Towards Concept-based Structuring of Electronic Scientific Information

Jon Ragetli
Jon.Ragetli@12move.nl

June 17, 2001

Masters Thesis for the Graduate Programme in Logic
at the Institute for Logic, Language and Computation
of the University of Amsterdam

This research is supported by Elsevier Science Publishers.

Preface

With mixed feelings I have been writing this thesis. Would I have chosen differently, this thesis would not exist and I would be busy trying to answer some of the questions raised in this thesis. However, I have decided not to continue the solitary work of getting a PhD. Therefore, the document that you are reading discusses the initial work performed by me as a PhD student in the Logic and Language Links project.

Despite my decision to stop, I have to say that working on the Logic and Language Links project was very instructive. Many people have made this possible, some of whom I would like to thank here. First of all my supervisors: Maarten de Rijke and Jan van Eijck. They gave me much support and guidance.

Joost Kircz, from KRA Publishing Research Amsterdam, deserves many thanks for introducing me to relevant research within the 'Communication in Physics' project, and discussions about the organization of information.

Christof Monz kindly allowed me to use his RETRIEVE system for doing the experiments with link generation, and gave me useful hints into the Information Retrieval literature.

I would also like to thank the other members of the L3 steering group — Arjen Sevenster, Pieter van Wiechen, Jonathan Clark — for the useful feedback and pleasant discussions during the quarterly project meetings.

Finally, Elsevier Science B.V. deserves many thanks for the funding; without the latter, the L3 project would not even exist in the first place.

I would like to wish all the L3 people, especially my successor, the best with the continuation of the project!

Jon Ragetli

Abstract

This thesis discusses the structuring of electronic information by means of a concept hierarchy. The information to be structured is scientific information of a unifying kind, as traditionally found in paper handbooks. The purpose of the concept hierarchy is to serve as an easy to use means of navigation through the electronic information.

We first describe the structure we impose on such a concept hierarchy, the concept hierarchy we built so far, and the text material used for this purpose.

Since the construction and maintenance of a concept hierarchy is very labor intensive, we would like to automate it as much as possible. Therefore, we discuss methods found in literature to automatically generate concept hierarchies on the basis of a collection of texts. We also address the automatic generation of links from the concept hierarchy to the scientific information. We discuss several approaches from literature, and our own experiments using information retrieval techniques. From those experiments, we conclude that it is not possible to fully automate the linking, but that it is possible to construct tools to support the editor of the hierarchy with this task.

Finally, we summarize the ideas presented in this thesis, and describe the next steps and the expected output of the L3 project.

Contents

Chapter 1

Introduction

This report describes initial research performed during the first year of the Logic and Language Links (L3) project. The L3 project is carried out at the Institute for Logic, Language and Computation at the University of Amsterdam, and supported by Elsevier Science Publishers.

The project plan [vE98, page 2] specifies the subject of the research in the L3 project in the following manner:

> Subject of investigation in this project is a general architecture for environments for electronic dissemination of scientific knowledge of a unifying kind, as traditionally found in a handbook, on the basis of an exemplary architecture for an electronic dissemination environment, to be called Logic and Language Links (*LoLaLi*).

The starting point of the research is the Handbook of Logic & Language[1] edited by Van Benthem and Ter Meulen [vBtM97]. An exemplar implementation of the ideas developed on structuring information for electronic dissemination should also be based on the Handbook, thus providing an electronic version of parts of the Handbook.

Electronic publishing offers a lot of opportunities. It makes distribution easier and quicker. Also, readers can be helped considerably when searching for information; already simple keyword searches are much more useful than scanning tables of contents or indexes. Tracing a reference can become as simple as a mouse click if the right hyperlinks are added to the publication.

Furthermore, electronic publications are less rigid than their paper counterparts: the publication can be adaptable to the reader, thus better satisfying the reader's personal information needs. It also offers a wealth of possibilities for the reuse and multiple use of written material. For example, when a contributor to a handbook wants to add a new version of his chapter to the handbook, he might reuse the existing introduction of that chapter. An example of multiple use is that one fragment of a handbook might be used for several purposes, when several reading paths visit the part. Electronic publishing also facilitates the development of tailor-made products and prices — instead of selling an entire journal or magazine, a publisher can sell the articles individually [Can00].

[1] In the following, we will sometimes refer to the Handbook of Logic & Language as 'the Handbook'; with 'handbook', we refer to handbooks in general.

Having a publication available electronically means that it can be integrated with other media types: e.g. computer simulations, movies, and tools. Those can be added to the publication itself (e.g. an author writes both an article and a simulation) or be available on the WWW (in this case an author writes the article and the author, or an editor, adds useful hyperlinks to the other media types).

Also, electronic publishing can make contact between the author and his readers more direct. For example, in the online *Journal of Interactive Media in Education* (JIME)[2] next to every section of an article, there is a so-called 'discussion space', in which readers can comment on and discuss with each other and the author about the contents of the respective section [SBS98]. Such features allow direct and focused interaction.

While electronic publishing offers a wealth of possibilities, it also has some drawbacks from the publisher's point of view. In particular, for scientific publishers, finding novel ways of publishing is an urgent problem, since new electronic possibilities create a threat of disintermediation (authors do not need a publisher to reach their readers), and the problem that copyright protection is harder to maintain. An example from the non-scientific literature market is the recent publication of Stephen King's novel 'The Plant', which is highly successful. King is publishing it in several parts at his own Web site,[3] and *asks* a payment per downloaded part, without *forcing* readers to pay. Still, 'sales' are high: 88% of the people who download a chapter do indeed pay. This way, King proves that publishing can well be done without a publishing company. It should be noted here, however, that King is a famous writer already, and the same success might not be achieved when his name were not known to such a large audience.

The threat of disintermediation forces publishers to be inventive enough to still attract authors to publish with them. Or, as a recent study of the CPB Netherlands Bureau for Economic Policy Analysis states it [Can00, Conclusion, page 5]:

> *Authors* will rely less on publishers to distribute and advertise their work. This may provide an incentive for publishers to prove to authors where their added value lies, ...

Research in the L3 project is a way to be inventive in this sense.

This thesis is organized as follows. As may be clear from the excerpt of the project description above, the aim of the project is described in quite general terms and can be satisfied in many different ways. Therefore, after the start of the project, we have refined this goal into a workable problem definition by analyzing the requirements of four groups of users of an electronic handbook: readers, authors, editors and the publisher. On the basis of those requirements the themes of easy navigation through and access to handbooks became evident. With those themes in mind, it was decided to investigate the use of a *concept hierarchy* for navigation and access to a scientific handbook. The requirements, and the reasons to use a concept hierarchy, are described in Chapter 2.

Then, in Chapter 3, we discuss the structure of the concept hierarchy we use for navigation and access, and ways of visualizing such a hierarchy. Of

[2] `http://www-jime.open.ac.uk/`, visited on January 5, 2001.
[3] `http://www.stephenking.com/`, visited on January 5, 2001.

course, the concept hierarchy has to be constructed and maintained. Those issues are described in Chapter 4. As construction and maintenance can be a quite labor-intensive task, we discuss tools to support authors with those tasks.

Chapter 5 describes ways of automatically linking the concept hierarchy to the text of the handbook, and experiments with the use of such algorithms. Automating the construction of links is very important, as it would be too much work for a so-called *link-editor* to create the links manually.

In the final chapter we conclude and give directions for further research. Appendix A contains a glossary explaining some of the more technical terms.

Chapter 2

Navigation and Access

We start this chapter by discussing the requirements of the users of an electronic handbook. In Section 2.2 we narrow down our subject of research and describe two themes arising from the requirements that we consider most important. In Section 2.3 we will describe how we will address both themes.

2.1 User Groups

We distinguish four user groups of an electronic handbook: readers, authors, editors and the publisher. Discussing the requirements of the user groups, we mainly focus on the first two groups — not taking into account readers' requirements there is no need for an electronic handbook, ignoring the authors' requirements there will be no content for a handbook. Editors and publishers form the bridge between the authors and the readers.

The user groups need not be disjoint; for example, the author of one part of a handbook may be the reader of another part.

2.1.1 Readers

An important function of scientific handbooks is to provide the reader with an overview of the state-of-the-art within a certain scientific discipline. This implies two things: an electronic handbook should contain up to date information, and it should refer to relevant material about the subject.

With respect to providing an overview of the *state-of-the-art* within a scientific discipline, electronic publication of handbooks certainly has the advantage that the handbooks can be kept up to date much more easily than paper publications, as the paper publications would require a reprint even if only minor parts of it need to be updated.

Good references in an electronic environment can be much more useful than references in a paper handbook: tracing references can be performed via simple clicking on hyperlinks, instead of going to the library looking for the publication. The growth of the number of publications available electronically — on researchers' homepages, in online (preprint) repositories (e.g. the Beta Preprint Server[1] of the University of Amsterdam) and on publishers' sites (e.g. Elsevier's

[1] `http://preprint.beta.uva.nl/`, visited on January 5, 2001.

ScienceDirect[2]) makes this possible. Those references can be kept up to date with recent developments in the field. Furthermore, other types of resources than merely scientific papers can be referred to: homepages of researchers, research institutes, course material, software and the like. As it is very annoying for a reader to encounter broken links, the links should be well maintained.

Paper books impose, by their very nature, a linear reading order. If not for the entire book (in the case of novels), then at least within smaller parts of the book — in the Handbook, the chapters can be read more or less in any order, but the chapters themselves should be read linearly. However, electronic books can easily facilitate a more modular way of reading. Many sites on the WWW consist of many relatively small modules connected by hyperlinks. Usually, the hyperlinks do not force a sequential or linear reading order from module to module, but allow several different reading orders, making the multiple use of modules possible. Like the linear reading order of books, this property of Web sites has arisen from practice, forced by, amongst others, constraints on screen size and download times. The 'small modules – many links' structure seems to be the most suitable way of information representation on the Web, so an electronic handbook should be structured in this fashion.

In [Har00, KH00], the authors propose a modular structure for articles in experimental science. Articles in experimental science seem to be good candidates to present modularly, but more research is needed to determine whether this approach can be adapted to handbooks in general, and in particular for a handbook like the Handbook of Logic and Language that contains more abstract content. The writing styles make up another difference; the Handbook of Logic and Language has a more essay-like character, as opposed to the problem solving oriented writing style in the experimental science literature.

A potential problem with the 'small modules – many links' structuring of information is disorientation of the reader, or as Conklin [Con87] calls it: "getting lost in space". A reader should know where he is in the hypertext network, and also know how to get to another place in the network. Therefore, there should be good navigation tools that clearly show how the reader arrived at the current module and where he or she[3] can go to next. Such tools can range from a browsable table of contents to more sophisticated tools, as we will discuss in Chapter 3.

In an electronic environment, sophisticated techniques for indexing and searching can be utilized. Different search and indexing tools are needed by readers of different types of electronic publications, since readers access them with different goals in mind. In our case, a handbook should be easily accessible to both novice and more expert readers. For example, we should take into account that novice readers do not know exactly what they are looking for and should be assisted first in discovering what information they need and then finding it. Therefore, we do not agree completely with Conklin [Con87] that search tools solve the disorientation problem; for novice readers they do not, since search tools require a precisely worded query.

An interesting feature of electronic publishing is that publications can be adaptable in the sense that readers are shown different content and links depending on reader characteristics such as interests and maturity in the scientific

[2]http://www.sciencedirect.com/, visited on January 5, 2001.

[3]For ease of writing, in the following we will use the word *he* to refer to persons that can be male or female.

discipline. For example, there can be different reading paths through the modules (cf. Harmsze's [Har00] essay path and complete path). The modules visited by different paths can, of course, (partially) be the same, making multiple use of written material possible. The degree to which the adaptability is automated can range from readers stating explicitly their preferences to the handbook (like a more extensive query to a search engine) and being shown content according to this statement, to an active handbook that tries to derive the user characterics from reader behavior. Too much adaptability of the handbook should be avoided, since this will disorientate readers.

Adaptability can also partly solve the problem of *cognitive overhead* [Con87] that occurs when reading hypertext: the reader has to choose constantly which links to follow or not. In linear text there are not many such choices since the author has already decided for the readers. If the handbook is adaptable in the sense that users can decide which (types of) links are presented to them, the cognitive load can be reduced considerably.

An electronic environment allows for other means of communication than just the written word. This can range from color illustrations to animations, or even fully interactive computer programs, making the content of the handbook more understandable and lively for the reader. For the Handbook of Logic and Language, one can think of animations explaining the semantics of dynamic logics, or theorem provers enabling the reader to examine the truths of a certain logic. Material that one might use for this purpose is for example [Jas99] or output of the ongoing Spinoza project 'Dissemination of Logic'.[4]

Notwithstanding the nice features that an electronic environment offers, we should take into account the desire of most readers to read things from paper, so it should be possible to print (selected parts of) the handbook. Paper still has many advantages over a computer screen: it is more comfortable for the eyes, it is portable, people like browsing through physical pages, etc.

2.1.2 Authors

With respect to the authors, the people creating the content of a handbook, we assume that they prefer to have as little work as possible writing the chapters of the book. This means that publishing in an electronic handbook should not take substantially more effort than publishing in a traditional one. Whenever possible, we should provide authors with tools to support this. We should not assume any technical knowledge on the side of the authors except basic word processing capabilities — the same capabilities they would need to publish in a traditional fashion. As a consequence, the interface of an electronic handbook environment with its authors should hide all technical details.

Like for readers, for authors an advantage of an electronic handbook over a paper handbook is that their contributions can be published shortly after they have finished writing it. There will be no delays caused by printing, distribution and waiting for other authors to complete their contributions; almost immediately after the review has ended, their contribution can be published.

We would like to stress that the speed-up of publication must not be achieved by omitting the review of handbook chapters. Authors will prefer waiting to have a thoroughly reviewed publication; otherwise the recognition by the scien-

[4] `http://www.cwi.nl/~pauly/spinoza.html`, visited on January 5, 2001.

tific community would be less and they might as well have published it on their personal homepage. Nevertheless, with respect to this review, new opportunities arise. For example, in the mentioned online 'Journal of Interactive Media in Education' (JIME)[5] each section of an article, has a so-called 'discussion space', in which readers can comment on the contents of the respective section and discuss with each other and the author about it [SBS98].

In the same way as readers understand a publication more easily when other media are used, the authors can express their ideas better with the help of other media, like animations, movies or computer programs. Another parallel to be drawn with the readers is that for authors it is also necessary that they do not get lost in the hyperspace they create themselves. Authors also need navigation and visualization tools. Furthermore, it would be very useful for an author if he could verify for example automatically whether a certain module he has written will have been read before the reader arrives at another module. Some logic-based proposals for such verification tools can be found in literature: Réty [Rét99] proposes a predicate logical approach; Stotts et al. [SFC98] an approach based on model-checking for modal logic.

A potential problem for authors is the demand to write their contributions in the 'small modules – many links' fashion as is common in electronic publications. This might demand too big an effort on their part, making them reluctant to contribute. Definitely, this problem should be avoided to every possible extent. Clear examples of handbook contributions, templates and checklists should avoid this problem by assisting authors in adapting their writing style. It also imposes a heavy burden on the editors of the handbook, as they should guide authors in this respect.

Issues of intellectual property are important for authors: the authenticity of their work should be safeguarded in an electronic environment. In this report, we will not deal with those issues — interesting and complicated though they may be.

2.1.3 Editors

For scientific editors, it would be nice to have some reviewing environment only accessible to them and the reviewers, that makes it possible to have a prepublication of new handbook contributions. With a discussion space like in the mentioned JIME, reviewing can become a more lively and interactive process. For editors it may be even more important than for the readers and authors to have a clear overview of the contents of the handbook that is being written, in order to be able to supervise its development.

As mentioned above, editors of an electronic handbook have a more important duty than editors of traditional handbooks, as besides their usual tasks they also have to safeguard the writing style of the author. For this reason, it will be necessary to have editors that are familiar with ideas about modular writing style.

2.1.4 Publisher

The publisher of the electronic handbook will be responsible for its maintenance. Tools are needed to perform this task, as the dynamic nature of an electronic

[5] `http://www-jime.open.ac.uk/`, visited on January 5, 2001

handbook complicates the maintenance. For example, when a new version of (a part of) a chapter is added, several potential problems should be taken into account. First of all, the old version should be retained, as there might be hyperlinks pointing to it. However, some, but not necessarily all hyperlinks should be updated to point to the new version. It must also be avoided that some parts of the handbook become unconnected due to links that were present in the old version and do not exist in the new one. Furthermore, one must keep track of who is the author of the new version, as it might be a different person than the original author.

When an entirely new chapter is added to the handbook, it must be well embedded within the rest of the handbook. The table of contents, hyperlinks and the index should be updated and kept consistent. Those 'link management' tasks are new compared to publication of paper material, and the publishing company will need technical editors that can perform those tasks.

