $k = 2$ the new methods give better effectiveness than the baseline. This outcome was completely unexpected, and means that just one bit can be used to code term importance, instead of the several bits required in conventional cosine implementations to store the within-document frequency value associated with each document pointer [Witten et al., 1999]. The two different surrogate weights in the binary case can be interpreted as "high relevance" and "some relevance" of the term to the document.

Third, (*IDF*, *TF*) performs poorly in all situations. This outcome is also somewhat surprising, since at face value it means that the *IDF* factor performs only a minor role in determining the contribution made by a term. One possible reason for the poor performance is that numerically the *IDF* takes on many more distinct values than does the *TF*, and so there is less opportunity for *TF* to be used in a tie-breaking role. On the other hand, if the primary sort key is given by *TF*, then the *IDF* is likely to be used relatively often to break ties. That is, in (*TF*, *IDF*) both of *TF* and *IDF* are likely to influence the ordering more often than they do in (*IDF*, *TF*). If this is the case, then there may be benefit to be gained by quantizing either or both of the *TF* and *IDF* components before applying the ordering step.

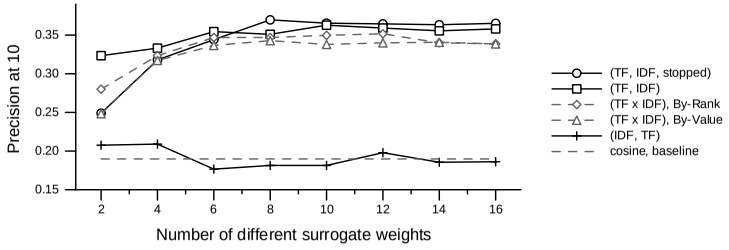
An issue that may be puzzling the reader is that in the (*TF*, *IDF*) regime, almost certainly the highest surrogate weights are assigned to the most common words; yet this does not appear to impact retrieval effectiveness. The key to understanding this paradox is to note that if a word such as "the" is assigned the same surrogate weight of k in every document in the collection, then appearance of "the" in a query (or non-appearance) has no net effect on the eventual ranking, since the similarity score for every document is adjusted in the same way. That is, the sheer ubiquity of common words ensures that they have only a small impact upon the document ranking.

The third set of experiments takes the best method – ordering mechanism (*TF*, *IDF*, *stopped*), with *Geometric* partitioning, and $k = 10$ – and compares it to other retrieval heuristics on a different document collection, against which a wide range of retrieval mechanisms have been applied.

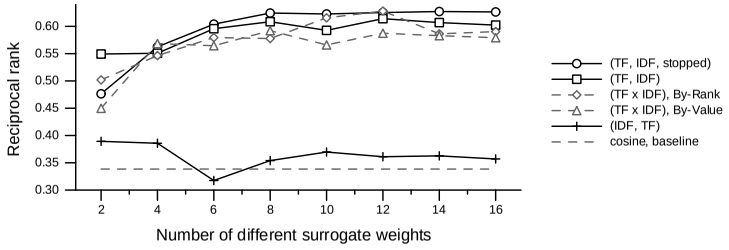
For this phase of the experimentation, collection *wt10g* is employed, as was used in *TREC-9*. The collection contains around 1.6 millions documents with a total size of about 10 GB. The documents were



(a) Average precision.



(b) Precision at 10.



(c) Reciprocal rank.

Figure 1: Retrieval effectiveness as a function of the number of distinct surrogate weights used, using three different effectiveness metrics, and dataset *WSJ2*. The queries are taken from the title field of *TREC* topics 051–200, and effectiveness values are averaged over the set of queries.

crawled from the Internet and can be considered as heterogeneous. The collection is accompanied by 50 ad-hoc queries (topics 451–500). The title fields of these topics are taken as queries. Hawking [2001] and Soboroff [2002] give information about the dataset *wt10g*.

The results of the third experiment are summarized in Figure 2. The effectiveness of the new mechanism is compared with the baseline implementation of BD-ACI-BCA, and with the performance reported by Anh and Moffat [2002] for their impact-transformation technique. The new method also performs well on this data set. Note that with the same data set, impact transformation was shown to perform well compared with other methods participating in *TREC-9* [Anh and Moffat, 2002], including systems employing additional heuristics such as relevance feedback, query expansion, and phrase indexing. (However we also note that it is inappropriate to include *TREC-9* results in Figure 2, as we manually edited the queries to correct spelling mistakes, whereas the *TREC* systems were permitted to only employ automatic correction techniques.)

5 Implementation and efficiency

Local reordering is straightforward to implement, and represents a relatively minor variation from the impact sorted indexes we have described previously [Anh et al., 2001, Anh and Moffat, 2002]. Each index list contains k blocks of sorted document numbers, and can be represented using the standard integer coding techniques [Witten et al., 1999]. Query processing is performed using a

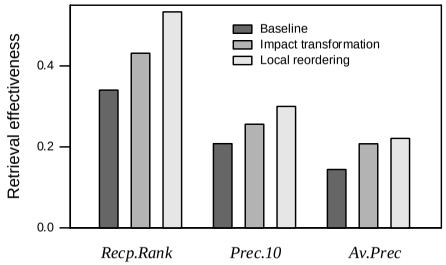


Figure 2: Relative comparison of effectiveness performance of three approaches: the standard BD-ACI-BCA cosine measure (as a baseline), the impact transformation mechanism described by Anh and Moffat [2002], and the local reordering technique described in this paper, for the collection *wt10g*. The three metrics shown are average values calculated over 50 queries taken from *TREC* topics 451–500, with spelling mistakes manually corrected. For the local reordering approach, the number of different surrogate weights is $k = 10$, the sorting method is (*TF*, *IDF*, *stopped*), and the *By-Rank* partitioning method is *Geometric*.

look-up table as a priority queue, and because all of the value manipulated are small integers, is fast.

To date we have not explored any pruning heuristics, and the results presented in Section 4 are for full evaluation. An extended version of this paper will investigate pruning operations, and quantify the tradeoff they offer between execution time and retrieval effectiveness.

6 Conclusion

Local reordering has three distinct benefits. First, it is largely free of the tunable parameters and "knobs" that plague other similarity heuristics. Our intention throughout was to "keep it simple", and we have succeeded in that aim.

Second, it achieves excellent retrieval effectiveness. In the experiments conducted to date, the local reordering matches the best systems that contributed to the *TREC-9* experiments in 2001. Experiments to validate this claim in other test environments are currently being planned.

Third, it allows an extremely efficient implementation. All query-time operations are on small integers, and query processing is rapid. There is little or no overhead cost in terms of index space.

As well as further experiments with the current implementation, we plan to incorporate a number of other "simple" extensions, including the appropriate indexing of anchor text and any meta-data, and perhaps the use of a simple rules based upon term location and context to adjust the surrogate weights. With these changes we hope to create a simple, efficient, and effective, web searching tool.

Acknowledgement This work was supported by the Victorian Partnership for Advanced Computing and the Australian Research Council.

References

V. N. Anh, O. de Kretser, and A. Moffat. Vector-space ranking with effective early termination. In W. B. Croft, D. J. Harper, D. H. Kraft, and J. Zobel, editors, *Proc. 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 35–42, New Orleans, LA, September 2001. ACM Press, New York.

V. N. Anh and A. Moffat. Impact transformation: Effective and efficient web retrieval. In M. Beaulieu, R. Baeza-Yates, S. H. Myaeng, and K. Järvelin, editors, *Proc. 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–10, Tampere, Finland, August 2002. ACM Press, New York.

C. Buckley and E. M. Voorhees. Evaluating evaluation measure stability. In N. J. Belkin, P. Ingwersen, and M.-K. Leong, editors, *Proc. 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 33–40, Athens, Greece, September 2000. ACM Press, New York.

A. Chowdhury, M. C. McCabe, D. Grossman, and O. Frieder. Document normalization revisited. In M. Beaulieu, R. Baeza-Yates, S. H. Myaeng, and K. Järvelin, editors, *Proc. 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 381–382, Tampere, Finland, August 2002. ACM Press, New York.

C. Fox. Lexical analysis and stoplists. In W.B. Frakes and R. Baeza-Yates, editors, *Information Retrieval: Data Structures and Algorithms*, chapter 7, pages 102–130. Prentice-Hall, Englewood Cliffs, New Jersey, 1992.

D. K. Harman. Overview of the second text retrieval conference (TREC-2). *Information Processing & Management*, 31(3):271–289, May 1995.

D. Hawking. Overview of the TREC-9 Web Track. In E. M. Voorhees and D. K. Harman, editors, *The Ninth Text REtrieval Conference (TREC-9)*, pages 87–102, Gaithersburg, MD, November 2001. NIST Special Publication 500-249.

G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, Reading, Massachusetts, 1989.

A. Singhal, G. Salton, M. Mitra, and C. Buckley. Document length normalization. *Information Processing & Management*, 32(5):619–633, 1996.

I. Soboroff. Does WT10g look like the web? In M. Beaulieu, R. Baeza-Yates, S. H. Myaeng, and K. Järvelin, editors, *Proc. 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 423–424, Tampere, Finland, August 2002. ACM Press, New York.

I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann, San Francisco, second edition, 1999.

J. Zobel and A. Moffat. Exploring the similarity space. *ACM SIGIR Forum*, 32(1):18–34, Spring 1998.

A Framework for Text Categorization

Ken Williams
Web Engineering Group
The University of Sydney
Bldg J03, Sydney NSW 2006
kenw@ee.usyd.edu.au

Rafael A. Calvo
Web Engineering Group
The University of Sydney
Bldg J03, Sydney NSW 2006
rafa@ee.usyd.edu.au

Abstract

In this paper we discuss the architecture of an object-oriented application framework (OOAF) for text categorization. We describe the system requirements and the software engineering strategies that form the basis of the design and implementation of the framework. We show how designing a highly reusable OOAF architecture facilitates the development of new applications. We also highlight the key text categorization features of the framework, as well as practical considerations for application developers.

Keywords Document Management, Text Categorization, Application Frameworks

1 Introduction

Automatic Text Categorization (TC) has been an active research area for over a decade and is increasingly being used in the development of commercial applications. These commercial applications usually belong to one of two system types: in-house systems implemented in order to solve a particular company's specific problems, and generic systems marketed to corporations as ready-made categorization solutions. The former tend to be ad-hoc solutions not suitable for use by others, and not made publicly available. The latter tend to be proprietary, closed-source, expensive solutions inaccessible to individuals, small companies, and researchers.

One result of this situation is that many techniques and design strategies are underdeveloped as they are not well-known to research or application communities. Systems such as Weka [14] or Libbow [7] are widely used by the research community, but tend not to focus on integration into real-world applications. By contrast, the commercial systems are often useless for research because they are closed-source, generalize poorly to new problems, or cost more than most researchers can afford. Therefore, researchers do not get the benefit of leveraging industry's TC applications, and industry doesn't get the benefit of the latest developments and knowledge from the research community.

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

It is our aim to create first-rate customizable tools for Text Categorization that apply equally well to the problems of industry and research. Our tools should also be accessible to the casual or small-time developer interested in TC. To accomplish this, we have implemented a framework for Text Categorization.