Furthermore, tools are needed that support the maintenance of the links, or that automate it entirely. For this, systems generating and updating hyperlinks automatically, or *version management* systems, that are used to keep consistency in large software engineering projects, can be very useful.

A consequence of those new tasks is that the publishers should have technical editors at their disposal that can perform the tasks, which differ highly from the tasks a technical editor traditionally performs.

2.2 Narrowing down the Problem

In this section, we will narrow down the problems arising from the requirements identified in Section 2.1 to two themes that we want to study: access and navigation. The requirements discussed in the previous section can be summarized as follows:

1. The electronic handbook should be well embedded into the Web, giving easy access to all kinds of relevant information for people with different levels of knowledge of the field.

2. It should have a modular structure, as opposed to a linear one.

3. Disorientation should be avoided; a clear overview of the contents should be available to every user, in order to enable him to navigate through the handbook. In line with this, there have to be searching and indexing facilities suited to the type of contents of an electronic handbook.

4. Writing for the handbook should not require any technical knowledge apart from word processing capabilities.

5. For authors, clear guidelines should be available on the writing style for an electronic medium.

6. The review of contributions to the handbook should be facilitated by electronic means.

7. The electronic handbook should be kept up to date in order to reflect state-of-the art developments in its field.

8. Tools should be available to facilitate the maintenance of the handbook.

9. It should be easy to incorporate other media than just the written word in it.

10. Although it is an *electronic* handbook, its contents must be printable with quality comparable to paper publications. This does not hold, of course, for the non-textual content like software programs and movies.

11. It should be adaptable to the reader.

12. Intellectual property issues should be taken care of.

This is an extensive list consisting of many challenging issues. To remain focused, we have to restrict ourselves. In this thesis, we zoom in on the first three requirements, but in the process of doing so we will also address requirements 4 – 10. Two main issues can be distilled from the first three requirements: easy *navigation* within the handbook and and easy *access* to its contents and relevant material on the WWW. They will be clarified below.

With respect to the eleventh requirement, this comprises the scientific discipline of Adaptive Hypermedia, and is too extensive to address within this project in addition to the subjects of navigation and access. The issue of intellectual property (the twelfth requirement) is outside the scope of this project as well since it requires extensive legal knowledge.

Navigation. Having good navigation tools at one's disposal can solve the problem of disorientation, caused by the 'small modules – many links' writing style for the Web. Navigation tools for an electronic handbook should satisfy the following demands:

1. A user[6] should be able to see where he is in the web of hyperlinked modules. More specifically, he should be able to

   - See what are the outgoing links of the module he is currently reading. Assigning types to the links (see Section 3.2.2), and indicating their destination can clarify this.
   - See how he arrived at the current module.
   - Backtrack to previously read modules.

2. However, if there are lot of links, seeing them all might distract the user rather than help him to orientate. Therefore, he should always be able to choose which of them are visible at one moment. The mentioned typing of links makes this possible.

3. The navigation tools should be intuitively understandable for users with a basic level of experience in using the Web or other electronic publications. If not, it would require too much effort from the user to learn how to use the navigation tools, possibly making him refrain from using them and thus causing him to still get lost.

[6]We employ the term *user* here to refer to both authors and readers of the electronic handbook.

Access. It should be easy for users with a certain information need to access the handbook and find the relevant information. This requirement can be fulfilled in several ways: a good index, advanced information retrieval tools, a browsable table of contents. We find it important that the contents of the handbook are accessible both to users with a very clear information need and to users with a vaguer one, which might be due to ignorance about the exact contents of the handbook or little knowledge about the scientific discipline the handbook is about. For readers with such a vague information need, the three solutions discussed above under 'navigation' do not suffice, and more sophisticated guidance is needed. We will discuss our specific proposal in the next section.

2.3 Addressing the Task

To address the navigation and access themes in Section 2.2, we propose to build a *concept hierarchy* that gives access to the contents of the handbook. We use the term 'concept hierarchy' to refer to a topic hierarchy, where the topics may also contain a description of the meaning of the topic. Thus, a concept hierarchy provides more information than a topic hierarchy. Nevertheless, topic hierarchies are already very useful.

Topic hierarchies are often used for the purpose of navigating through a large collection of documents. They are very useful for the organization, display and exploration of large amounts of information. One famous example is *Yahoo!*'s topic hierarchy for exploring the Web.[7] Many other portal sites also provide topic hierarchies for facilitating the finding of information on the Web, an indication of the success of using topic hierarchies for navigation through large amounts of information.

An example of the use of topic hierarchies for navigation that dates from before the Web age is WordNet [Fel98], an electronic lexical database developed at Princeton University. In the initial stage of its development the idea was that WordNet could be used for exploring machine-readable dictionaries on the basis of semantic similarities. Since then, WordNet has evolved and has been applied to several other tasks, like word sense identification and information retrieval.

Of course, the organization of information through topic hierarchies is not a novelty of the computer age. For centuries, librarians have classified and catalogued material in the library according to subject classification schemes. Because of this, most people are familiar with using such hierarchies for the purpose of locating information.

Why Concept Hierarchies? There are some obvious advantages of using concept hierarchies over other ways of locating relevant information. One advantage is that users do not have to know exactly what information they are looking for. Without having to word their information need in precise terms, they can browse from general to more specific concepts and thus get a clearer idea of the information they are looking for. This especially helps readers less familiar with the subject matter for which they are consulting the handbook. An advantage of a topic hierarchy on the Web over a search engine is that sites

[7] `http://www.yahoo.com`, visited on January 5, 2001.

are added manually to a topic hierarchy and have been approved by the editor of the topic, so the chance of visiting a useless site is considerably smaller than when using a search engine.

It has also been shown that users in a hypertext search task who had hierarchical browsing patterns through the hypertext performed better than users who had sequential browsing paths [McE99]. Therefore, it is very important that the handbook allows, or even forces such hierarchical patterns, and a concept hierarchy is a good way to do this.

Links from the Hierarchy. Besides a short description, to provide access to the handbook text, concepts in the hierarchy should have links to the part(s) of the handbook discussing the topic. Ideally, concepts higher in the hierarchy should refer to larger parts of the handbook than more specific concepts lower in the hierarchy. There must also be links to other resources on the Web covering the handbook's subject matter. The above two sorts of links depart from the hierarchy to external sources, but other links, internal to the hierarchy, are useful as well. Besides the hierarchical relationship, such links can express other relationships between concepts. For example, a 'similar' relationship can link concepts in the hierarchy similar to other concepts in the hierarchy, which are not connected by the hierarchical relationship. This can enhance the user's understanding of the contents of the handbook and provides navigational shortcuts, which is very beneficial if the hierarchy is deep.

Fulfilling Requirements 4 – 10. By addressing the issues of access and navigation, we do not focus explicitly on the requirements 4 – 8 in the list in Section 2.2. However, building a concept hierarchy for navigation also forces us to work on those requirements as well. For example, inviting authors to help build the hierarchy makes it necessary to have edit facilities and guidelines for introducing a new concept in the hierarchy. In the same way as for an entire electronic handbook, there should be means of reviewing additions to a concept hierarchy, and it should be kept up-to-date as much as possible.

The requirements 9 and 10 (inclusion of other media and printability) only have to do with technical layout details and are not worth too extensive research in the context of the L3 project, but will also be addressed during the construction of the concept hierarchy. With respect to this, it should be mentioned that Adobe's Portable Document Format (PDF) seems very suited for the presentation of handbook material. PDF files can be created from LaTeX source files, and most authors are already familiar with LaTeX. Furthermore, the Acrobat Reader program for viewing PDF files is available for free[8] and can be embedded in Web browsers. It offers facilities for following links, both to other PDF files and Web pages. The latter, as well as the adaptation of the generated PDF files to be suitable for reading on the screen is made possible by LaTeX packages such as `pdfscreen` and `hyperref` [GR99].

To summarize, in this chapter we have first stated the requirements of the four user groups of the handbook. After that, we discussed in Section 2.2 the relative importance we attribute to every requirement. As a result, the themes of access

[8] Acrobat Reader can be downloaded from `http://www.adobe.com/products/acrobat/` (visited on January 5, 2001).

and navigation emerged. Both themes are well addressed by a concept hierarchy to be used for navigation, as we have seen in the current section. In the next chapter we discuss the structure of the concept hierarchy for navigating through the handbook.

Chapter 3

Structuring a Navigation-Oriented Concept Hierarchy

3.1 Introduction

In this chapter we describe the structure of the concept hierarchy for accessing and navigating through the handbook. In Section 3.2 we describe the internal structure of the hierarchy. The main components of such a hierarchy are the concepts, which we will discuss in Section 3.2.1, and several relationships between the concepts, which will be discussed in Section 3.2.2.

Those relationships are *internal* to the concept hierarchy. But in order for the hierarchy to be useful as a means of access to the handbook and other material, it should be properly linked to that material. We call such connections, which also include links to information on the World Wide Web, *external links*; we will discuss them in Section 3.3. Section 3.4 discusses various ways of visualizing the concept hierarchy in order to enable easy navigation. In Section 3.5 we show an example concept in the hierarchy and conclude.

3.2 Internal architecture

3.2.1 Concepts

The building blocks of the concept hierarchy are the concepts, and its cement consists of several semantic relationships between the concepts. In line with WordNet, a widely used electronic lexical database [Fel98], we make a distinction between words or terms on the one hand, and concepts on the other hand: a concept is denoted by a *synset*, a set of synonymous words. Words are synonymous if they have (more or less) the same meaning. Contrary to dictionary entries, a concept can have only one meaning. Therefore, polysemous words (words that have more than one meaning), can occur in several synsets: all the synsets that correspond to one of its meanings. For example, the word 'logic' can occur both in the synset {logic, logical system, system of logic} and {logic, mathematical logic}. So, instead of dictionaries where words are the primary objects, we take a more semantic point of view and use concepts, denoted by a set of words with the same meaning, as the primary objects.

Next we will discuss what we understand by synonymy, then we will discuss three other types of information that may be present with a concept in the hierarchy: a gloss, a description and metadata. We conclude this subsection by the discussion of some requirements on the concepts in the hierarchy.

Synonymy. Synonymy is often defined in terms of the interchangeability of words in a sentence, without changing its meaning significantly. For example, the terms 'regular language' and 'Type 3 language' are synonyms, as they can be used interchangeably in the sentence *Regular languages / Type 3 languages are less expressive than context free languages.* The interchangeability condition can be applied very strictly or more liberally. The most strict definition is that words are synonyms if they are interchangeable in *every* context; following this definition, 'regular language' and 'Type 3 language' are indeed synonyms. However, natural languages do not contain many synonyms if this definition is used. One can relax it by taking synonymy as the interchangeability in *some* contexts. It is the latter definition that we use for the construction of synsets. Besides 'regular language' and 'Type 3 language', which can be interchanged in every context, 'logic' and 'logical system' are now also synonymous since they can be interchanged in *some* contexts, e.g. *We studied the properties of the K5 logic / logical system*, but not in *every* context: e.g. *Logic / Logical system has evolved significantly since the death of Aristotle.* One can define synonymy more formally, like *cognitive synonymy*, which is defined in terms of truth conditions, but for our purposes the less formal definition above suffices.

Other Ingredients. A minimal concept in the hierarchy would only consist of a synset. However, if it is necessary to distinguish between different children of a mother concept, a short *gloss* describing this particular child can be added, like 'the branch of philosophy that analyzes inference' and 'a system of reasoning' to distinguish between two 'logic' concepts. The need for such glosses became evident during the construction of the concept hierarchy.

Furthermore, like a glossary that is often present in textbooks and handbooks, a more extensive *description*, possibly containing examples, counterexamples, pictures and movies that clarify the concept, can be added to the concept in the hierarchy. Such descriptions will make the hierarchy useful for novice handbook readers, while the experts might only need the synset or gloss in order to find the information they need. Whenever appropriate, references to relevant literature can be made here. If the publication is available online, there should be a hyperlink to it, as discussed in Section 3.3.2.

Finally, every concept in the hierarchy should carry bibliographic *metadata*. The metadata with every concept should at least contain information on the author and its affiliations and the date of inclusion in the hierarchy. The metadata should also allow the storage of revision information; an electronic handbook makes continuous revision possible, even by someone who is not the original author. Those issues should well be taken care of in order to avoid reluctance to contribute on the author's side.

Requirements. Although the synsets denoting the concepts consist of a set of terms, like *polythetic concepts*, they are *monothetic*, in the sense that membership of the concept is based on only one feature. Polythetic concepts are concepts that are defined by a set of different terms, each representing an aspect of the concept that should be present in some degree with an instance of the concept. An example of a polythetic concept is the following:

> *export international unit japan trade manufacturer citation german output trd news south* (taken from [SC99]).

Polythetic concepts should not be confused with polysemous terms, or with concepts denoted by synsets: a polysemous term is a term with more than one meaning (so in our hierarchy it would occur in different concepts); the synset denoting a certain concept can consist of several terms, but each term has the same meaning, which is not the case with a polythetic concept. Most manually constructed hierarchies are *monothetic*.

For navigational purposes, we think monothetic concepts are to be preferred over polythetic ones. In monothetic hierarchies, every concept or cluster clearly deals with one topic, a topic chosen by the constructor of the hierarchy. In polythetic clusters, however, it is not always entirely clear what is the topic of this concept, as it is defined by several terms. Furthermore, for sub-concepts, or documents that might or might not belong to this concept, membership conditions are gradual and not clear-cut [SC99]. This is demonstrated by the fact that polythetic clusters are often displayed visually, with the distance between a cluster's center and a sub-cluster indicating the degree of membership. To avoid this vagueness, we prefer monothetic concepts, thereby accepting the drawback that monothetic hierarchies seem to be harder to generate automatically.

We have two other requirements for the hierarchy: *completeness* and *correctness*. With completeness we mean that the concepts in the hierarchy should cover all the important information that the handbook and other connected sources contain. This does not mean it is a restructured copy of the underlying information; it rather is a well structured and more abstract representation of it. If the hierarchy is not complete, certain information is not accessible for the reader. *Correctness* is the demand that the hierarchy should not cover information to which it does not give access. Fulfilling both requirements will cause the hierarchy to be a structured and coarse grained mirror of the information it unlocks.

3.2.2 Concept Relationships

In this section we turn to the cement of the concept hierarchy: the semantic relationships between the concepts. First we discuss the most important relationships, the *related subtopic relationship*, and its inverse. After that, we will argue why we *type* the links in the hierarchy. The last part of this subsection deals with other, optional, relationships giving further structure to the concept hierarchy.

The Related Subtopic Relationship

The relationship that structures our concept hierarchy, which we call *related subtopic*, or *subtopic* for short, is a hierarchical relation, in the sense that more general concepts are related to more specific ones. Most concept hierarchies are structured with an 'is-a' relationship. For example, a *modal logic* is-a *logic*. This indicates that the class of modal logics is included in the class of logics. The hierarchical relation we use to structure our hierarchy is broader than a strict is-a relationship (also called 'subsumption'); besides the is-a relation, we allow the subtopic relation to cover meronymic cases as well. An example of this is the relationship between *compactness* and *logic* in the sense of 'a system of reasoning'; compactness is a related subtopic of logic, but not subsumed by it. The same kind of hierarchical relation is employed by Sanderson and Croft [SC99], who use hierarchies to give an overview of the contents of a large set of documents. The relationship seems to be well suited for this purpose.

A disadvantage of our subtopic relation compared to a strict is-a relationship is that it does not form an inheritance system, as often used for knowledge representation or in object-oriented programming. But since our aim for the concept hierarchy is navigation, not computation, this disadvantage is not serious. It makes it easier for users to navigate through the hierarchy, as they do not have to be aware of the differences between meronymy and hyponymy.

Another consequence of our focus on navigation is that we do not require that the concept hierarchy be a strict tree, that is, a hierarchy where every child (except the root) has only one parent. It might well be possible that both a reader interested in symbolic logic and a reader interested in knowledge representation might want to find out more about predicate logic. That is why we allow a concept to be the subtopic of several other concepts; in this case predicate logic is a related subtopic of both knowledge representation and symbolic logic.

We do not allow cycles in the hierarchy, since cycles disorientate readers and because of the computational pitfalls caused by them. Furthermore, the whole idea of a subtopic relation indicating more specific concepts would be lost with cycles in the hierarchy. Therefore, the concepts in the hierarchy together with the subtopic relationship between them should form a *directed acyclic graph* (DAG): a graph whose edges (the subtopic relation) are ordered pairs of vertices (the concepts), and in which there is no path that starts and ends in the same vertex. Furthermore, we do not allow any concept to be unconnected to the rest of the hierarchy; it should at least have an incoming or outgoing edge. Yahoo! has a similar structure [LF99], but WordNet [Fel98], is a strict hierarchy.[1]

The Related Supertopic Relationship

A relation that is equally useful for navigation is the inverse of the related subtopic relationship, the related supertopic relation. Being its inverse, it can be derived with minimal effort from the related subtopic relationship. This relation gives more context to concepts in the hierarchy, by indicating to readers where they are in the hierarchy and improving their understanding of the concept, in the same way dictionary definitions do: they often give the meaning of words by stating their supertopic or superconcept and restricting and distinguishing it from its siblings by adjectives or relative clauses.