Before discussing the details of the framework, we will briefly look at some general background on frameworks. Different software engineering architectures are used for different sets of requirements. The most common kinds of software architectures include:

Applications Application developers focus on improving internal reusability and interfacing with users. Developer or user extensibility need not be considered–the application is considered complete as delivered. A popular example of a classification application is the Weka Machine Learning system [14].

Toolkits and libraries Library developers focus on generic reusability for multiple applications. Examples include the mathematical or networking libraries that exist for most programming languages. The "bow" library [7] is an example from TC. Developers who use a library do not have to learn its internal architecture, and the library does not dictate the structure of the application under development.[4] The internal implementation of the library is considered to be hidden from its users.

Frameworks A framework is a set of classes that embodies an abstract design for solutions to a family of related problems [4, Ch. 2]. Framework designers focus on applicability to a certain set of problems, and on flexible best-practices embodied in software. An "inversion of control" puts the framework in charge at a high level inside the application, with custom application code playing a

subordinate role–therefore, interfaces between framework classes must be documented and stable. Common examples of frameworks include generic application frameworks like Apple's "Cocoa." Weka may also be considered a framework when it is used to implement new categorization algorithms through subclassing.

Before deciding on one of these approaches it is important to define the main user audience for text categorization systems in order to determine requirements for a useful TC system. We see typical TC users in terms of the following roles:

Application Developer A professional such as a web developer or engineer that needs to add automatic categorization features to a software application. The application developer may have no prior experience with Text Categorization. The end user may have varying degrees of control over the categorization process.

Researcher A TC researcher interested in novel approaches to machine learning or document processing. This professional is often not interested in implementing a real world application, but wishes to improve existing TC algorithms and methodologies.

Domain Expert Complex applications often require a domain expert who dictates project requirements and has expertise in the application domain (e.g. financial documents, knowledge management). The domain expert often makes high-level decisions about when TC could be effective in the given domain, and needs to exert fine control over the TC process.

Of course, one person may play several of these roles simultaneously.

A researcher will most often want to use a TC system as a framework, because they need to integrate custom code into the system at a low level. A researcher may also find it convenient to use a TC system as an application which provides a convenient user interface for running common kinds of experiments. By contrast, an application developer may want to use a TC system as a library or set of libraries, providing no custom code of his or her own.

Given these requirements, we decided to implement our software as a framework rather than as an application or set of libraries. One reason for this is that a framework can easily be turned into an application by providing simple wrapper code, and it can be turned into a library by providing concrete implementation classes. However, libraries and applications can not typically be turned into frameworks very easily. Therefore, a framework provides the best coverage for the perceived needs of the TC community.

The framework described in this paper includes classes for managing documents, collections of documents, categorization algorithms, and so on. The core framework includes both concrete classes like "Naïve Bayes Learner" which may be used without custom development, as well as abstract classes like "Boolean Learner" which require the user to implement certain behaviours before using them. Abstract classes provide a starting point and an interface for new development and reduce repeated work.

2 Design Requirements

A framework must be able to accommodate functionality in a number of essential areas, providing common behaviour while allowing users and developers to customize behaviour through configuration parameters and/or framework subclassing. We summarize the design issues in this framework as follows. Note that some of these issues are general framework design issues, while others are more specific to this particular domain.

Framework reusability The main reason for building a framework rather than a single text categorization application is to increase reusability of design and implementation. Framework research literature provides guidelines on building application frameworks.[4]

Modularity The components' internal implementations should be able to change without affecting the other components.

Integration The framework should be able to interface easily with existing categorization solutions (e.g. Weka, libbow, various Neural Net libraries, and feature selection packages), uniting many solutions under a common interface.

Rapid Application Development Prototyping new applications should be very quick, with a minimum of custom code in each case. Custom code should generally implement new behaviors rather than new structures within the framework.

Rapid Research Cycle Researchers should be able to quickly investigate new questions, using the framework as a starting point.

Model Flexibility The framework structure should be flexible enough to accommodate the needs of many different categorization algorithms that may operate on different representations of the underlying data.

Computational Efficiency The data sets involved can be quite large, so it is important to have a design and implementation that is efficient in memory, CPU time, and other practical measures such as the time it takes to load a categorizer from disk and generate a hypothesis.

Separability Pieces of the framework should be usable in isolation for users that only need a feature selection package, a vector categorizer, etc. The most separable pieces of the framework should in many cases be completely separated and available under separate distribution, and used as a software dependency in our framework.

With the above issues in mind, we have chosen to implement the framework using the Perl programming language. [13] A vast number of Perl modules are freely available for many different tasks, which extends the domain of applicability for the framework. Many of these modules are tools for processing text, and can be used by the framework. Perl is widely used, multi-platform and integrates well with other languages, so it enables fast prototyping. Perl is also natively object-oriented, with a very flexible object model.[2]

In Perl, the basic unit of reusable code is called a *module*; our framework is implemented as the `AI::Categorizer` module.

3 Functional Areas

The framework supports several functional areas of Text Categorization. We describe them here together with the tradeoffs and design decisions that may be useful to other researchers developing TC systems.

Figure 1 shows the architecture of the framework. Attributes and methods of each class have been removed for the sake of brevity. Each of the classes will be discussed in the context of their Text Categorization function. `Categorizer` is the top level class, which manages the data-related classes (`KnowledgeSet`, `Collection`, `Document` and `Category`), as well as the machine learning `Learner` classes and `Hypothesis`, and a class for reporting the results.

Data format

Since documents come in a wide variety of formats such as XML, plain text, or PDF, the framework should support the importing of knowledge in several formats and have a mechanism by which the user may extend these capabilities for a particular environment. The base class `Document` allows the user to specify the content as a string. The user

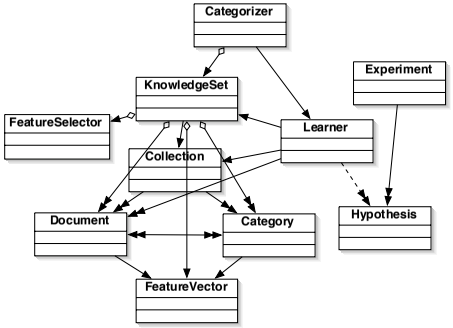


Figure 1: Simplified UML class diagram for the framework

may also subclass the `Document` class, overriding the `parse()` method for direct importing of data in its natively stored format.

In the `Collection` class and its subclasses, the framework also supports the notion of a collection of stored documents, such as a directory of text files, a database of stored documents, or an XML file containing multiple documents. The most common storage formats can be a part of the core framework, while proprietary or unusual formats can be implemented through subclassing. Note that the document format and collection format are independent characteristics; a project may have a directory of text files, a directory of XML files, or a directory of PDF files, but these would all be handled by the `Collection::Files` class with the appropriate `Document` subclass. Likewise, a project may have a collection of XML documents stored in a single file, as a directory of files, or in a database, but these would all be parsed by the `Document::XML` class with the appropriate `Collection` subclass. Note that `Document` and its subclasses exist mainly for the purpose of importing data; after the data is read and parsed, the rest of the system will throw away most of the information in the `Document` object, keeping only its `FeatureVector` object and the list of `Categories` associated with the `Document`.

Structured documents

Each document may have several sections of content, such as "body", "subject", "signature", and so on. In `AI::Categorizer`, the user specifies the content by providing a hash of key-value pairs, where the key indicates the name of the section, and the value is a string containing the content data. The user may also specify "weights" to assign to the features found in each section. In the future, other treatments for the different sections of a document may be supported as we develop effective ways to use this structure.

Tokenizing of data

The default implementation tokenizes document data by extracting all non-whitespace byte sequences between word-character boundaries. This is usually sufficient in English, but non-English language documents or documents with unusual content will certainly necessitate custom tokenization. To achieve this, the user may subclass the `Document` class and override its `tokenize()` method if a different algorithm is required. We may also add other tokenizing options to the default implementation, controlled by parameters, if other common tokenizing needs are found.

Linguistic stemming

The default implementation provides support for the Porter stemming algorithm, a standard algorithm for removing morphemes from English words to obtain their "stems," or root forms. By default no stemming is performed, but a `stemming` parameter can be set to `porter` to activate stemming. Alternatively, the user may override the `stem_words()` method of the `Document` class for custom stemming. This may be extremely important in highly morphological languages or in certain application domains.

Feature selection

Feature selection is handled by the abstract `FeatureSelector` class and its concrete subclasses. These classes implement `scan_features()` and `select_features()` methods. The `select_features()` method works on an entire `KnowledgeSet` in-memory at once. The `scan_features()` method can scan a collection of documents for the best features without necessarily loading the entire collection into memory. Both methods return a `FeatureVector` object to the client (typically a `KnowledgeSet`), which saves the list of highest-ranking features to use when parsing future documents.

The default implementation uses a simple Document-Frequency criterion for selecting features to use in model-building and categorization. This is very efficient, and has been shown in [16] to be competitive with more elaborate criteria in many common situations. We will add more criteria as the project develops.

Vector space modeling

The full range of TF/IDF weighting from [11] are supported, controlled by a `tfidf_weighting` parameter. If the user wants to employ a different weighting scheme, the `weigh_features()` method in the `KnowledgeSet` class may be overridden.

Machine Learning algorithm

Choosing a machine learning algorithm is done by choosing a subclass of the `Learner` class. Several algorithms have already been implemented including Naïve Bayes [6], Support Vector Machines [12] [3], Neural Networks [1] [15], k-Nearest Neighbors [15], and Decision Trees [9]. Any `Learner` class needs to implement the virtual methods `create_model()` and `get_scores()`, which supply the semantics behind the `train()` and `categorize()` methods, respectively. Since many Machine Learning algorithms are implemented as a series of binary decisions concerning individual category memberships, an abstract `Learner::Boolean` class is provided to help developers of new categorizers–in this case, one need only implement the smaller `create_boolean_model()` and `get_boolean_score()` methods.

Note that the `Learner` class does dual duty as a learner and a categorizer. No class distinction is made in the framework between a `Learner` before and after it has been trained–they are objects of the same class. This allows for the possibility of on-line learning, in which a trained learner incrementally uses additional training examples to improve its current model.

Machine Learning parameters

Because each ML algorithm may have several implementation parameters to control behavior, each `Learner` subclass accepts different parameters. To facilitate the wide variety of parameters that different classes may require, we use the `Class::Container` module[1]. This module allows each `Learner` subclass to declare the parameters it accepts, so that a Neural Network class can declare arguments for number of input, hidden, and output nodes, a k-Nearest Neighbor class can declare arguments for k and for thresholding strategies, and so on. These parameters are passed through the framework transparently using a variation on the "Factory Method" pattern. [5]

In fact, the `Learner` and its subclasses are not the only pieces of the framework in which varying parameters control operations. Because this situation is common throughout the framework, `Class::Container` is employed consistently for all structural classes in the framework. This goes a long way toward reducing the number of classes necessary to implement varying behavior.

Hypothesis behavior

Certain applications (e.g. newswire categorizers) may need to find "all categories that apply" for each document, whereas other applications (e.g.

[1] available at http://search.cpan.org/author/KWILLIAMS/Class-Container-0.08/

automatic email routers) may only be interested in the "best N categories," where N is often 1. These scenarios are supported by the Hypothesis class, which provides a generic interface to the scoring decisions of the categorizers. Methods like `categories()`, `best_category()`, and `in_category()` provide application-level access to categorization decisions based on the scores assigned by the `Learner` class.