Besides giving more context, supertopic relations also facilitate going back to a previously visited concept and allow for interesting paths, going up and down

[1] To be precise, WordNet consists of several strict hierarchies, with some amount of cross-referencing between them.

between various levels in the hierarchy. For example, from the concept 'knowledge representation' one can go down via the subtopic relation to 'predicate logic' and up via the supertopic relation to 'symbolic logic'.

Typed Hyperlinks

Besides the sub- en supertopic relationship, there are other relationships in the concept hierarchy: the *external relations*, that will be discussed in Section 3.3, and relations internal to the hierarchy but different from the subtopic relationship. We will discuss the latter below, but first address a problematic consequence of having concepts containing many links, and a solution to this problem.

In an electronic environment, relations between concepts are represented by a hyperlink, irrespective of whether this hyperlink is a textual one, or a link on some kind of navigational map. Such a link is a connection between its source and its target, making a jump form the first to the latter possible. In most implementations of hypertext (like the Web), it is not obligatory to attach any further information to such links.[2] However, when there are different types of relationships within a hypertext, as in our hierarchy, a lack of a uniform characterization of each link can cause confusion on the side of the reader.

Therefore we want to have *typed links* in the concept hierarchy, links that do carry an attribute denoting the kind of relationship between the source and the target, thus introducing structure in the links. The most important advantage is that this additional information helps readers to decide whether or not to follow a link. In order to be able to really help readers, there should not be too many types, and the types should be easily distinguishable from each other — both conceptually and visually. Conceptually, it should be entirely clear to a reader what to expect from a hyperlink: does it lead him to a related article or to a homepage of a researcher? By 'visually distinguishable' we mean that whether icons, a textual description, or both are used to indicate the presence of a link, a reader should easily see of which type it is. Otherwise, the benefits of link typing are canceled entirely.

Another advantage of typing is that the overall topology of the hierarchy is simplified: the topology formed by one of several link types is simpler than the topology formed by all the link types together. Also, typed links can decrease the cognitive load of readers, who can select the types of links they want to have visible, instead of all links. More advantages of link typing can be found in [All95, Chapter 1]. Next, we comment on the work of some researchers that have tried to create a link typology, as far as relevant to our work.

Harmsze. Harmsze has conceived a typology of link types for modular experimental science articles [Har00]. In Section 4.3.1. of this thesis she clearly indicates what such a typology should be like, and clarifies the connection between links and relations. She characterizes links by the kind of relation they express, their source and destination, and also meta-information, like the name of the 'author' of the link and the creation date.

Meta-information is not only important with respect to intellectual property issues, but also useful for the reader. As described in Chapter 5, we want to

[2] In HTML, the `<A>` tag can carry attributes indicating the link type, but in practice they are hardly used.

generate some links automatically. The quality of such links might be different from the quality of manually generated ones, so it is useful for a reader to know the (animate or inanimate) author of a link. By not having this distinction explicit he might lose confidence in the links, because of their diverse nature. Therefore we will also attach this kind of meta-information to links in the concept hierarchy.

The main distinction in Harmsze's typology is between two classes of relationships: *organizational relations* and *scientific discourse relations*, each class consisting of several different relation types. The goal of her work in this direction is providing a systematic, fine-grained typology for links between the modules she identifies in her modular model for experimental science articles. Since every module in her model represents a basic kind of idea, it would be interesting to see if the typology can be applied more widely.

Allan. Another typology is introduced by Allan in [All95]. He distinguishes between three main categories, based on the way in which they can be identified: only by humans, by simple computational means or by more advanced algorithms. The respective categories are *Manual*, *Pattern-matching* and *Automatic*, and are further subdivided, which results in a total of 16 different types. In our opinion, however, the type of a link should say something about the relationship between the two documents that are linked, to help the reader decide to follow the link or not. Allan's top-level distinction is not based upon the relationship between the linked documents, and therefore is not useful for our purposes. However, the *Automatic* links, for example, are further divided in link types such as *summary*, *revision* and *equivalence*, which do say something about the relationship between the linked documents, and therefore are more useful. As we will discuss in Section 5.3.4, Allan has conceived interesting algorithms to generate *Automatic* links and determine their type automatically.

Blustein. Blustein [Blu99], distinguishes between *static* and *dynamic* links. For static links, the source and the destination are determined before a reader activates the link, while for dynamic links they are generated by conditions. We do not think this distinction is very important, as most computers are fast enough to hide the difference between them for the reader. Furthermore, this distinction is not useful as it is not mutually exclusive; a hypertext system might run programs to create links and store them once in a while. Those links would both be static, as the destination and source are known before the activation by a reader, and dynamic, as the program uses certain conditions to generate the links. Therefore, the distinction between *manual* links, links created by a person, and *automatic* links, links generated by a computer program, is more appropriate. We also employ this distinction — although not literally — in the hierarchy, by including the author name in the meta-information attached to each link. Like us, Blustein does not try to provide an exhaustive link typology.

Others. Trigg [Tri83] was the first to discuss link typing in his PhD thesis; other typologies were introduced by DeRose [DeR89] and Parunak [Par91].

There seems to be no consensus on which typology is the most comprehensive or useful one, but we do not consider this problematic. The types one distinguishes may vary with the purpose of the electronic publication. We only demand that the distinction between them is clear and easy to grasp for a reader and helps him to decide whether or not to follow a link. Thus, we do type links in the hierarchy, but do not intend to create an exhaustive typology of links types that can occur.

Further Structure

Having discussed the sub- en supertopic relationships and the typing of links, we will now turn to the other types of links that give further structure to the concept hierarchy.

While every concept (except for the root) should be connected to the rest of the hierarchy with at least one subtopic relation, the relations we discuss next are not compulsory. Furthermore, unlike the hierarchical relation, which must form a DAG, the relationships in this paragraph do not have to fulfill such conditions as the main structure of the hierarchy is already provided by the sub- and supertopic relations.

Similarity. The first non-hierarchical relation we want to allow is *similarity*. Concepts in the hierarchy are similar if they share some properties or are analogous or parallel to each other. For example, the similar relation holds between *finite-state machine* and *regular language*. Both concepts are not connected directly in the subtopic hierarchy, but the relationship between them is important and should be made explicit. As we will show in Chapter 5 we can automate (at least partly) the creation of similar links with the help of Information Retrieval (IR) techniques.

Antonymy. Another useful relationship is *antonymy*. Antonymy is a relation between *words* with opposite or contrasting meanings. Learning the antonym of a word not only learns us more about the meaning of the antonym, but also about the word itself [Mue97]. Although there are many different sorts of antonymy, some of which are gradual, we do not try to distinguish between them. Because of this graduality, we will not define antonymy precisely, but will only include antonyms with a concept if a substantial number of people agree that they are antonyms. Note that antonymy is a relation that holds between *words*, not between *concepts*. So if words in the synset of the concept have antonyms, it should which word has which antonym, since in general it is not the case that synonyms have the same antonyms.

Sibling. For readers, it is also useful to know what are the *siblings* of the current concept in the hierarchy: the subconcepts of the superconcept(s) of the concept at hand. However, a sibling relationship does not have to be stored explicitly, as it can be derived automatically from the super- and subtopic relationships.

Other Meanings. Another useful and automatically computable relationship is the *other meanings* relationship. For every name in the synset of the current concept, there should be connections to the other concepts in the synsets of which the name also occurs. For example, the name 'sense' occurs both in the synsets {sense, meaning} and {sense, intention, connotation}. From the concept corresponding with the first synset, there should be a link to the concept corresponding with the second one and vice versa.

3.3 External Connections

As opposed to the links discussed in the previous section, the links discussed in this section are *external* in the sense that they are not between concepts within the hierarchy, but between concepts and targets outside the hierarchy. Within those links, we distinguish between *handbook* links to information in the handbook but outside the concept hierarchy, and *Web* links to information sources on the Web. Again, we make this distinction for the benefit of the reader; for him it is important to know whether it is a link internal to the electronic handbook, or a link leaving it.

Within both types of links, further distinctions are made based on the *kind of target* the link points to. A distinction based on the kind of *relationship* between source and target is not needed anymore, as the external links are in a relationship of aboutness with the concept at hand.

When a reader would like to read about something standing in some kind of other relationship than 'about' with the concept at hand, he navigates via *internal* links to the concept standing in this relationship, and follows *external* links to the desired information. This way of navigating therefore requires that the concept hierarchy covers all the information connected to it (the completeness requirement discussed in Section 3.2.1).

3.3.1 Handbook Links

A *handbook link* is a link from a concept in the hierarchy to a passage in the handbook relevant to the concept. The destination can be of different levels of granularity: a part, a chapter, a subsection, a page, an example, a definition, etc. Ideally, if the structure of the hierarchy corresponds to the handbook, concepts higher in the hierarchy refer to larger fragments in the handbook, while concepts lower in the hierarchy refer to smaller parts. However, we do not expect this to happen, as the handbook chapters are written by different authors, resulting in a different structuring and writing style for every chapter.

We distinguish between the following types of handbook links:

Definition. Definition links are handbook links from concepts in the hierarchy to their definition in the handbook text. This definition might already be included in the description of the concept at hand, in which case such a link is not necessary.

Theorem. Handbook links from the concept at hand to theorems and lemmas included in the handbook text involving this concept are very useful for a reader to gain more insight in the concept.

Discussed-in. The discussed-in links are less specific than the above links, and are links from concepts in the hierarchy to passages in the handbook where the concept is discussed. Those passages can range from small passages of a size comparable to pages in a paper handbook, to larger passages such as sections or even chapters. It should be investigated what is the right passage size to refer to. Concepts higher in the hierarchy will probably be linked to larger passages than concepts lower in the hierarchy.

In Chapter 5, we will discuss the automatic creation of handbook links.

3.3.2 Web Links

We distinguish between several kinds of links from the concept hierarchy to the Web.

Research group. Links from the concept to homepages of research groups doing research about the concept.

Homepage. Links to homepages of individual researchers doing research about the concept.

Tool. Links to software about the concept. For example, at the concept *modal logic* there can be links to theorem provers that can handle modal logics.

Publication. Publication links are links from the concept at hand to publications about the concept. Those links are extremely useful, since there are a lot of online paper repositories, and this number is still growing. Such repositories are maintained by various institutions: publishers (e.g. Elsevier's ScienceDirect), scientific organizations (e.g. ACM's Digital Library) and universities or university departments (e.g. the University of Amsterdam's Beta Preprint Server). Some are accessible for free, some have to be subscribed to or require a separate payment for every downloaded publication. The links should indicate whether the download is free or not. For this purpose, users of the handbook should be able to customize the electronic handbook to indicate to which online services they are subscribed.

Like the handbook links, Web links can also be generated automatically; we will briefly address this issue in Section 5.5.

3.4 Presentation

Now that the structure of the concept hierarchy has been described, we can discuss how we want to visualize it. We will do that in the next subsection. An important but often overlooked aspect of visualization is the evaluation of the interface that is used. In Section 3.4.2 we discuss this topic.

3.4.1 Methods for Visualization

The structure of the concept hierarchy should be visualized and made browsable for the user of the handbook. This visualization is very important, since the manner in which the information is presented strongly influences the ease with which a user can find the information he needs.

The field of information visualization deals with such topics, and is large and rapidly expanding, witness the publication of textbooks on the field (e.g. [War99]) and the yearly IEEE International Symposium on Information Visualization.[3]

[3] `http://www.infovis.org`, visited on January 5, 2001.

Trying to find the optimal user interface for browsing a concept hierarchy is not a goal of this project and would take too much time because of the mentioned user evaluation. Therefore, we have made an inventory of common approaches and chosen one of them. We list the approaches below, after which we discuss our choice. We do not claim this list to be exhaustive; for example, hybrids of several interface types also exist.

Stable. A stable interface shows the entire hierarchy fully expanded, like a table of contents of a paper book [CS94]. By clicking on the nodes in the hierarchy one can access the information connected to the nodes; the only way to navigate within the hierarchy is scrolling. Using this interface, users easily get lost because of the lack of context; if there are many nodes at the bottom of the hierarchy, the screen will be occupied by those nodes so the user loses overview of the high-level structure. Stable interfaces are not very useful for DAG-like structures; the fact that the entire hierarchy is shown and that one child can belong to several parent nodes causes major redundancies, obscuring the high-level structure even more than is already the case with a strict hierarchy.

Yahoo!-like. In a Yahoo!-like interface one navigates through the hierarchy via several pages, where each page represents a category in the hierarchy. Users are shown some *local* context for the category currently displayed: its children in the hierarchy and the path from the root to this concept. There also is a collection of links to Web documents falling under the category. Although Yahoo! has a DAG structure and allows categories to have more than one parent, this is not completely reflected in the interface. Every category in the hierarchy seems to have a 'preferred' parent, which is the only parent shown. So, for example, one can go from the top level to category `Regional` and then to `Geography`. One would expect to be shown the path `Home` > `Regional` > `Geography`. However, `Geography`'s preferred parent is `Science`, so one only gets to see the path `Home` > `Science` > `Geography`.[4]

Expand/contract. This is a very common interface type often used by programs for navigating directory structures. When users start to navigate with this interface, they first see only items at the top of the hierarchy (i.e. the children of the root). By clicking on one of those, the items below the clicked on item appear (the item *expands*). Users can also make the items disappear by clicking the expanded item again (the item *contracts*). This interface gives more context than the above two; depending on how many items the user has expanded, the users sees more or less of this context. This interface type is not well suited for showing a DAG: it is not possible for a node to have several different parents visualized at the same time.

Menu. A menu interface is a restricted form of expand/contract interface, in which only the path in the hierarchy from the selected node until the root of the hierarchy is visible. Most menu structures of computer programs work this way. Like an expand/contract interface, it is not suited to visualize a DAG.

[4] `http://www.yahoo.com`, visited on January 5, 2001.

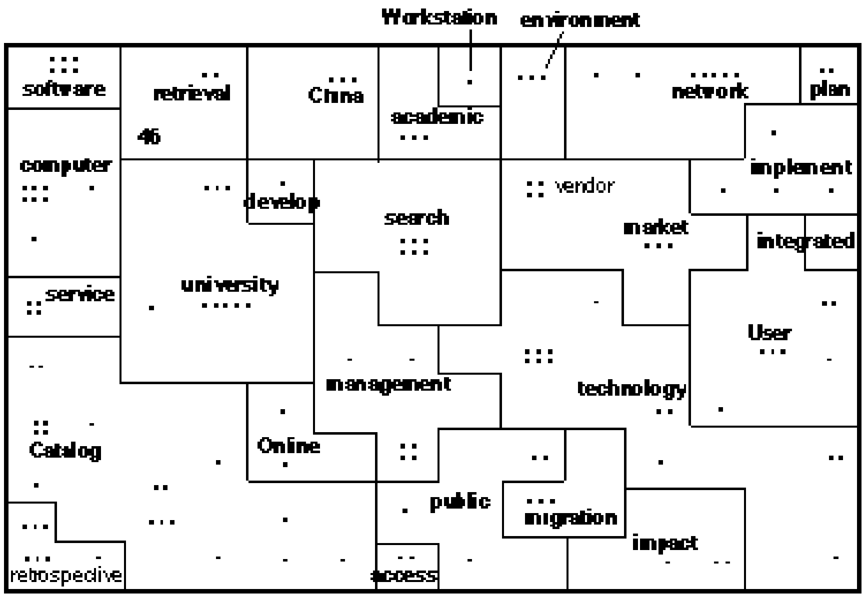

Figure 3.1: Kohonen map.

Multipane. In a multipane interface, different levels of detail within a hierarchy are shown in separate panes, which are all visible at the same time. It gives more context than an expand/contract interface, as items higher in the hierarchy remain visible no matter what the user selects to be expanded.

Maps. Sometimes maps are used to visualize groups of documents. They mostly display overviews of large collections, automatically generated with polythetic clustering techniques. The topology of the map is determined by the clusters and the calculated distance between them. For instance, Kohonen maps have been used for this purpose (Figure 3.1).[5] An evaluation of the usefulness of a Kohonen map versus a Yahoo! category hierarchy showed that the latter presentation of the hierarchy is more useful for navigation [Hea99]. If there are strict hierarchical relations between monothetic concepts, other means of presentation are more useful, since a map does not show the hierarchical relationships.

3-D Visualizations. Nice three-dimensional structures are sometimes used for the visualization of hierarchies. For example, the Cat-a-Cone system [HK97] integrates searching and browsing in a very useful manner, as shown in Figure 3.2. The three-dimensional display of the category hierarchy gives a very clear overview of the position of retrieved documents within the concept hierarchy, thus providing additional information about the documents.

From the above mentioned interfaces, only the Cat-a-Cone and the Yahoo!-like interface can show both information about the hierarchy and textual content: in

[5]The picture is from Xia Lin, *Searching and Browsing on Map Displays*, `http://www.uky.edu/~xlin/asis95.htm` (visited on January 5, 2001).

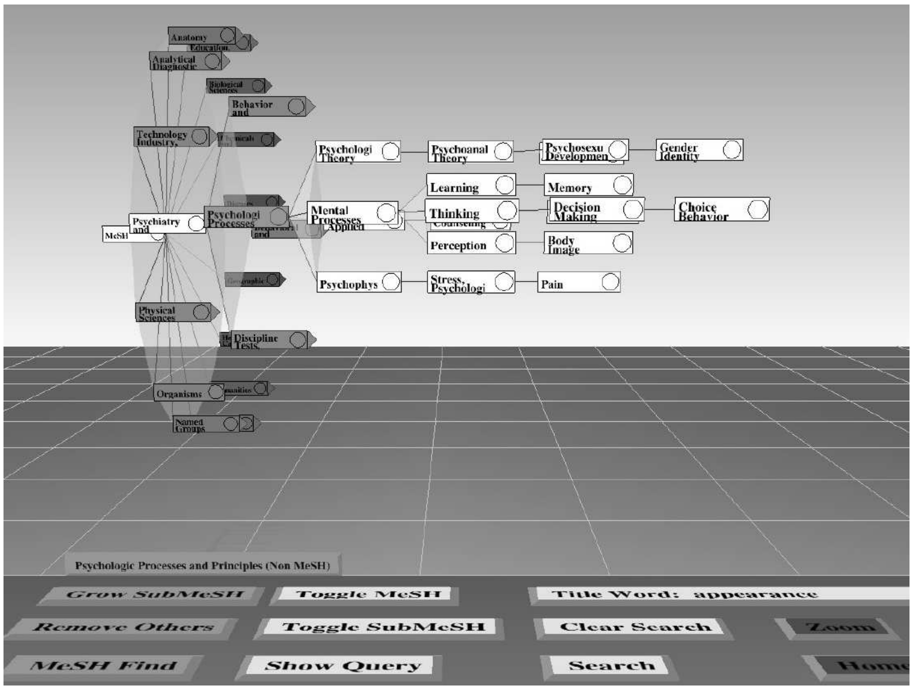

Figure 3.2: The Cat-a-Cone interface.

Yahoo! links to Web sites and a description of the site, in Cat-a-Cone documents at that position in the hierarchy (not shown in Figure 3.2). But the other ones, which only show information about the hierarchy can be easily adapted to form a so-called *overview plus details* interface [Hea99], where the 'details' consist of a text screen showing the description of the concept, hyperlinks to the handbook and Web-based information sources.

We think a Yahoo!-like interface is most suitable to visualize the concept hierarchy of an electronic handbook. One reason for this is that the hierarchical information and the information about the concepts are tightly interwoven this way. This is especially important since evaluations of graphical overviews have given only negative evidence of their usefulness [Hea99]; users seem to need textual information to get a clear overview of the contents of a document collection. Another advantage of Yahoo!-like interfaces is that they can allow for concepts having more than one parent.

A disadvantage is that it is not easy to jump from one part in the hierarchy to another part, which is easy in graphical overviews. But this is not too big a problem, since we allow non-hierarchical links in the hierarchy, via which one can visit related concepts in another part of the hierarchy. If the hierarchy is well structured and contains many useful links, users will not need such jumps.

3.4.2 How to Evaluate Interfaces

In the previous subsection we discussed some different interfaces and our decision to use a Yahoo!-like interface. This decision was not supported by much evidence indicating that this is a good type of interface, since few evaluations of interfaces with real users can be found in literature.

This is probably due to the fact that it takes a lot of time to collect data involving human users. Also, when the data has been collected, it is often difficult to extract relevant information from it. It is difficult to find patterns in user behavior, so most research only draws conclusions about directly observable behavior, like the time needed to answer questions [CS94], or the number of questions answered correctly within a certain amount of time [McE99]. Another problem is that it is very much dependent on the type of user, whether an interface is useful or not. User characteristics such as spatial ability and verbal aptitude result in different degrees of usefulness for different interfaces [Hea99].

It is important to evaluate by monitoring objectively observable behavior, since asking users to rate a hypertext system does not provide reliable information. User performance with hypertext systems and their satisfaction or perceived performance is often quite dissimilar [Blu99].

An example of the evaluation of a user interface is the research of Chimera and Shneiderman. In [CS94], they describe the evaluation of three commonly used graphical interfaces for browsing through a hierarchical structure. The hierarchical structure used in their evaluation was a table of contents; the interfaces were a *stable interface*, an *expand/contract interface* and a *multipane interface*, as described in the previous subsection.

The experiments were carried out with 41 undergraduate students, most of which had some experience using computers. They had to perform 9 question answering tasks, ranging from simple to more complex browsing tasks. An example of a simple question is 'What is the title of subsection 8.2.3?'; a complex one is 'Does this table of contents contain more information about radar or emergencies?'. All the questions were only related to elements of the table of contents, not to the text to which the table of contents gives access.

The authors' hypothesis that the multipane interface was the best, the expand/contract less good and the stable interface the worst, was confirmed by the evidence. The multipane interface was only slightly better than the expand/contract interface.

Although the above conclusion by itself is not very surprising, a lot can be learned from the set-up of the experiments by which it was achieved. The authors have conceived good questions, ranging from simple to more complex. Furthermore, besides measuring the time in which the answers were found by the subjects, they also observed how they arrived at the answer. An example of those observations is that the subjects physically moved away from the screen when a lot of items were inserted in the expand/contract interface. This raises interesting and important questions about the animation techniques used for this interface. Exploring such questions might result in a better interface.

3.5 Discussion

In the previous sections we discussed the ingredients of a concept hierarchy for navigation: the concepts, the links between the concepts, and the links leaving the concept hierarchy. Figure 3.3 summarizes this.

In Figure 3.4 we show an example of a concept in the hierarchy. The textual way of visualizing a concept closely resembles the way topics are visualized in topic hierarchies like Yahoo!, and the implementation of our concept hierarchy

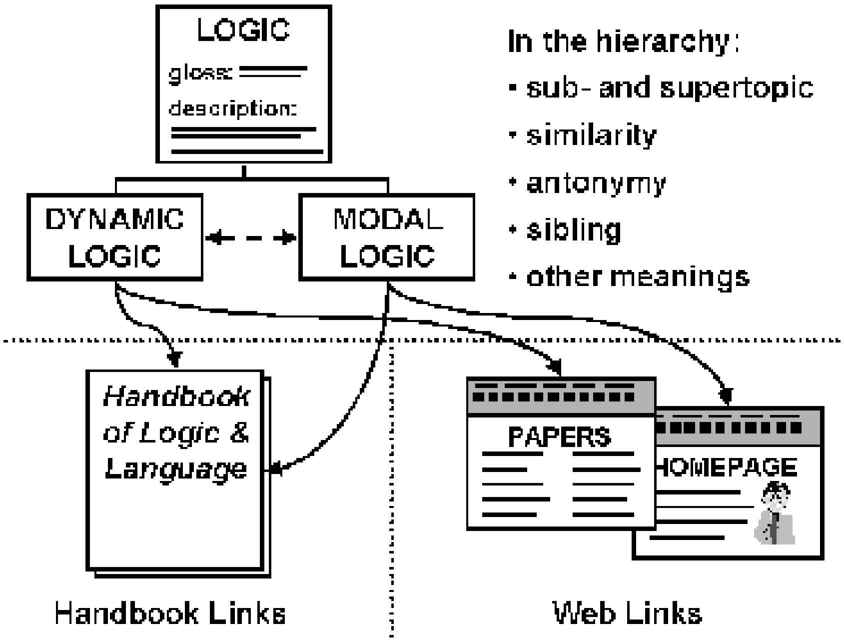

Figure 3.3: Links inside and leaving the concept hierarchy.

will be in this fashion.

The example contains most, but not all, ingredients of concepts in the hierarchy, since it is not the case that every concept should contain all ingredients. We only require that every concept should be connected to the rest of the hierarchy through the subtopic relationship, and contains a short gloss to aid the user navigating through the hierarchy. The other ingredients should be added whenever useful.

Concepts that are of the same level of granularity as current entries in the glossary of the Handbook (see Section 4.2.1), should contain besides the gloss a description of the concept like a description in the Handbook's glossary. Such entries may also contain a lot of subtopic links and external links, as they represent core concepts in the field. However, other entries are at a lower level, such as some entries of the Bluffer's Guide to Computational Semantics that have a description of one or two lines (see Section 4.2.2), and therefore do not need such an extensive description and so many in- and external links. The concepts of which they are a related subtopic can provide enough information to get an idea of the meaning of the concept.

The concepts that carry most information will probably turn out to be located somewhere in the middle of the hierarchy: not so general as to need too large a description, and not so specific to be a concept of minor importance. Very general concepts might of course get a large description, but this will not really facilitate navigation. Furthermore, a lot of their meaning can be understood from the links to their subconcepts.

We focus on using the concept hierarchy as a navigation tool. But as soon as we have a hierarchy at our disposal, we might concentrate further research on using it in another way. For example, we might use it as a textbook; higher level concepts explain a general concept, while lower level concept specialize on more specific topics. In order to be useful as a textbook, we should provide one or more 'reading paths' through the hierarchy, taking into account that some concepts should be known before others can be taught. In general, a parent concept should be known before one starts to learn about its child concepts, but there might be more such relations between concepts in different parts of the hierarchy. Two obvious sorts of paths are *breadth-first paths*, in which first many general concepts are visited before the more specialized child concepts are visited, and *depth-first paths*, who lead the reader first through all the children of a more general concept, before another general concept is visited.

Now that we have described the structure of the concept hierarchy, we will next give attention to two important issues concerning the hierarchy: how to construct and maintain it (Chapter 4), and how to link it to the handbook and the WWW (Chapter 5). Both are quite labor intensive tasks, so we describe tools that can reduce the work involved.

Modal Logic

synset: Modal Logic

A system of logic whose formal properties resemble certain moral and epistemological concepts.

You came here via:

- ▶ System | Logic | Symbolic Logic | Modal Logic

Position in the hierarchy:

- ▶ Computer Science | Artificial Intelligence | Knowledge Representation | Modal Logic
- ▶ Computer Science | Artificial Intelligence | Logic | Knowledge Representation | Modal Logic
- ▶ Mathematics | Logic | Modal Logic

Description:

Extension of propositional logic with two unary sentential operators $\Box$ and $\diamond$ that are intended to capture the meaning of "it is necessary that" and "it is possible that". Modal logic gained a lot of momentum with the development of *Kripke semantics*, which was developed independently by Kanger, Hintikka and Kripke in the late 50's.

Subtopics	**Siblings**		**Similar**
Alethic Logic	Boolean Logic	Higher Order Logic	Intensionality
Deontic Logic	Conditional Logic	Inductive Logic	
Doxastic Logic	Dynamic Logic	Intermediate Logic	**Antonyms**
Epistemic Logic	Feature Logic	Intuitionistic Logic	
Kripke Semantics	Fuzzy Logic	...	
			Other meanings

Handbook

Discussed in: 2.4.8 – Arrow logic and modal state semantics.
Discussed in: 8.3.2 – Modal Feature Logics.
Discussed in: 16.3.2 – Modal Logics for formalizing temporal relations.
Definition: 2.3.1 – Translation from propositional modal logic to predicate logic.

WWW

Research Group	**Homepage**
Institute for Logic, Language and Computation	Yde Venema
	Maarten de Rijke

Publication

Advances in Modal Logic (AiML) | Modal Logic (Blackburn, de Rijke, Venema)

Tools

- ▶ BLIKSEM Hans de Nivelle's resolution based theorem prover for modal logic and first-order logic with equality.
- ▶ **KtSeqC** A theorem prover for the minimal tense logic Kt.
- ▶ **Logics Workbench (LWB)** A sequent based theorem prover for a range of propositional logics, including modal logics, temporal logics, intuitionistic logics and non-monotonic logics. WWW interface

Author: Jon Ragetli, Created 21-11-00, Last change 24-11-00

Figure 3.4: Example concept in a textual, Yahoo!-like, interface

Chapter 4

Construction and Maintenance of the Concept Hierarchy

4.1 Introduction

As described in the previous chapter, we want to have a concept hierarchy for navigating through the handbook. Currently, we are constructing this hierarchy manually, with the help of existing material: The Bluffer's Guide to Computational Semantics [Fra96], and the glossary of the Handbook of Logic and Language [Gro97]. Both are discussed in Section 4.2; the manual construction of the hierarchy and the current prototype are discussed in Section 4.3.

In Section 4.4 we describe algorithms for the automatic construction of monothetic hierarchies. Since we are of the opinion that current tools do not create hierarchies of sufficient quality to be used for navigation, we will use them as tools to help a human editor build and maintain the hierarchy, but not to construct the hierarchy without supervision.

Finally, in Section 4.5 we discuss how to evaluate concept hierarchies, both empirically and by using metrics. In Section 4.6 we conclude.

4.2 Available Material

In this section we describe material used for the initial construction of the concept hierarchy. We mainly use it for the construction of the glosses and descriptions, because it does not contain much explicit information about relations between concepts.

4.2.1 Glossary of the Handbook of Logic and Language

The Handbook of Logic and Language contains a glossary of key terminology in the field of Logic and Language, written by Willem Groeneveld [Gro97]. The idea behind the glossary was both to provide the meaning of important terms, and to give some attention to topics that the editors wanted to include in the handbook, but that could not be discussed in its chapters. In this latter respect, our concept hierarchy will be different. The handbook's glossary does not perfectly match the handbook's contents, but a good match is really necessary for a concept hierarchy that has as its main purpose navigation and access. As such, it should be an abstract and coarse-grained mirror of the handbook's contents. However, when the concept hierarchy also provides pointers to the Web, it should not restrict itself to mirror the handbook, as long as it mirrors all the material it gives access to.

Figure 4.1 displays some example glossary entries of the Handbook of Logic and Language [Gro97], from pages 1182–1183. The glossary contains 87 entries, 38 of which really contain a description of an important term. The other 49 'empty' entries only refer to other entries. There are two forms of referring; the first is as in the case of the entry *cataphora*, which refers to *anaphora*. Here, the notion of *cataphora* is explained somewhere in the descriptive text of *anaphora*. The other form of referring is as in the case of the entry *Boolean algebra* referring to *algebra* (not shown in the figure), where *Boolean algebra* is a sub-entry of *algebra*, and is typographically distinguishable as such (in the same way as *finite state machines* and *push-down automata* are sub-entries of *automata theory*, shown in Figure 4.1).

In addition to descriptions of key notions within the Logic and Language field, the glossary has an important reference function to the literature: it contains 357 references to 293 different publications. Every non-empty entry except for one (*language acquisition*) refers to at least one publication in the literature. Of course the entries also refer to the Handbook of Logic and Language, but not in the same amount as to the literature: there are only 33 references to handbook chapters or sections. Two chapters (4 and 11) are not referred to at all by the glossary.

The entries in the glossary also refer to each other. The glossary contains 102 such references. Part of those are the 49 references of the items with no description (like the mentioned *cataphora*); the other part consists of references from a description to another glossary entry. They are typeset in a bold font and can be found either in the descriptive text itself (like the reference to **recursion theory** and **complexity theory** in the descriptive text of automata theory) or below the description, in sentences such as 'see **other entry**'.

The amount of descriptive text with the entries varies considerably. Some entries, like *category theory*, get a description of less than 10 lines, while the biggest entry, *logic*, occupies 4 pages of text. Although some glossary entries have a quite short description, the glossary clearly is more than a dictionary for quickly looking up a short description of the meaning of a term. The descriptions explain the meaning of the terms quite extensively and indicate their position in the wider context of the scientific discipline. Concepts in our concept hierarchy that have approximately the same level of generality as glossary entries, will carry an extensive description and many links, both internal and external, as they are the key concepts in the Logic and Language field.

Most entries do not have any structure, but some of them contain several sub-entries. An example of this is *algebra*, with sub-entries like *lattices*, *Boolean algebras* and *Lindenbaum algebras*. *Logic* is the only entry with two levels of sub-entries; it is divided into six sections (e.g. *Classical logic*, *Important concepts and results*), which are again divided in subsections (for instance *first-order logic*, *completeness*). This subdivision of the entries provides us with useful

automata theory Branch of logic and theoretical computer science that studies mathematical models of computing machines. The subject is closely connected with **formal language theory**; a common concern of these fields is the development of efficient algorithms for the automatic recognition of different types of languages. The subject is also closely related to **recursion theory** and **complexity theory**.