On-line training

Some machine learning algorithms can easily integrate new knowledge into the knowledge base without going through the potentially expensive process of re-training the categorizer from scratch. For instance, most kNN implementations can do this, whereas most Neural Network implementations cannot. For categorizers that support this, a virtual `add_knowledge()` method in the `Learner` class is supplied. Currently no `Learner` subclasses in `AI::Categorizer` support on-line learning, but the architecture supports it when an implementation is needed.

4 Framework Customization

Like C++ and Java, Perl is natively object-oriented, but unlike them it does not have strict separation of compilation and execution stages. Rather, the compiler and interpreter work in tandem, trading back and forth to execute a Perl application, allowing runtime compilation of code. In addition, Perl's object model is fairly loosely bound (similar in this respect to Objective-C's model), permitting class names to be stored in variables and/or specified at runtime. Because of these properties, the choice of specific classes to be used in the framework can be made at runtime, controlled by parameters, facilitated by the `Class::Container` module. It allows several classes to cooperate as a framework without having to know about each others' class names, constructor parameters, and so on, and provides the glue to do strict early checking of parameter names and types, facilitating transparent factory patterns within the framework.

For instance, to use the built-in SVM learner, one could either create an `AI::Categorizer::SVM` object directly, or one could specify the class name by providing it as a value for the `learner_class` parameter. This behavior is implemented at the framework level, so different `Document`, `Collection`, `FeatureVector`, etc. classes can be pressed into service by the `document_class`, `collection_class`, and `feature_vector_class` parameters, respectively. This helps facilitate quick architectural changes, letting developers drop their own subclasses into the framework with relative ease.

5 Evaluation

Although the focus of this paper is the framework discussion and design, we present here some basic evaluation of its performance. We have evaluated our framework by building classifiers in several applications. We have implemented Naïve Bayes, Support Vector Machine, k-Nearest-Neighbor, and Decision Tree classifiers in the framework. We have trained classifiers using the standard Reuters ApteMod corpus and obtained similar results to the ones described in [15]. We have also trained and tested classifiers on other corpora in financial, educational, and discussion group domains. Due to space constraints and the proprietary nature of some of our other corpora, we will only describe results on the Reuters ApteMod corpus here, using the Naïve Bayes algorithm.

In training categorizers, we typically use two passes through the corpus when loading the data. The first pass scans the documents in order to perform feature selection, while the second pass actually loads the data into memory. This allows memory to be used more effectively than if we only made one pass over the data, because we avoid loading extraneous features. On the Reuters corpus, using Porter stemming [8] and a standard list of stopwords [10], the first pass over the 7769 training files may take roughly 59 CPU seconds and consume 11 MB of memory, while the second pass takes about 57 CPU seconds and consumes 32 MB. The memory figures reflect the total size of a running program, not just the size of the document data in memory.[2]

After the data is loaded, we pass it to a Learner object for training. Our Naïve Bayes training process takes 8.1 CPU seconds and consumes 40 MB of memory. Categorizing the 3019 test documents takes about 95 CPU seconds and consumes 14 MB.

With experimental settings similar to the ones described in [15] (we used Document Frequency feature selection, since we have not yet implemented χ^2 or Information Gain selection algorithms), we achieve recall, precision, and F_1 scores of 0.724, 0.851, and 0.782 when micro-averaged, and 0.366, 0.497, and 0.396 when macro-averaged. We believe any discrepancies with [15] are due to differences in feature selection and/or document tokenizing, but we have not tested this belief thoroughly.

6 Integration and Further Work

The framework has been used in a number of applications including an extension to the SQL language of the PostgreSQL relational database. It has also

[2] Tests were performed on a machine with a Pentium III 800Mhz chip, running Red Hat Linux release 7.0 and Perl 5.6.1. Results are not comparable across different architectures, but may be useful as a rough guide.

been used as distributed service for classification using an XML/RPC architecture, and integrated into multi-tier web applications and desktop applications.

We know that much work has been done by previous developers and researchers in the area of Text Categorization. While we are in one sense retreading ground by implementing generic TC software, we see our work as a way to extend the reach of others' work, rather than as a replacement for it.

To this end, we have tried to make the framework very inter-operable and provide interfaces to existing TC products. For instance, we have implemented a `Learner` subclass called `Learner::Weka` to provide an interface to any Weka classifier the user would like to use. In this way, `AI::Categorizer` benefits when progress is made in Weka, as well as the other way around.

We hope to create interfaces to other existing products as well. If the `AI::Categorizer` project gains enough momentum that other people wish to contribute code to it, we will encourage this code to be as independent and generic as possible so that we may simply create an interface to it in our framework. For instance, this is how the SVM learner in our framework was created recently–our `AI::Categorizer::SVM` class is just a thin wrapper around a generic `Algorithm::SVM` module by another author we collaborated with, and this in turn is a wrapper around the C library `libsvm`.

It is our hope that this strategy will extend the reach of both our framework and related existing and new TC software.

In designing the `AI::Categorizer` framework architecture, we have focused on aspects of Text Categorization that tend to remain common from one task to the next, allowing for growth in aspects that tend to change. For instance, we have specified that document features are encapsulated in a `FeatureVector` object, but we have not specified that object's internal implementation. Likewise, we have specified that the Machine Learning TC algorithms are encapsulated by the `Learner` class, but the specific algorithms will tend to vary from task to task.

In the first public versions of the framework, we have tended to implement the simplest versions of each of these classes, with more elaborate or optimized implementations deferred to later work. For instance, our `FeatureVector` class is currently implemented using Perl hashes, but other implementations (for instance, using C structs to implement sparse integer vectors) may be implemented in order to improve memory usage and/or speed. Other `Learner` subclasses may also be added, and the existing subclasses may be improved to provide more feature-rich implementations or improve efficiency. Because they are encapsulated in subclasses, these implementations may be traded at will, allowing experimentation with different implementations. In particular, we expect the `Learner` and `FeatureSelector` areas of the framework to grow as new algorithms are added and existing algorithms are informed by current research.

7 Conclusions

We have developed a new framework for Text Categorization which is publicly available and leverages existing work as much as possible. We have primary goals of providing usable TC software for application developers, researchers, and domain experts, as well as providing bridges between existing and new TC software. Our framework design endeavors to embody the key requirements which are common to most work in TC, and thus should improve reusability of design and implementation in applications that use text categorization. The analysis of its architecture may be useful to those embarked in building their own TC systems, so we have discussed the design decisions of the different functionalities supported by the framework.

Periodic point-releases of `AI::Categorizer` are available at http://www.cpan.org/modules/by-authors/id/KWILLIAMS/ , and bleeding-edge development versions are available via CVS at http://www.sourceforge.net/projects/ai-categorizer/ .

References

[1] Rafael A. Calvo. Classifying financial news with neural networks. In *6th Australasian Document Symposium*, page 6, December 2001.

[2] Damian Conway. *Object Oriented Perl*. Manning Publications Company, August 1999.

[3] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, Volume 20, Number 3, pages 273–297, 1995.

[4] Mohamed Fayad and Douglas C. Schmidt (editors). *Building Application Frameworks*. John Wiley & Sons, 1999.

[5] Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides. *Design Patterns*. Addison-Wesley, 1995.

[6] David D. Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. In Claire Nédellec and Céline Rouveirol (editors), *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 4–15, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

[7] Andrew Kachites McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/ mccallum/bow, 1996.

[8] M.F. Porter. An algorithm for suffix stripping. *Program*, Volume 14, Number 3, pages 130–137, 1980.

[9] J. R. Quinlan and R. L. Rivest. Inferring decision trees using the minimum description length principle. *Information and Computation*, Volume 80, Number 3, pages 227–248, 1989.

[10] Gerard Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, Reading, Pennsylvania, 1989.

[11] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management: an International Journal*, Volume 24, Number 5, pages 513–523, 1988.

[12] Bernhard Schölkopf, Christopher J.C. Burges and Alexander J. Smola (editors). *Advances in Kernel Methods – Support Vector Learning*. MIT Press, 1999.

[13] Larry Wall, Tom Christiansen and Jon Orwant. *Programming Perl*. O'Reilly and Associates, 3 edition, 2000.

[14] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, Chapter 8, pages 265–320. Morgan Kaufmann Publishers, 1 edition, 1999.

[15] Y. Yang and X. Liu. A re-examination of text categorization methods. In *22nd Annual International SIGIR*, pages 42–49, Berkley, August 1999.

[16] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In Douglas H. Fisher (editor), *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 412–420, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.

Improved use of Contextual Information in Cross-language Information Retrieval

Ying Zhang, Phil Vines

School of Computer Science and Information Technology, RMIT University
GPO Box 2476V, Melbourne Victoria 3001, Australia

{ *yzhang, phil* }*@cs.rmit.edu.au*

Abstract

In this paper, we explore Dictionary based context-sensitive translation, a framework for query translation to reduce the translation ambiguity and improve the translation quality in English to Chinese cross-language information retrieval (CLIR). Our paper explores the effect of the context window size on translation effectiveness. We assume that the correct translations of the query key terms tend to co-occur together at a high frequency and incorrect translations do not. Our experimental results showed that when using a window size of 10, context-sensitive translation results in a dramatic improvement in retrieval performance, it brings about a 30% improvement compared to the results of previous Dictionary based approaches that used only immediately adjacent words for context.

Keywords Information Retrieval

1 Introduction

With the enormous increase in recent years in the number of multilingual text databases available online, there has been a growing interest in the research done in the area of Cross-Language Information Retrieval (CLIR). This paper is concerned with one type of CLIR, namely, issuing a query in English and retrieving a Chinese document.

In order to match a document and a query in two different languages, either the document or the query should be translated. By far the simplest approach is to convert queries to the document language and do the monolingual ranked retrieval. This approach is called query translation. Most research has concentrated on query translation [1, 4, 5, 6], as it is computationally less expensive than document translation, which requires a lot memory and processing capacity. Within the query translation framework, basic approaches to CLIR are: the Machine translation based approach, the Parallel corpus based approach and the Dictionary based approach. Regardless of the cross-language approach taken, the main hurdle to improved CLIR effectiveness is resolving ambiguity associated with translation. In our work, we aim to translate each English key term using both a dictionary lookup technique and the available context, into the most appropriate corresponding Chinese term. Rather than use only immediately adjacent words to determine the context, we experiment with a window size *w* of words around the key term.

The rest of the paper is organised as follows: section two presents the related work; in section three, we discuss the formula and the selection algorithm used in our context-sensitive translation approach; section four describes the experiment setup; section five presents and summarizes the evaluation results and provides our discussions, and section six concludes the paper.

2 Related work

Translation ambiguity is a major problem in CLIR and arises from the fact that many words have multiple possible translations. Some researchers in the past have used approaches such as First-Match and Every-Match [7]. More recent work has concentrated on adjacent words to provide the context and thus help select the appropriate translation. Ballesteros and Croft [1] describe a technique that employs co-occurrence statistics obtained from the corpus being searched to disambiguate dictionary translation. They measure the importance of co-occurrence of the elements in a set by using the *em* metric - the percentage of the occurrences of term *a* and term *b* which are net co-occurrences (co-occurrences minus expected co-occurrences).

Our work applies this technique to English to Chinese CLIR and extends this approach by using a larger window of contextual information.