Some well known types of automata are the following (for precise definitions we refer to the literature):

finite state machines have a finite number of states, one of which is the initial state, and some of which are accepting states. A transition from one state to another can only be based on reading one input symbol. This next state function may be deterministic or non-deterministic. Somewhat surprisingly, this makes no difference for the recognizing power of finite state machines: they recognize exactly the **regular languages** (the Type 3 languages of the Chomsky hierarchy).

push-down automata have a finite set of states, an input tape, and a

⋮

linear bounded automata are a restricted type of Turing machines that recognize exactly the **context sensitive languages** (Type 1 languages).

Turing machines are at the top of the hierarchy, in the sense that they have

⋮

theory and complexity theory.

Literature: (Hopcroft and Ullman, 1979; Lewis and Papadimitriou, 1981; Partee, Ter Meulen and Wall, 1990).

Boolean algebra see algebra.

cataphora see anaphora.

category theory Branch of abstract mathematics that studies categories, which are defined as a collection of two kinds of entities called objects and morphisms. Typically, the objects are (structured) sets and the morphisms are (structure preserving) mappings between these sets. Category theoretical techniques are used in mathematical areas such as topology and algebraic geometry, but also in theoretical physics. In logic, category theory is applied in advanced studies in model theory and set theory, and is relevant in the model theory of intuitionistic logic.

Literature: (Goldblatt, 1984; Kock and Reyes, 1977; Lambek and Scott, 1986; MacLane, 1971; MacLane and Moerdijk, 1992; Moerdijk, 1995).

Figure 4.1: Example glossary entries from [Gro97]. The dots represent parts we left out.

information about the subtopic relationship.

Visualization of Relationships

In order to investigate the relationships that are present between entries in the glossary of the Handbook of Logic and Language [Gro97] and other entries, the handbook chapters and the literature, we programmed a Java applet [DD97] for visualizing those relationships.[1] Most of the relationships could be extracted automatically from the glossary text by Perl programs [SC97], although some manual corrections had to be made as well.

Figure 4.2 shows the references from the glossary to the handbook. Boxes containing text represent glossary entries, while the numbered boxes are the handbook chapters. The picture does not show that all the references from the glossary are to entire handbook chapters (e.g. 'see this Handbook, Chapter 1'). The only exception is the reference from *Cylindric algebras* (in the *Algebra* entry) to subsection 2.4 of Chapter 7.[2] The chapter referred to most often is Chapter 3 on Discourse Representation Theory (5 times); most chapters are referred to only once. Somewhat surprising is the fact that *logic*, the biggest glossary entry, only refers to Chapters 14 and 20, and that no other entries refer to those Chapters, giving the impression that *logic* is not a central topic of the handbook. On the other hand, maybe the concept 'logic' is too general and discussed in too many parts of the Handbook, to have a lot of references to the handbook text.

References from the glossary to the literature are shown in Figure 4.3, where the textboxes represent glossary entries and the small squares are the cited works. Having two glossary entries connected via a square therefore means that they cite the same work. Not shown is the *language acquisition* entry, since it does not refer to the literature.

As expected, *logic* plays a central role here: it contains a lot of references, and shares a lot of citations with other glossary entries, which demonstrates the importance of logic in the handbook. It is also no surprise that *formal language theory* and *automata theory* (almost invisible behind *formal language theory* at the bottom of the figure) have a lot of references in common, although one might expect them to share some references with *complexity theory* on the upper right of the figure. There are 25 entries in the large 'web' with *logic* and *intensionality*; still, there are 10 entries sharing no references with other entries at all.

The internal structure of the glossary, the references from glossary entries to each other, are visualized in Figure 4.4. Again, logic has a dominant position. All three clusters have a well-defined topic; there is one about formal languages and automata, one dealing with pragmatics and one dealing with logic. However, many glossary entries do not refer to each other; a situation that we want to avoid in our concept hierarchy.

[1] They are accessible from `http://www.illc.uva.nl/~ragetli/` (visited on January 5, 2001).

[2] In Figure 4.2 this is shown as an arrow from *Algebra* to Chapter 7, since the figure does not show subsections of chapters and sub-entries of glossary entries.

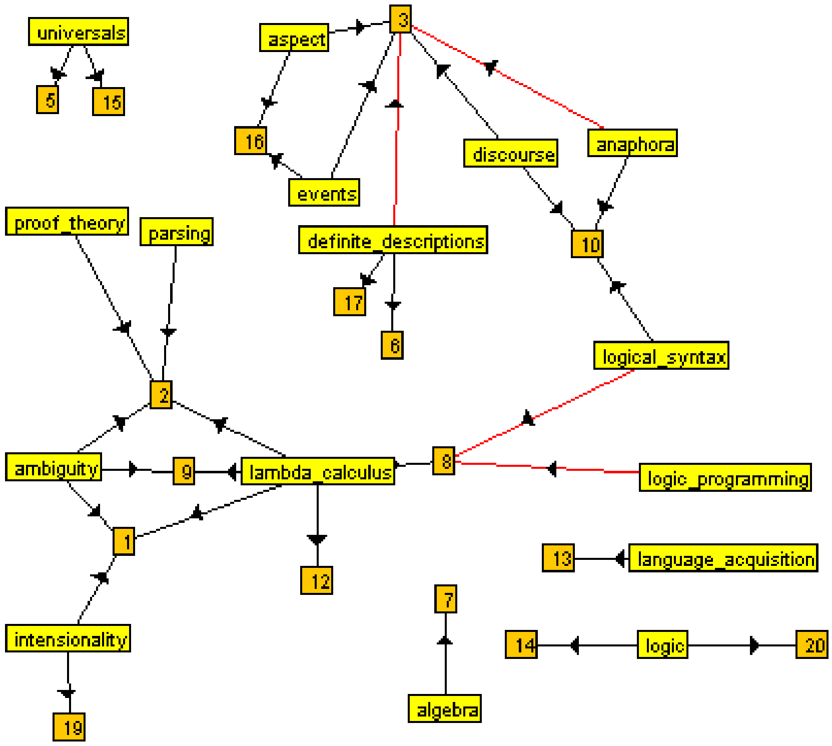

Figure 4.2: References from the glossary to the handbook. Non-referring glossary entries have been left out.

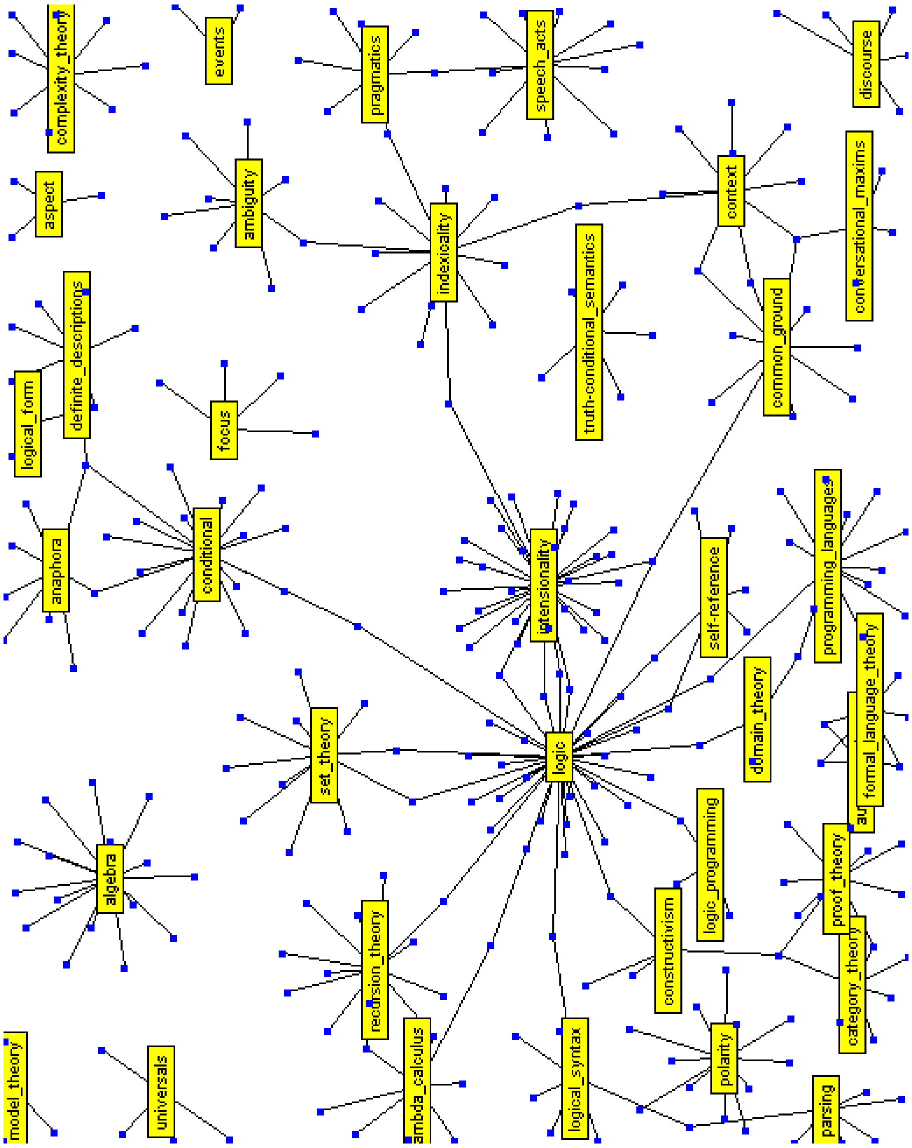

Figure 4.3: References from the glossary to the literature.

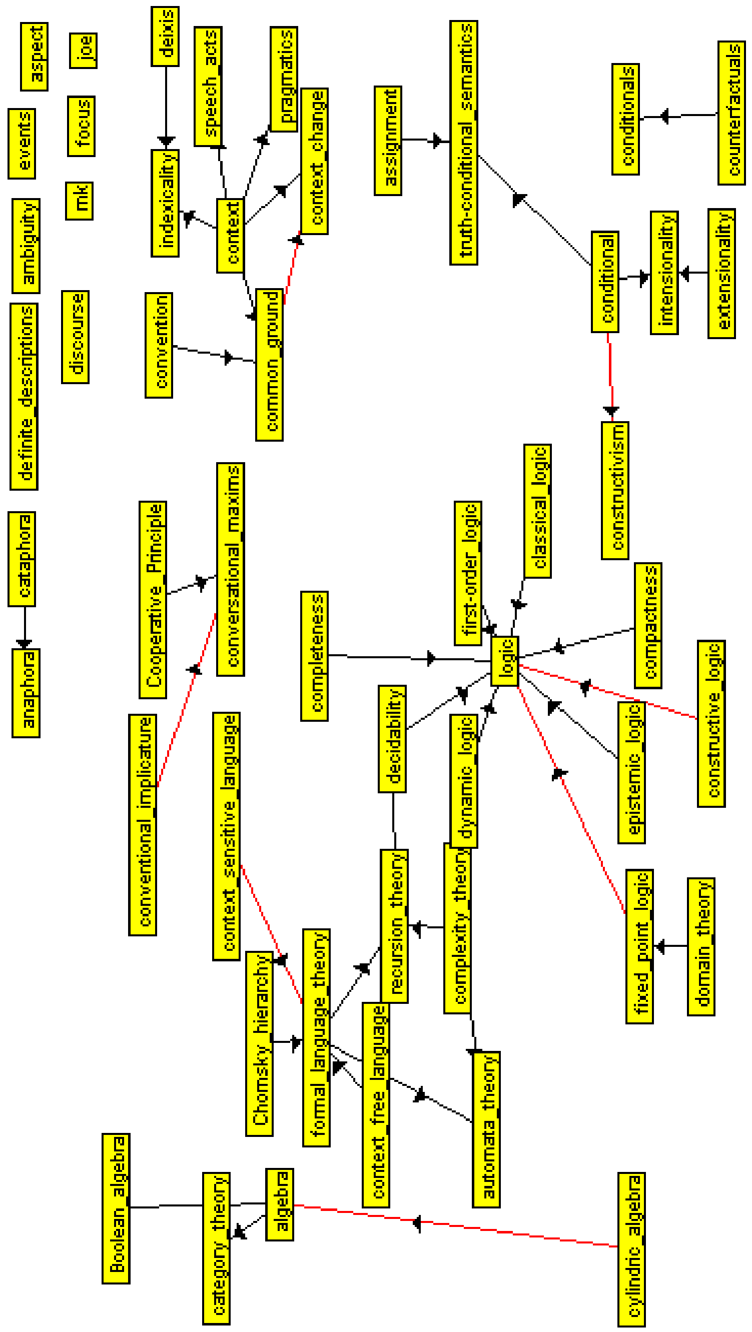

Figure 4.4: References from glossary entries to each other.

antonymy Meaning opposition. In the simplest cases, two concepts are antonyms if their extensions partition a certain domain of application. Example: *male* versus *female.* There are also less simple cases, clearly in the same family, but less easy to characterize semantically: *fall* versus *rise*, *come* versus *go*, *back* versus *forth.* One of the task of lexical semantics is to spell out the precise connections in each of these cases. See also MEANING RELATION.

application Antonym: abstraction. Application is the process of combining a functor with an argument.

artificial intelligence Systematic attempt at making machines think for themselves.

aspect In English: perfect and progressive.

Further reading: Verkuyl [68].

Figure 4.5: Example glossary entries from [Fra96].

4.2.2 Bluffer's Guide to Computational Semantics

Another information source for the concept hierarchy is the Bluffer's Guide to Computational Semantics [Fra96], a glossary resulting from the FraCaS project in which Jan van Eijck was involved. The purpose of that glossary was to establish a common vocabulary between members of different traditions in computational semantics research, in order to enhance their mutual understanding. Figure 4.5 shows a few entries from the Bluffer's guide.

This glossary is quite different from the Handbook's glossary. The number of entries is much higher, 227, and only 13 of those are empty, solely referring, entries. The length of the description with each entry is generally shorter than in the handbook's glossary: it ranges from one line (e.g. *synonymy*) to one page (e.g. *theorem proving*). The Bluffer's guide is more a mixture of a dictionary and a glossary such as that of the Handbook of Logic and Language. Some entries are very short and of minor explanatory value, but still useful for a quick look-up of a term's meaning, while others contain an extensive and clear description, often with a number of examples. The short entries in the Bluffer's guide are useful for inclusion in the concept hierarchy, but mostly at lower levels, and with a small or no description.

Unlike the Handbook's glossary, there are no entries which are divided into sub-entries, so the Bluffer's Guide provides no information about the subtopic relationship. Contrary to the Handbook's glossary, it does provide explicit information about the *antonym* relationship (e.g. *application* versus *abstraction*, *assertion* versus *presupposition*).

Of course, it does not refer to chapters of the Handbook. And its references to the literature are less numerous than with the handbook; there are only references to 69 different publications, and a lot of entries do not refer.

4.3 Constructing the Hierarchy

In this section, we will first discuss the manual construction of the hierarchy as it currently takes place, and after that the Prolog implementation we now have.

4.3.1 Manual Construction

Currently, domain experts at the ILLC are constructing the concept hierarchy for navigating through the handbook. At the time of writing (October, 2000), the hierarchy consists of 445 concepts, 439 subtopic relationships and 40 glosses, mainly provided by Maarten de Rijke. We do not provide any descriptions of entries yet. We plan to provide part of them ourselves, if possible by adapting and updating existing glossary entries. The focus at the moment is on the construction of the hierarchical relationships. When the hierarchy is of sufficient size, we will select the concepts that should get an extensive description. Those will probably include most of the entries in the glossary of the Handbook, as they are important concepts in the subject area of the Handbook. We also expect them to be at an intermediate level in the hierarchy: general enough to deserve an extensive description and specific enough for this description not to be overly general and almost meaningless.

The idea is that the L3 group will exploit their domain expertise to provide a number of example descriptions, and that after this additional ILLC members will provide concept descriptions. This requires, of course, that we must have clear guidelines for contributing to the hierarchy, and that we have an adequate implementation of it. We will return to this in Section 6.2, where we discuss the next steps to be taken in the L3 project.

Although we think the hierarchy should be constructed by a human editor to guarantee high quality, the editor can get substantial support of automatic tools. In Section 4.4 we will further discuss this matter.

4.3.2 The Current Prototype

The current prototypical implementation is in the Prolog programming language [Bra90]. Because of its declarative nature, this logical programming language is very suited for handling the static data about the hierarchy. Furthermore, it is very easy to program search procedures, as Prolog uses depth-first search in the evaluation of programs. Prolog also allows for rapid and easy prototyping, the main purpose of the current implementation.

A disadvantage of using Prolog for building the hierarchy is that it does not require any structure in the hierarchy, so we had to manually add procedures to check for cycles in the subtopic relation. At the moment, we have an interface to this database similar to WordNet's *textual* interface (WordNet also has a *graphical* interface for browsing). We programmed procedures for showing the glos(ses) of a certain word or synset, showing all the siblings of a word in the hierarchy, showing the entire hierarchy below a certain synset, etc. Figure 4.6 shows the output of the predicate `position`, which shows the position of a certain concept in the hierarchy.