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

3 Context-sensitive translation

Given a set of *n* original query terms $\{e_1, e_2, ..., e_n\}$, we obtain a set C_i of all possible translations for each e_i through a bilingual dictionary lookup and then try to select the best translation that co-occurs with all other sets of translations in the same context window at the highest frequency, from each C_i.

3.1 Mutual information

Mutual information *(MI(x,y))* [2] is used to measure the correlation between all word pairs in the same context window in our application of query translation. In other words, the correct translation of the given query key term is not only determined by the immediately adjacent words but also the words that are in the same context window. Previous work has tended to look only at these immediately adjacent words. This is the major point of difference between our work and previous work [1] and has contributed significantly to our improved results.

We extend the *MI* formula, which previously used to measure mutual information between immediately adjacent words, to measure the *MI* between a given word and every other word within a window size *w*.

$$MI(x,y) = \log_2 \frac{Nf_w(x,y)}{f(x)f(y)}$$

Where $f_w(x,y)$ is the frequency that term *x* and term *y* co-occur within a window size *w*, $f(x)$ is the collection frequency of *x*, $f(y)$ is the collection frequency of *y*, and *N* is the total number of words in the document collection. A window size *w* is the context of the given term that consists of *w/2* words before and after the given term.

3.2 Selection algorithm

There are *n* terms in the English query, for each original English query term e_i $(i \in (1,n))$, we obtain a set C_i of all possible Chinese translations c_{ij} $(j \in (1,m))$ through a bilingual dictionary lookup:

For each set C_i, *do*

  For each translation c_{ij} *in* C_i, *do*

    For each set C_k $(k \in (1,n), k \neq i)$, *do*

      Compute the mutual information

      $MI(c_{ij}, C_k) = Max_{c_y \in C_k} MI(c_{ij}, c_y)$;

    Calculate the score of the translation c_{ij}:

$$S_{c_{ij}} = \sum_{k=1,k\neq i}^{n} MI(c_{ij}, C_k)$$

  Select the c_{ij} *with the highest* $S_{c_{ij}}$

Where *n* is the number of key terms e_i in the English query, *m* is the number of Chinese translations c_{ij} in the Chinese translation set C_i. For more detailed descriptions, see [8].

4 Experiments

We conducted the English to Chinese CLIR experiments in the Chinese collection of TREC 5 and TREC 6 [3]. There are 54 queries and 164,789 documents (170MB) of articles drawn from the People's Daily newspaper and the Xinhua newswire. In our experiments, the first 28 topics (CH01 to CH28) were processed as queries to retrieve the documents from the Chinese collection. The queries use all sections of the topic. In our experiments, we use a dictionary from the Linguist Data Consortium (LDC) - ldc2ec, which has 110,834 entries (http://www.morph.ldc.edu/ Projects/ Chinese). The Chinese stop list was manually selected from the statistical results we obtained from the Chinese document collection. The top 118 words with the highest collection frequency (CF) and relatively unimportant meanings are selected as stop-word and put into the Chinese stop list.

4.1 Monolingual IR experiment

We have carried out monolingual IR experiments using both character-based and word-based approaches. In word-based approach, we use the dictionary based method with greedy parsing, where a dictionary that contains 58,667 entries is employed. These experiments have shown that word-based approach is statistically better than character-based approach. The retrieval performance of the word-based monolingual run is in the second column in Table 1. This provides a benchmark for our CLIR results.

4.2 CLIR experiment

We explored two methods for disambiguating dictionary based query translation, the First-Match method and the context-sensitive translation.

To provide a baseline for CLIR results, we obtained a recall-precision average for the First-Match method. Only the first match translation per query term is retained instead of using all of the listed translations in lcd2ec when there is more than one translation for that term. When there is no exact matching for a single-word term, the term is skipped. The retrieval performance of the First-Match run is in the third column in Table 1.

Our context-sensitive translation scheme works in three stages: pre-processing, dictionary based query translation and translation disambiguation. At the first stage, English stop words are removed from

English topics. When an English word has a trailing "s", it is removed according to Porter's algorithm. Other steps of stemming are not done in order to avoid changes in meaning. The Chinese document collection is segmented into the words; and then Chinese stop words are removed from the Chinese document collection. The second stage does actual query translation based on ldc2ec dictionary look-up. For each Chinese translation, we applied a Chinese stop list filter. At the third stage, the $MI(x, y)$ statistics was used to determine whether the Chinese translations from different sets generated by the translation process are "compatible".

5 Results and Discussion

Using the TREC data and queries described earlier, we have gathered in Table 1 a comparison of the recall precision values for the experimental results. Columns two through seven correspond respectively to the word-based monolingual information retrieval, the First-Match method, and our context-sensitive translation method using a window size of 2, 8, 10 and 12. It is seen that when using a window size of 10, our context-sensitive translation disambiguation method successfully brings effectiveness to over 62% of monolingual retrieval. This is a considerable improvement over previous work (using adjacent information, i.e. $w = 2$) which yielded 33.5% of monolingual retrieval effectiveness. More detailed results and discussion are presented in [8].

Figure 1 plots the precision-recall curves for a comparison of the retrieval performance of the six runs. It is seen that the context-sensitive translation approach significantly outperforms the First-Match method, however, the gap between monolingual and cross-language retrieval is still large.

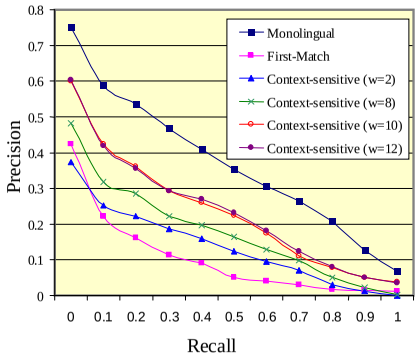


Figure 1: Recall-precision average of queries 01-28

Figure 2 shows the effect of the context window size on translation effectiveness. As the window size is enlarged from 2 to 10, the average precision increases from 0.1267 to 0.2354. There is no significant improvement for window size greater than 10.