Now that the hierarchy is growing bigger, we already feel the need to store the data in a more systematic manner, in order to avoid duplicates, and other problems that occur when building large databases. Although Prolog *allows* for checks to avoid such problems, the self-written procedures to verify this must be invoked manually — something that is easily forgotten. Especially when authors external to the L3 group will be providing information on concepts this should be taken care of; having authors do duplicate work will not increase their willingness to cooperate.

```
?- position('modal logic').
Synsetname:  modal_logic_0
       Synset:  [[modal logic]]
       Glos:  the logical study of necessity and possibility

Synsetname:  modal_logic_1
       Synset:  [[modal logic]]
       Glos:  a system of logic whose formal properties resemble
certain moral and epistemological concepts

----
Type the sense of the word which position you want to see...
modal_logic_1
<< modal_logic_1
    << knowledge_representation_0
       << artificial_intelligence_0
          << computer_science_0
       << logic_2
          << artificial_intelligence_0
             << computer_science_0
    << logic_4
       << mathematics_0
    << symbolic_logic_0
       << logic_1
          << system_0
------------------------
>> modal_logic_1
    >> alethic_logic_0
    >> deontic_logic_0
    >> doxastic_logic_0
    >> epistemic_logic_0
    >> kripke_semantics_0
```

Figure 4.6: Output for the predicate `position('modal logic')` of the Prolog implementation. Since 'modal logic' occurs in two synsets, the user is asked to indicate which concept he wants to see the position of. In this case, 'modal_logic_1' is chosen. The upper part shows the hierarchy above the modal_logic_1 concept, the lower part the hierarchy below it. The hierarchy is not a tree as modal_logic_1 has three different parents and knowledge_representation_0 has two parents.

Therefore, besides the extension of the contents of the hierarchy, an important task to work on is to enter the data into a database system that can be connected to the web. The current data representation in the Prolog files is implementation independent enough to have an automatic format conversion to enable storage in a database system. But first, of course, a good design of the database must be made.

4.4 Supporting Manual Construction by Automatic Construction

4.4.1 Introduction

In the next two subsections, we describe two methods to automatically generate monothetic concept hierarchies, which can be used for the manual construction and maintenance of the concept hierarchy. Those methods are useful as they generate hierarchies without any user effort. But we do not consider the generated hierarchies good enough to be employed for navigation through a handbook. Besides this, for a lot of entries in the hierarchy, a description and a gloss will be needed, which are not supplied by the above algorithms. So there should always be a human hierarchy editor, who can be supported by the hierarchy generation tools. Apart from algorithms to automatically construct hierarchies, there are other means to aid the hierarchy editor: metrics about the structure of the hierarchy, which we will discuss in Section 4.5.2.

If a concept hierarchy would have to be created only once, the use of automated tools for hierarchy construction to aid this process might be too big an effort. However, the hierarchy should be maintained on a continuous basis. The reason for this is that we want the hierarchy to be an abstract and structured mirror of the material it gives access too. And this material is subject to change of course, in particular material on the Web, where more and more research papers can be found. Also, the scientific area underlying the hierarchy can change. The Logic and Language community, for example, has shown a renewed interest in Game Theory over the last few years. Such interest generally results in the use of new concepts, and sometimes also in the use of other names for existing concepts. Therefore, this research area will occupy a larger part of the hierarchy than it would 10 years ago.

On the other hand, when research in a particular area dies out, or a paradigm shift occurs, this area will gradually become a relatively smaller part of the entire discipline, and as a consequence will deserve a less prominent position in the hierarchy.

Algorithms for building a hierarchy have as input a collection of documents. In the case of the concept hierarchy we are building, we can use several document collections for the algorithm to work on. The most obvious document collection is the text of the handbook itself. But the collection can also be augmented with the publications that are referred to by the handbook chapters or already existing concepts. But it is not likely that using those texts alone will extend the hierarchy, as the material was already linked to the handbook or hierarchy in some way. More knowledge can be obtained if we use entire online repositories with publications or preprints on the discipline of the handbook.

Automatic hierarchy builders can support the editor in several ways, as listed below.

- The hierarchy that the editor constructs should be *complete*, in the sense that all the contents of the handbook are covered by it. The hierarchy editor can check this by comparing automatically generated hierarchies to the hierarchy he has constructed. As the hierarchy generation tools only work with the available documents and do not use additional knowledge, the generated hierarchies exactly cover that data, and are therefore useful for checking completeness. Concepts occurring somewhere in the generated hierarchy, but not in the manually edited one, should probably be added. The generated hierarchy also suggests at which position in the hierarchy it should be added.

- Automatically generated hierarchies can be used to verify the *correctness* of the concept hierarchy, in other words, whether it does not contain concepts not covered by the handbook's contents. If a concept occurs in the manually generated concept hierarchy and does not occur in the automatically generated hierarchy, this might be a superfluous concept.

- If an automatically generated hierarchy positions a concept at a different place than the human editor, he can consider positioning the concept somewhere else in the hierarchy. It can also be caused by the fact that the editor is using another meaning of a polysemous word than the meaning that occurs in the handbook. In this case, the editor should include the concept with the sense of the word used in the handbook. He need not remove the other sense from the hierarchy, since it can be useful to clarify the meaning of the sense used in the handbook.

We will now discuss two recent methods for the construction of hierarchies.

4.4.2 Sanderson & Croft's method

An algorithm to derive monothetic concept hierarchies from text is presented by Sanderson and Croft in [SC99]. They aim to generate a hierarchy with the same subtopic relation we employ in our concept hierarchy, a mixture of hypernymic (is-a) and meronymic (part-of) relationships.

Their method works as follows. In response to a query,[3] the set of 500 top-ranked documents is taken into consideration. Terms have to be selected to become the concepts in the hierarchy. For this, they use the terms in the query, and add a number of about 70 other terms created by the Local Context Analysis method for query expansion. Some more terms are added by extracting terms from the passage that is most similar to the query in every document. If the frequency of a term in such a passage is more than 10 times the frequency of this term in the document collection, it is added to the term collection.

With the resulting term collection, the process of building the hierarchy is started. For every pair of terms it is verified whether they stand in a hierarchical relation,[4] which is defined as follows:

[3] They use the Text REtrieval Conference 1997 (TREC) document collection and queries [VH97].

[4] They call it *subsumption*, but as mentioned before it is not a strict subsumption relation.

Term x subsumes term y if $P(x \mid y) \geq 0.8$ and $P(y \mid x) < 1$.

In words: x subsumes y if maximally 20% of the set of documents containing y lies outside the set of documents containing x.

These comparisons result in a hierarchy quite different from polythetic hierarchies generated by other methods. A small-scale user evaluation shows that the generated hierarchies do really make sense.

To be applied to the construction of a concept hierarchy for the Logic and Language area, some adaptations should be made to Sanderson and Croft's algorithm. We do not have a query whose terms can be used as potential concepts in the hierarchy. For the handbook, the index terms that are already present might be used for this purpose. Alternatively, we can construct a preprocessing step to the algorithm that includes words, for example on the basis of their frequency. The people constructing the hierarchy may as well add the concepts they have already positioned in the hierarchy, in order to check whether their position is good. Local Context Analysis and adding terms that occur frequently in certain passages can still be applied to increase the number of terms with which to form the hierarchy.

4.4.3 Sequitur

Another method for automatically building concept hierarchies is presented in [NMWP99]. Like in the above approach, the authors use a bottom-up algorithm. It is based on the Sequitur algorithm to detect hierarchical structure in a sequence of tokens. The generated hierarchy is also based entirely upon the collection for which it has been built and requires no additional knowledge.

By taking words as the individual tokens, the algorithm makes a grammar for the entire set of documents on which it is invoked. Individual words are the terminal symbols of this grammar. To construct the concept hierarchy, for every word occurring in the collection the grammar rules in which it plays a role are retrieved. In those rules, words adjacent to the word searched for give a refinement of the concept represented by a word. For example, selecting 'grammar' one receives a list with concepts like 'attribute grammar', 'context free grammar', and so on, as there are rules where 'attribute', or 'context free' are adjacent to 'grammar'. In this way, those two become subconcepts of 'grammar' in the hierarchy.

A disadvantage of this algorithm is that it does not create one hierarchy; in fact, for each word occurring at least a (user definable) number of times in the document collection, it produces a hierarchy by looking for refinements for the word. This way, it cannot clearly show the relationship between different concepts represented by different words. Another drawback is that it will not recognize synonyms as such, something that is needed for the construction of a concept hierarchy. As it depends on literal words, for example 'Turing machines' and 'finite-state automata' will not be related in any manner, while ideally they should be sisters in a hierarchy, or at least close to one another. The algorithm will not recognize polysemous words either.

Despite those drawbacks, we think it can be employed as a very useful search tool. A user can enter a search term, and the algorithm shows the hierarchy for this word, after which the user can navigate from the search term to a relevant subconcept. In the spectrum from keyword search to hierarchy based browsing it is somewhere in the middle: it does not require a precisely worded information need as with keyword search, but the generated hierarchies do not support the user as much as with hierarchy based browsing. The hierarchy generated with SEQUITUR can be added, next to the manually created hierarchy, as another way of accessing the handbook, suited for readers who know better how to word the information they need.

Several hierarchies are generated for words that might occur low in the hierarchy. Therefore, the hierarchy editor can use the SEQUITUR algorithm to refine concepts at the leaves of the hierarchy. But, as lower level concepts often do not need a description, the editor might not necessarily have to be involved in the creation of those concepts, so the SEQUITUR algorithm might be used to add finer grained leaves of the hierarchy automatically and unsupervisedly.

4.5 Evaluation of the Hierarchy

Now that we have discussed methods that can help an hierarchy editor with the construction of the hierarchy, we will discuss in this section two different ways of evaluating such a hierarchy. The first, discussed in the next subsection, is an empirical evaluation by test persons who judge the hierarchical relationship between two concepts. The second, discussed in Section 4.5.2, is by calculating the value of certain metrics that give an indication of the structure of the hypertext.

4.5.1 Evaluating the Hierarchical Relationship

The hierarchical relationship can be evaluated in several ways. The most simple one is to present test persons with a number of generated relations between concepts and asking them whether one concept is more general than the other. However, when evaluating in this way, the purpose of the hierarchy, in our case navigation, is not taken into account. Therefore, concept hierarchies are better evaluated by giving the test persons assignments that involve the finding of relevant information. However, at the time of writing we were unable to find such evaluations in literature. A straightforward setup of experiments in this direction would be very similar to Chimera and Shneiderman's evaluation of interfaces, as discussed in Section 3.4.2. The only difference would be that different hierarchies are used, instead of different types of interfaces.

An example of the simpler type of evaluation was the evaluation performed by Sanderson and Croft (see Section 4.4.2 and [SC99]), who evaluated their automatically derived concept hierarchy. Eight test persons were used in the evaluation. They were presented two terms,[5] and had to decide whether there was an interesting relationship between the terms, or not. If the relationship was judged interesting, the persons had to choose one of 4 relationship types best indicating the kind of relationship: holonymic (part-of), hypernymic (is-a), synonymy and antonymy. They could also choose a fifth option, to indicate that they did not know the type of relation, or that it was another type of relationship than the listed four.

Besides terms which were related according to their algorithm, the test persons were also presented pairs of randomly selected terms. A comparison of

[5]Only *terms* in the hierarchy were compared, not *documents* belonging to concepts.

the judgements given by the test persons to the automatically derived relations and the random relations showed that their hierarchical relations satisfied their requirements on a concept hierarchy.

Although the latter method of evaluation is simple, it can be useful to get an idea of the quality of a(n automatically generated) hierarchy. However, a sound hierarchy (i.e. with the right hierarchical relationships) is not necessarily good for navigation. For example, hierarchies that have too much depth will probably not be very effective, since reaching some concepts might require too much traversals down the hierarchy.[6] In the next subsection we discuss metrics that can indicate those kinds of problems.

4.5.2 Metrics of Hypertext Structure

Besides algorithms that generate hierarchies with which human hierarchy builders can assess the contents of manually constructed hierarchies, tools monitoring the structure of the graph underlying the hierarchy can be useful for the constructor of the hierarchy. Simple tools for the calculation of the number of nodes and links, the average depth and the average branching factor of the hierarchy can help the editor to maintain an overview of the shape of the hierarchy. However, more advanced metrics have been investigated, such as the following two [BRS92].

Compactness. Compactness indicates how easy it is to reach nodes in the hypertext. At first sight, a high compactness would seem desirable, since it is good that every node can be reached easily. However, a fully connected hypertext is equivalent to a disconnected one where every node is accessible via an index and readers do not get any information about which nodes are useful to read next. Low compactness, on the other hand, indicates that the hypertext is poorly connected and that some nodes might be unreachable from other nodes. Therefore a hypertext should have moderate compactness.

Stratum. To get some insight in whether there is a reading order in the hypertext, one can calculate the stratum. A hypertext with a stratum of 1 is completely linear, while in a hypertext with a stratum of 0 structurally it makes no difference where one starts reading.

Stratum and compactness are dependent measures. For example, if the compactness of a hypertext is 1 (i.e. the hypertext graph is total), then its stratum is 0. They are not each other's inverse, so they are not redundant.

The above metrics can be very useful for the design of the navigational concept hierarchy and the electronic handbook itself, as they give at least some indication of the usefulness of the structure. Since the values are easy to calculate, it would be practical for the authors of the concept hierarchy to have them available. However, compactness and stratum do not take into account individual nodes. If the quality of the entire hypertext is low for some reason, as indicated by the value for compactness or stratum, it is useful to be able to point to the nodes causing the problem. Therefore, we also need metrics about individual nodes. Botafogo et al. present the following:

[6] This might be prevented by the relations other than the subtopic relationship, functioning as short-cuts through the subtopic hierarchy.

Depth. The depth of a node in the hypertext is just its distance from the root. It is unlikely that readers will arrive at nodes very deep in the hypertext (or nodes unreachable from the root). If nodes have such high depth, a link might be forgotten.

Imbalance. In general, it seems useful that a hypertext is a well balanced tree. Although imbalance does not necessarily mean that the hypertext is bad, it can be an indication of a poorly designed hypertext. Two imbalance measures are discussed: *depth imbalance* and *child imbalance*. The first indicates imbalance in the depth of the subtrees of a node; the second imbalance with respect to the number of nodes in the subtrees below a node.

For those metrics, it is assumed that the hypertext is in a tree structure instead of a general graph, so they are not directly applicable to our hierarchy with a DAG structure.

McEneaney [McE99] has continued research with hypertext metrics by defining metrics about the structure of *paths* followed through hypertext, that are predictive of the degree to which the reader has understood the text.
McEneaney has done experiments with 89 students, who had to answer as many questions as possible within 15 minutes, using an electronic student advising handbook. With the data from these experiments, he tries to discover which types of navigational paths through hypertext imply success on the search task.

He first visualized the paths that users have followed through hypertext. He visualized the *individual* paths of high scoring and low scoring subjects as a graph, and compared those paths. This comparison showed that low scoring subjects had mostly linear paths, resembling page turning in paper books. High scoring subjects had a more hierarchical pattern of movement, with the table of contents of the handbook as the top of the shallow hierarchy.

In order to formalize the observations from the visualizations, he wanted to verify whether two measures, *path compactness* and *path stratum*, correlate with user behavior in the experiment. The measures are similar to the compactness and stratum from Botafogo et al. explained above, but now applied to actual paths followed. He hypothesized that the *path compactness* correlates positively with the hypertext search scores, and that *path stratum* correlates negatively with those scores. Both hypotheses were confirmed after the calculation of Pearson correlation coefficients.

A feature of this work is the discovery of the relation between easy to calculate numerical metrics and actual performance by test persons. The metrics can for example be employed in an intelligent tutoring system on top of the handbook, that monitors user behavior and warns users when they are browsing in a linear way. The metrics can be calculated during reading, thus providing immediate information about the probability that a user is learning well about this hypertext, as opposed to testing a user with questions, which cannot be performed during reading without interfering with it.

To conclude, we think that metrics (both about hypertext structure and paths followed) can be very useful in the construction of hypertext, and in particular a concept hierarchy. When the relation between empirical data and metrics has been investigated further, such metrics are easily computable indicators of hypertext quality. Interesting questions for further research include how to cope with typed links in the calculation of the metrics, and with built in browser functionalities (such as ‘Back’ and ‘Forward’ buttons and bookmarks). Furthermore, it should be investigated whether there are new metrics that are even more strongly correlated with user performance.

4.6 Conclusion

In this Chapter we have discussed the building of a navigational concept hierarchy. First, we discussed the available material used in the construction of the hierarchy, and the advantages and shortcomings of this material for the purpose of constructing the concept hierarchy. In Section 4.3 we described the manual construction of the hierarchy at the ILLC, and the status of the Prolog prototype, which should be replaced by a more advanced one, accessible through the WWW.

Section 4.4 dealt with the use of algorithms for the construction of hierarchies. Since we require a very well structured concept hierarchy, we cannot use hierarchy construction algorithms without human supervision. However, the hierarchy builder can use the automatically generated hierarchies to verify correctness and completeness of his hierarchy, and the position of concepts in the hierarchy.

Finally, in Section 4.5 we discussed two methods for the evaluation of the hierarchy: by using test persons to assess the hierarchical relationship and by metrics about hypertext structure. By relating the metrics to empirical data, they can constitute a very useful evaluation tool.

Chapter 5

Linking the Hierarchy

5.1 Introduction

In this chapter we discuss the automatic generation of *external* links, i.e. links from the concept hierarchy to other sources. As discussed in Section 3.3, external links have as their target either the handbook text or information sources on the Web.

The main reason to automate the construction of such links is the effort needed to construct them manually. For example, when we only consider the handbook links, currently there are already more than 400 concepts that might be linked to a handbook comprising over 1000 pages. For a human link editor, it is too much work to check for every concept to which part of the handbook it should be linked. Furthermore, as both the text of the handbook and the concept hierarchy are dynamic, this link editing should be performed on a regular basis. Therefore, it is useful to automate this task. We have experimented using automated tools for link generation. This gave promising results, but as will be described later in this chapter, the results were not (yet) good enough to completely automate this task. Rather, such techniques can provide a lot of support to a human link editor and save him much work.

Besides saving work, another advantage of using automated tools is that the links generated will be more consistent, since there is one algorithm that always applies the same criteria to create a link, as opposed to several human linkers that have different criteria and are not perfectly consistent in applying those. Studies have shown that there is not much agreement in the hyperlinks created manually by different people, see [Gre97, Section 2.4, Chapter 3] for an overview of the literature and experimental results on this subject. This is not a surprising observation; the same inconsistency has been observed earlier for human indexers assigning index terms to documents (see, e.g. [FLGD87]), a task similar to inserting links. This inconsistency might make it difficult for readers to decide whether or not to follow the links.

Some people doubt whether the automatic generation of links really is useful. Thistlewaite [Thi97] states that if documents are often added and removed from a very large hypertext system, recalculating the links can be a large effort. And the generated links might not be useful enough, as compared to a search engine, to warrant the effort.

We disagree for two reasons. First, the re-computation of the links can take place in an overnight batch, causing no real practical problems. Second, generated links really are much more useful than a search engine. In the setting of an electronic handbook, users with a vague information need should be guided through the available material, something that links do and a search engine does not. User evaluation of automatically generated hyperlinks by Tebbutt also showed this [Teb99, p. 12].

With respect to hypertext, Thistlewaite [Thi95] has found 4 properties that it should *not* have, based on user studies. We list them below, and discuss their relevance to automatic link generation:

Inconsistent criteria for link generation. Automating link generation can avoid this, since one algorithm to generate links is more consistent than one or more humans.

Difficult to maintain. That is the main problem automatic link generation avoids.

Unsound links. There should only be links between passages that the user finds related. Therefore, an automatic link generator should create high quality links, which is not straightforward, as we will see in this chapter. When generating typed links, the passages should be related exactly in the way that the type indicates.

Incompleteness. Incompleteness occurs when the hypertext does not link passages that its readers would expect to be related. Avoiding incompleteness, however, can result in the hypertext having too many links, which confuses readers. In this case, the typing of links can reduce this confusion: although there are many links, they are at least divided in several classes.

The idea of generating links automatically is relatively new, and the techniques that have so far been used for this purpose include pattern-matching algorithms, text classification algorithms and techniques from the field of Information Retrieval (IR). Therefore, in the next Section we will briefly introduce text classification and IR. In Section 5.3 we will discuss several approaches to automatic link generation, after which, in Section 5.4, we will discuss experiments we performed ourselves to generate links from the concept hierarchy to the handbook using IR techniques. After that, in Section 5.5 the generation of links from the hierarchy to the web is addressed. Since we evaluated the links in our experiments in quite a simple manner, in Section 5.6 we will discuss some more advanced methods to do this. We finish this chapter with some discussion and concluding remarks.

5.2 Preliminaries

Since the link generation algorithms that we will discuss later in this chapter employ techniques from the field of Information Retrieval and text classification, we will shortly introduce both topics here.

5.2.1 Information Retrieval

The field of *Information Retrieval* deals with providing users the information they need by selecting documents from a database on the basis of a query that expresses the user's information need. Researchers have developed several different methodologies for this purpose. Of those, we will only briefly discuss the so-called Vector Space model since we have employed it for the generation of hyperlinks. For a general overview of IR, see the textbook by Baeza-Yates and Ribeiro-Neto [BYRN99].

In the Vector Space model, each query and document of a collection is represented by a vector of length t, where t is the number of distinct *terms* in the document collection. Those terms are usually all the words occurring in the document collection, minus so-called *stopwords*, words that occur frequently and are not informative about the subject the document is about (for example the articles 'the' and 'a'). Furthermore, the number of terms is often reduced by *stemming* words in the documents, i.e. by removing their prefixes and suffixes. For example, the words *connected*, *connection* and *connecting* are reduced to the same stem *connect*, thus allowing the recognition of words originating from the same root concept. Stemming and removal of stopwords are methods to decrease the number of terms, while trying to preserve the semantic content of the document.

The elements of the vector are the weights for each of the terms in that document or query. Many algorithms use a *term frequency-inverse document frequency* (*tf.idf*) weighting scheme, where the weights of a term are based on term frequency and inverse document frequency. When the *term frequency* of a certain term in a document is high, it is likely that this term is important for that document, and the weight should be high. On the other hand, when the *document frequency* of a term is high (the term occurs in many documents), the term is not useful to distinguish between documents and its weight should be low. Often, the weight vector is also *normalized*. The normalization prevents big documents, where because of this size many terms have a relatively high term frequency, to have weights that are significantly higher than those of smaller documents. Normalization is usually performed by dividing every individual weight by the entire weight vector's length.

The similarity between two documents, or between a query and a document, is the inner product of the two weight vectors. When both vectors are normalized, this will result in a similarity value between 0 and 1, which coincides with the cosine of the angle between the two vectors in t-dimensional space. To retrieve documents similar to a document or query, an IR system calculates the similarity between the query and all the documents and ranks the documents according to this similarity.

In IR, retrieval performance is measured in terms of *recall* and *precision*. Recall is the proportion of relevant documents in the collection that was retrieved. Precision is the proportion of retrieved documents that are relevant to the query. As most IR systems rank documents, but do not present one answer set, it is useful to evaluate the precision at different levels of recall. In general, at lower recall levels, systems have high precision, which decreases when recall increases.

IR can be used for link generation by using a concept as a query and linking from the concept to the best retrieved documents.

5.2.2 Classification

Classification or categorization is the task of determining of which class from a set of known classes a certain input is an instance. Much research on classification and in particular text categorization has been devoted to automatically generating or learning such classifiers from *labeled* examples; several machine learning techniques, such as neural networks, Naive Bayes and decision trees have been applied to the task (see [YL99] for a comparison of several methods). In text classification, a labeled example is a text document together with the class label of the class it belongs to. Having been induced on the basis of those examples, the classifier should be able to accurately predict the class of unseen texts.

Whereas IR techniques start with a query and end with a (set of) document(s), text classification techniques start with a document, and end up with a class assigned to the document. Text classification can be used to generate links by classifying a document as an belonging to one of the concepts in the concept hierarchy, and then making a link from this concept to the document. This method can be applied to documents both from the Handbook itself (e.g. chapters or sections) and documents on the WWW.

5.3 Approaches to Link Generation

In this section we will discuss several methods found in literature to generate links. The methods in the next three subsections employ classification techniques; the methods in section 5.3.4 and 5.3.5 use IR techniques.

5.3.1 Labrou & Finin

In [LF99], Labrou and Finin present experiments with the classification of documents into Yahoo!'s category hierarchy, which is very similar to classification into our navigational concept hierarchy. However, they did not classify with the purpose of creating links from the category to the documents, but to use the resulting classification as a descriptor of the document's contents, similar to terms in the subject classification schemes as used in libraries. The idea is that such universal descriptors can facilitate the work of search engines and other web-based services for information access.

The main contribution of this work is not the method for classification that they used (a classifier based on n-grams called *Telltale*), but experiments with variation in the representation of the documents and the classes. The documents were represented in three ways: by their title, their summary and their entire content. The representations for the Yahoo! categories were the following: the summaries of all the entries in the category, the summaries of the entries and their titles, and finally the summaries, titles and name of the category.[1]

Best classification performance was achieved by using its summary to represent a document, and the summaries and titles within the category to represent the category. As we will discuss in Section 5.4.4, this is somewhat conflicting with our experiments to generate links using IR techniques.

[1] A Yahoo! category is represented by a web page, showing, among others, the name of the category, and a number of *entries*: a title and a summary of a web site, with a hyperlink to this site.

The classification performance they achieved using these types of descriptions of documents and categories was quite high: 73% was correctly classified. Correctly classified here means that the test documents were classified in the same category they were taken from to generate the test. If documents are not correctly classified, this does not necessarily mean that they are bad classifications; some documents might be classified into one Yahoo! category but also fit reasonably well into another category. It is worthwhile to investigate this, since if the latter really is the case, Labrou and Finin's method might be used to create links unsupervisedly, due to the fact that it's performance would then be even better. However, concepts in our concept hierarchy differ considerably from the Yahoo! categories, so we do not know a priori whether it will work for us.

Labrou and Finin's conclusion is that currently their method cannot yet be used without human supervision; it can be used to suggest index terms to human indexers.

5.3.2 Vector Space Classification

Zhu et al. [ZGG+99] discuss an architecture for the organization of information on the Web based on distributed intelligent agents. An important component of their system is the Local Characterizing Agent (LCA) that indicates how much information about subjects in a subject hierarchy is present at a web site. For this purpose, the LCA classifies documents at the site into a category hierarchy for browsing the web (they do not mention the particular hierarchy used). The vector space classification they employ amounts to using a document as a query to a vector space IR system, which returns the best super-documents (i.e. categories) for the query. The most highly ranked category is considered as the category to which the document belongs.

The main difference with the approach discussed in the previous subsection is the representation of the documents and the category. The representation of a category in the hierarchy by Zhu et al. is a *super-document*, consisting of a concatenation of all the documents that fall under the category. Unlike with Labrou and Fining, no use is made of the summaries that always appear in such concept hierarchies. The documents to be classified are represented by their entire contents. Note that, according to Labrou and Finin this is not the best way to represent them: a summary is better for this purpose.

Zhu et al. compared their LCA to a Naive-Bayes classifier, and present results indicating their classifier to be significantly better. However, we do not think their classifier was performing very well. Using the number of documents to represent a category that gives the best accuracy (6), only 17% was correctly classified. The average distance[2] from a classified document to the right category was 4.3, which is quite high. They found their LCA to be accurate enough for an entirely automatic approach, but we do not think it is. Using it to create links from concepts to documents in the handbook, less than one out of five links would be good links.

[2]The distance between two categories in the hierarchy is the length of the path between them. E.g., the distance from a parent to its child is 1; the distance between siblings is 2.

5.3.3 Cora

McCallum et al. present several machine learning techniques to automate the construction of Internet portals in their very interesting article [MNRS00]. Using those techniques they have built a portal site for computer science research papers, called "Cora". Cora contains a topic hierarchy with 70 leaves to give access to the research papers on the Web. With the help of a spider they find relevant papers on the Web and add them automatically to the hierarchy, using a text classifier.

For text classification, they use an extended version of the Naive Bayes classifier, that does not require any labeled examples as input. Using the keywords of the papers and keywords assigned to the topics in the topic hierarchy (assigning keywords to topics is the only manually performed task), they generated preliminary labels for the papers. With the help of this information and the Expectation-Maximization bootstrapping process the Naive Bayes classifier has been learnt.

The experimental results show that the resulting classifier is quite good: 66% of the documents is classified into the right position in the hierarchy. Even if this method does not require any labeled documents, it outperforms Naive Bayes without bootstrapping that needs many labeled examples. Agreement on the test set by two human experts was 72%, so their classifier comes very close to human agreement.

5.3.4 Allan

Allan [All95, All97] has developed an algorithm to generate links using vector space IR. What makes his approach unique, is that he can generate several *types* of links, and automatically determine their type. Furthermore, he provides very nice graphical visualization techniques to show the relationship between different documents. His algorithm to type a link between two documents works as follows:

1. Decompose each document into parts. Allan divides the documents into paragraphs.
2. Calculate the similarity between every part of the first document and every part of the second document.
3. For each document pair which had a similarity higher than zero, break the documents into subparts, and calculate the similarity between every subpart. If the similarity between at least one pair of subparts exceeds a certain threshold, mark the pair of parts as 'good'. If not, mark them as 'tenuous'.
4. All the 'good' pairs that have a similarity exceeding another (higher) threshold are marked as 'strong', the others as 'weak'.
5. Merge links close to each other in order to simplify the connection graph between the documents. The merging takes places when the two links are close to each other in the documents. The distance between links is defined as the proportion of both documents that lies in between the links, and a threshold on the distance determines whether or not two links are close. Links resulting from this merging process are called *meta-links*.

6. Create and type a link on the basis of patterns that can be recognized within the remaining links.

The last step is based on some values on the links between the documents:

Convolution is a measure of how parallel the links are.

Expansion describes how much extraneous text was added in one endpoint of the meta-link and not in another.

Relative size indicates the proportion of the text at the start- and endpoint of a meta-link.

Absolute size describes the importance of the meta-link's endpoints to each other.

The last step of the algorithm determines the type of the link with a number of conditions on the above values. The types distinguished are the following: *aggregate*, *tangential*, *revision*, *summary/expansion*, *strong equivalence*, *equivalence*, *contrast*, *condensed treatment*, *comparison*.

The evaluation of the generated links has been quite poor. Eleven subjects were asked some questions about the relationship between the text at the source and destination of a generated link. The tentative conclusions are that the linked passages are usually distinct from the enclosing text, and the material in the linked sections is well focused. The similarity measure was a mediocre measure of the relatedness between two texts, which is somewhat disappointing.

Nevertheless, the work of Allan is very original in the sense that he is the only researcher (as far as we know at the time of writing) who, besides automatically generating links, also has attempted to give them types automatically.

5.3.5 Green

In traditional IR, polysemy can cause unrelated documents to be considered related, while synonymy can cause related documents not to be recognized as such. This is due to the fact that vectors of word stems are used, and the stems do not always represent the *concepts* discussed in the documents. In an attempt to avoid those problems, Green [Gre97] uses lexical chaining for the construction of links both within and between newspaper articles from the online version of the *Globe and Mail*.[3] He compares the links generated with the help of lexical chaining with the links generated by a vector space IR approach such as Allan's, discussed in the previous section.

Lexical chaining is a technique to find so-called *lexical chains*, sequences of semantically related words in a text. Most documents contain several lexical chains; every chain captures a portion of the cohesive structure of the document. When, for example, the words 'apple' and 'fruit' would appear in the same text, they should be in the same lexical chain because they are semantically related (via a subsumption relationship).

In order to determine the lexical chains present in a document, Green's lexical chainer uses WordNet [Fel98], and assesses the strength of the semantically relatedness based on the path in WordNet between the synsets containing the words, and the types of relationship used to go from the first to the second.

[3] `http://www.theglobeandmail.com`, visited on January 5, 2001.

Those strengths are used again to determine to which lexical chain a word belongs.

Lexical chains are used in two ways: for linking *within* and *between* articles.

Linking within articles. When all the words in the document have been assigned to at most one lexical chain,[4] the relative density of a chain in a paragraph is calculated, i.e. the fraction of words in a paragraph that belongs to this chain. The chain densities can be weighted in several ways, and the paragraphs can now be represented as vectors of chain weights, as opposed to vectors of term weights as used in standard vector space IR.

The similarity between the weight vectors can now be calculated, and for deciding whether or not to include a link, Green converts each similarity to a *z-score*, indicating how many standard deviations the similarity diverges from the mean similarity. A threshold on the z-score then determines which links are included. The z-score is a good measure for comparing similarities relative to each other, thus avoiding too many links when almost all similarities are high. However, Green does not let the system decide on the value of the threshold; the user is given some sort of sliding control between more links and fewer links.

Linking between articles. For the purpose of linking different articles, the lexical chains Green uses are defined more sharply. Words can now only be in the same chain if they are synonyms, antonyms, or connected via one is-a or part-of relation. Other paths are not allowed anymore, in contrast to the linking within articles. A problem with this scheme is that it is difficult and computationally expensive to compare different documents, since documents contain different lexical chains.