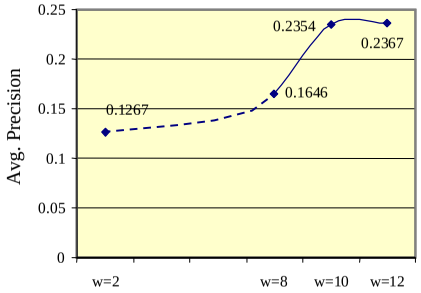


Figure 2: Comparison of the widow size effect on CLIR effectiveness (Avg. precision).

5.1 Factors affecting translation effectiveness

In this section, we examine several factors that may directly or indirectly have degraded the retrieval effectiveness of English to Chinese CLIR.

First, the translation of certain English query key terms cannot be found in the English-Chinese dictionary (ldc2ec) we used in the experiment, such as place names, person names and some terminologies. For example, Xinjiang (新疆，place name) and WTO (世界贸易组织，terminology). Out-of-vocabulary words not found are left as un-translated. This is one reason that our retrieval effectiveness is degraded.

Second, for some English query key terms, there is no exact equivalent Chinese translation, although some approximate equivalents are provided in the ldc2ec. For example, we obtained Chinese translations "/ 数 据 / 数 位 /" for English term "digital", however, the corresponding Chinese translation should be "数字". Therefore, we failed to select the best translation in this case.

Third, since ldc2ec does not support phrase translations, we obtained some inappropriate translations for phrases. For example, we translated peace talks (和平会议) into "和平会谈".

Fourth, some best Chinese translations are missing in the segmentation dictionary. Even when the correct Chinese translation is obtained through the lcd2ec lookup, it is not treated as a word in the segmentation dictionary and the worse thing is that the translation is missing in the segmentation dictionary. Therefore we cannot select it as the best translation. For example, the Chinese translation "海洛因" of English term "Heroin" is missing, so we can only select "白面儿" as the translation of the English term "heroin".

Recall	Monolingual	First-Match	Context-sensitive Translation			
			($w=2$)	($w=8$)	($w=10$)	($w=12$)
0.00	0.7489	0.4245	0.3745	0.4815	0.5996	0.6023
0.10	0.5884	0.2224	0.2518	0.3173	0.4251	0.4188
0.20	0.5355	0.1609	0.2221	0.2858	0.3607	0.3569
0.30	0.4672	0.1139	0.1876	0.221	0.2919	0.2939
0.40	0.4096	0.0906	0.1586	0.1975	0.2588	0.2697
0.50	0.3521	0.0504	0.1240	0.1647	0.2235	0.2325
0.60	0.3055	0.0414	0.0967	0.1277	0.1742	0. 1829
0.70	0.2645	0.0303	0.0698	0.0972	0.1120	0.1240
0.80	0.2063	0.0178	0.0297	0.0502	0.0782	0.0812
0.90	0.1266	0.0157	0.0121	0.0232	0.0513	0.0513
1.00	0.0675	0.0116	0.0008	0.003	0.0368	0.0349
Avg. precision	0.379	0.089	0.1267	0.1646	0.2354	0.2367
% Monolingual	100	23.51	33.47	43.48	62.18	62.52

Table 1: Evaluation Results

6 Conclusion

As explained above, there are several factors that degrade the English to Chinese CLIR effectiveness. In this paper, we have looked in detail at only one of these problems, which is the translation ambiguity problem, and shown how significant improvement may be obtained. We adopted an improved context-sensitive translation method to improve dictionary based query translation in English to Chinese CLIR. We translate each English key term using both a dictionary lookup technique and the available context, into the most appropriate corresponding Chinese term. Through our experiments, we showed that our approach significantly outperforms the previous work that used only immediately adjacent words for context, i.e. a window size of 2, and leads to a relatively high effectiveness.

References

[1] Ballesteros, L & Croft, W.B. Resolving Ambiguity for Cross-Language Retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and* Development in Information Retrieval, August 24-28 1998, Melbourne, Australia. ACM 1998. pp. 64-71.

[2] Church, K. W. and Hanks, P. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 1990, 16(1), pp. 22-29.

[3] E. Voorhees and D. Harman. Overview of the Sixth Text REtrieval Conference. In *E.M. Voorhees and D.K. Harman, editors, Proceedings of the Sixth Text REtrieval Conference* (TREC-6), pages 1 -- 24, Nov 1997.

[4] Jianfeng Gao, Jian-Yun Nie, Endong Xun, Jian Zhang, Ming Zhou, Changning Huang. Improving Query Translation for Cross-Language Information Retrieval using Statistical Models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, September 9-13, 2001, New Orleans, Louisiana, USA. ACM 2001.

[5] Hull, D. A., and Grefenstette, G. Querying Across Languages: A dictionary-based approach to Multilingual Information Retrieval. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 49-57). Zurich, Switzerland: ACM Press.

[6] Jang, Myung-Gil Jang, Sung Hyon Myaeng and Se Young Park. Using Mutual Information to Resolve Query Translation Ambiguities and Query Term Weighting. *In ACL-99*. College Park, Maryland, pp. 223--229.

[7] Mohammed Aljlayl and Ophir Frieder. Effective Arabic-English cross-language information retrieval via machine-readable dictionaries and machine translation. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, September 9-13, 2001, New Orleans, LA USA. ACM 2001. pp. 295 - 302 J.

[8] Zhang, Ying. Multilingual Querying for Information Retrieval. A Minor thesis submitted in partial fulfillment of the requirements for the degree of Masters of Applied Science, School of Computer Science and Information Technology, RMIT University, pp. 16-17.