Therefore a simpler method is used, that is based upon the disambiguation performed by the lexical chaining. For every document, two vectors are constructed, each with an element for every synset in WordNet. The first vector (called *member vector*) stores the number of occurrences of words in the document belonging to the synsets, while the second (called *linked vector*) contains as elements the counts of synsets that are one (is-a, antonomy, or part-of) link away from words in the document. Then Green defines the similarity between two documents d_1 and d_2 as the sum of the inner products of both member vectors, the member vector of d_1 and the linked vector of d_2 and the member vector of d_2 and the linked vector of d_1.

Green has evaluated the links generated by a question-answering task and compared them with links generated by vector space IR (see Section 5.2.1 and 5.3.4). He could not show that his inter-article links were better, nor that intra-article links are useful.

5.3.6 Conclusion

In this section we have discussed several approaches to link generation, based on classification or IR techniques. A difference between the approaches based on classification and IR is that the accuracy of classification based techniques (Sections 5.3.1 – 5.3.3) has been computed, but this has not been done for the IR techniques (Sections 5.3.4 and 5.3.5). Both Labrou and Finin's and

[4] Some words cannot be assigned to a lexical chain since they do not occur in WordNet.

Cora's classifiers achieved high accuracy (about 70%). The idea of Labrou and Finin to experiment with different representations for the documents and the classes (summary, full text, summary of the entries in the category, etc.) is very interesting. In the next section we will discuss our experiments with linking concepts to the handbook, where we also varied the representation of the concept by assigning different weights to the name of the concept.

Despite the proven success of the classification methods, we have decided to use IR techniques. The reason for this is that the number of links from the concepts to the handbook should be under our control. By classifying passages from the handbook into the hierarchy we lose this control and some concepts might get too many links and others too few. By using IR techniques with the concept as a query and the passages from the handbook as the documents to be retrieved, we can take the top n passages for every query and create links to them.

Another advantage is that Allan showed us that automatic link typing is possible with IR techniques. Since Green showed us that a more advanced technique than vector space IR does not give us much advantage, we have used vector space IR for our exploratory experiments. In the next section we discuss those experiments.

5.4 Our Experiments: Linking Concepts to Pages

We have experimented with the generation of links from concepts in the hierarchy to pages in the handbook dealing with those concepts. In the experiment we used IR according to the vector space model, as discussed in Section 5.2.1. Christof Monz (ILLC, University of Amsterdam) kindly allowed us to use his RETRIEVE system for this purpose. In this section we describe the experiments: first the queries and documents we used, after that the similarity measure, the way of evaluating, the results and finally we present our conclusions.

5.4.1 Queries & Documents

We used the concepts in the hierarchy as queries to the IR system. We hypothesized that the name of the concept would be more important than the descriptive text, so we decomposed the query vector c (concept) in two sub-vectors, c^n (concept name) and c^d (concept description), permitting the assignment of different weights to both.[5]

As the documents to be retrieved, we used chunks of text in the handbook, that corresponded to pages in the paper version. Those chunks are all about 350 words in length. This is a somewhat arbitrary decision, but the diversity of writing styles of the authors contributing to the handbook made it impossible to use the more obvious sections or subsections. An advantage of using sections or subsections is that if the chapter is well written, they will deal with a clearly identified topic and are therefore a very useful unit for a reader to jump to. However, a disadvantage that we found decisive, is that authors contributing the handbook have divided their chapters in too diverse a way in sections and subsections. Some chapters contain many small sections, while others consist of

[5]Notation: c_i^n (c_i^d) denotes the i-th weight in the vector corresponding the the concept name (concept description); p_i is the i-th weight in the weight vector for the Handbook page.

fewer but larger sections. This would make the size of the destination of a link too unpredictable for a reader to be useful.

Furthermore, although pages are not the most useful unit for readers, they seem to be very suitable for IR systems. Kaszkiel and Zobel [KZ97] present experiments giving evidence that retrieval precision when using pages of about 300 words is higher than when using sections.

5.4.2 Weighting & Similarity

For weighting the query and document vectors we did some preliminary experiments, which suggested us the use of Salton and Buckley's tfx.nfc weighting scheme, giving the following weights to the terms in the query and the documents:[6]

$$c_i = \frac{\left(0.5 + 0.5 \times \frac{f_i}{\max f}\right) \times \log \frac{N}{n_i}}{\sqrt{\sum_j c_j^2}} \quad (\mathit{Query}) \tag{5.1}$$

$$p_i = f_i \times \log \frac{N}{n_i} \quad (\mathit{Document}) \tag{5.2}$$

In the formulas, c_i is the weight of the i-th term in the concept vector, f_i is the frequency of the i-th term in the concept or page, $\max f$ is the maximum term frequency occurring in the vector, N is the total number of pages in the collection, and n_i is the number of pages with term i. It is unusual and contrary to Salton and Buckley's findings that the document vectors in the above formula are not normalized, while the query vectors are. The reason for this is that in our setting the queries, the concept descriptions, do vary in length, and thus need normalization, but the pages are of about equal size. In most retrieval situations, this is the other way around.

It can be argued that the query normalization is superfluous, since it amounts to multiplying every similarity value for the current query with a constant factor. This means that the ranking for the query at hand remains unaffected. This is true in most applications for information retrieval, where one wants to find the documents most similar to a query. However, using IR for the generation of links, we are also trying to find a general threshold on the similarity (and perhaps the ranking) in order to decide whether to include a link. Therefore, we must be able to compare different queries to each other and do need the normalization.

Using the above weights, we calculated the similarity between a concept c and a page p as follows:

$$\mathsf{sim}(c, p) = \alpha \times (c^n \cdot p) + c^d \cdot p \tag{5.3}$$

In this formula, α is the weight assigned to the name of the concept, relative to the weight of terms in the description. In the experiments we used values of 1, 3 and 5. We also performed an experiment in which we used a similarity $\mathsf{sim}(c, p) = c^n \cdot p$, using only the concept name.

5.4.3 Way of Evaluating

For the evaluation of the generated links we were unable to calculate the *recall* (see Section 5.2.1, since this would require to manually assess which pages of

[6] We use c_i to refer to both c_i^n and c_i^d, since they are weighted in the same way.

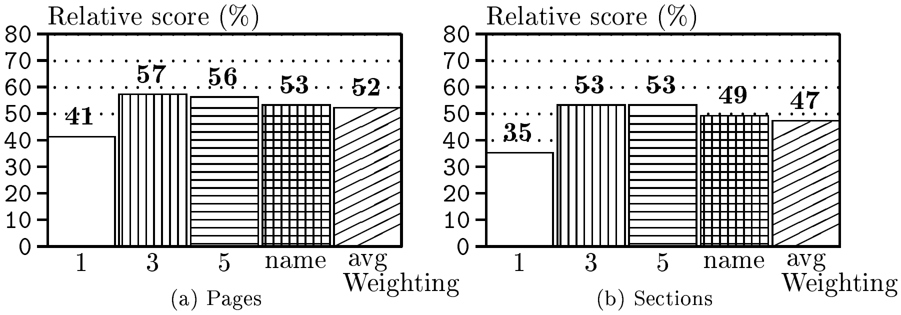

Figure 5.1: Average relative scores

the handbook are relevant to which concept, a very labor intensive job. If we would have performed this assessment manually, generating links would not be needed anymore.

Therefore, we evaluated only using a form of precision. The difference with standard precision is that links were not rated on the usual binary scale of relevant and irrelevant, but on a ternary scale, with the additional category 'somewhat relevant'. For about 30 concepts, the precision within the 5 top ranked retrieved pages were evaluated. Besides assessing whether the retrieved *page* was relevant to the query concept, we also assessed whether the smallest *section* or *subsection* to which the page belongs was relevant.

This method of evaluating is very simple and borrowed directly from the field of information retrieval. However, the evaluation of hyperlinks can be performed differently, by taking into account the overall hypertext structure. For example, automatically generated links can be compared to manual ones by having users perform information seeking tasks using both the automatically generated hypertext and the manually generated one, and comparing the users' performance with the different hypertexts. In Section 5.6 we discuss the evaluation of links in more detail.

5.4.4 Results

Let us first consider the results shown in Figure 5.1 on page 56. In the charts, the horizontal axis indicates the query used: α values of 1, 3 or 5, only using the concept's name, and the average over the four methods. The vertical axis gives the relative score: the ratio of the actual score and the total score that could be achieved. The charts show the *average* relative score of all the documents in the top 5 ranked documents.

It turned out that using a value of 3 or 5 for α (i.e. assigning 3 or 5 times more weight to the name of the concept than the descriptive text) gave the best results. Evaluating the relevance of the page at hand, it resulted in an average score of respectively 57% and 56%. For the sections, both values of α gave 53% scores.

Comparing Figures 5.1 (a) and (b) one can see that the scores for the retrieved *pages* are higher than the *sections* surrounding those pages. On average, the score of the pages is 5% higher.

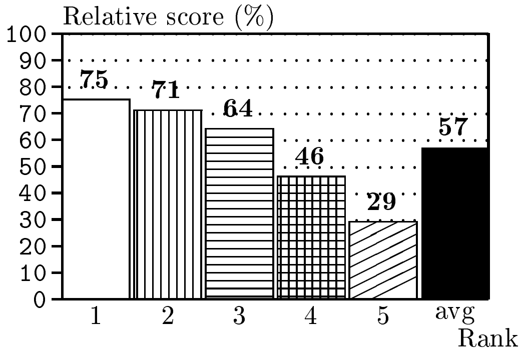

Figure 5.2: Relative scores per rank (3 × name), pages

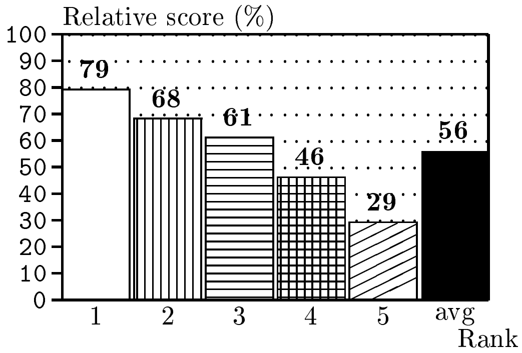

Figure 5.3: Relative scores per rank (5 × name), pages

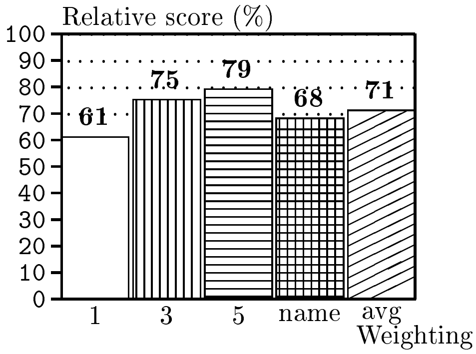

Figure 5.4: Relative scores top ranked documents (pages)

Figures 5.2 and 5.3 show the relative score per rank within the top 5 for the experiments with α-values of 3 and 5. Not surprisingly, this score decreases for lower ranked documents, which means that the vector-space similarity value coincides with our human opinions about similarity. The score for $\alpha = 5$ decreases more quickly, but it also starts higher.

The 79% score in Figure 5.3 means that the documents with the highest similarity were rated on average 1.6 on our ternary scale from 0 to 2. So automatically linking from the concept to the section with the highest similarity can result in a large proportion of useful links. However, we do want more than one link from the concept to the handbook. Suppose we want 5 such links, we will get links with an average rating of about 1.15 using either method, which is too low. At present, therefore, we cannot use this method without human supervision.

There were large differences in the scores for different queries. Using $\alpha = 3$, for example, the two 'worst' queries received an average score (for pages) of 20%, whereas the 'best' three queries got a score of 90%. Inspecting the bad queries, it seems that shorter concept descriptions will more likely generate worse links, but there are exceptions. Therefore, we cannot derive a general rule to avoid the use of shorter concept descriptions.

Inspecting the actual pages retrieved, there is only 9% difference between the $\alpha = 3$ and $\alpha = 5$ queries. However, this difference has such minor effects on the average score, since of the different pages, 83% was on the fifth rank, and 17% on the fourth in both the $\alpha = 3$ and $\alpha = 5$ ranking, therefore unlikely to have a high score.

This might cause one to think that only the lower ranked documents will differ across the different types of query used, and the top ranked documents are the same. This is not the case, however. Figure 5.4 shows the relative score for the most similar page retrieved using the different queries.

5.4.5 Conclusions

From the experimental results, we can draw two conclusions: first, IR techniques can support a human link editor, but not yet replace him. Second, concept names are very important. Both conclusions are discussed below.

IR can support humans with link generation, not replace them. It seems promising to use IR techniques for the construction of editor's tools for link generation. At present, the need for high quality links, as required by the navigation function of the concept hierarchy, prevents us from having an autonomous, unsupervised link generator; a human link editor should still make the decision whether or not a link will be included. For links from the hierarchy to the web discussed in Section 5.5 we do not demand the same high quality.

Inspecting the results of the experiments, it is not clear how to automate the decision whether or not include a link. Both the rank of a retrieved document and the similarity measure can provide information about this, but inspecting those values we were unable to find thresholds separating the bad from the good links. Some documents received relatively low similarity values but were still relevant, and vice versa. The same problem appeared with the ranking. But Figures 5.2 and 5.3 show that there is at least some correlation between the rank and the relevancy.

Maybe the application of machine learning algorithms can provide us with a classifier that can make the right decision. Also, more advanced functions on the similarity, like the z-score that Green uses (see Section 5.3.5), might provide a solution. The advantage of the z-score is that it is about relative similarity values, instead of absolute values. But using a z-score is only possible when the similarities have a normal distribution. Green found that this was the case for the similarities between paragraphs, but not for similarities between different articles. Unfortunately, Green does not provide a means of determining the threshold automatically and leaves the its setting to the reader of the hypertext. Experimenting with several kinds of algorithms should answer the question whether it is possible to distinguish between the useful and useless links on the basis of only the similarity measure and the ranking.

The metrics for hypertext structure that we discussed in Section 4.5.2 can also be taken into account when making the decision whether or not to include a link. For example, links that do not score high according to the similarity or ranking might still be included because the value of the metrics indicate that their inclusion would improve the general hypertext topology.

The importance of the concept name. The second conclusion is that the above experiments show that the name of a concept is indeed important, and assigning more weight to it can improve performance. Best performance was achieved when α (Equation 5.3) was assigned the value 3 or 5. Using only the name of the concept also gave quite good precision, while using the standard weighting scheme where the terms of the description and the name were equally weighted gave much worse precision. The relatively high precision using only the concept name as a query offers opportunities when links should be generated real-time. Concept names are short compared to concept descriptions, so their weight vectors can be computed quickly.

A reason for this importance of the name might be that it is a key term, so if there are pages that contain the name, they should be retrieved. This brings us to a question for future research. For example, would phrase-based IR improve precision? In the Vector Space model, we represent the documents as a 'bag of words', i.e. we only represent them by the number of occurrences of each term in the document, and do not take into account sequential information. However, many concept names are multi-word terms (e.g. 'modal logic', 'speech acts', 'lambda calculus'), for which the order of the words is important. Phrase-based IR does not discard sequentiality in the representation of the documents, and might therefore improve the retrieval results.

The finding that the concept name is important is more or less contrary to Labrou and Finin's observation (see Section 5.3.1) that a category is best described by the entry summaries and the entry titles, without the *category name* (which is comparable to our concept name). It is interesting to know the reason for this. It might be the type of concepts used; we have a concept name with a description explaining the concepts, while Yahoo! has a concept name (in their vocabulary: category name) and a list of entries: titles with a short summary of the contents. Another reason can be the algorithms used: we use IR techniques, whereas they use text classification techniques. This is something that deserves further investigation.

We might use more of the concepts for retrieval than just the name and the description in order to enhance precision, since the concept hierarchy offers much more information than this. Although this information is meant to be used by humans, it can of course be used by IR algorithms as well. However, it is important to have such information explicitly available as such; only if we know that certain words make up a title, we can weight this title differently to improve retrieval. Therefore, we should not store the hierarchy as a collection of HTML or LaTeX files, since those are markup languages that ignore semantic content. Instead, we should use storage formats like XML or SGML [vH94], that do allow for the explicit storage of semantic information.

5.4.6 Next Steps: Exploiting the Hierarchy

Since the experiments discussed in this section were exploratory, we used quite basic techniques for the purpose. However, since we generate links from a *concept hierarchy* to a handbook, we can profit from the hierarchical structure that is present. In this section we discuss two such alternatives for the process of generating links, that are worthwhile to explore next.

Hierarchical Similarity. In the experiments, we only used the concept names and descriptions, and did not make use of their position in the hierarchy. However, we feel that this information might help retrieval.

One way of doing this is using another similarity measure. Hierarchy-based similarity uses, besides the concepts themselves, also the similarity between subconcepts and the documents. The similarity function between a concept c and a document d can then be defined as follows:

$$\mathsf{sim}(c,d) = c \cdot d + \frac{\beta}{n} \times \sum_{i=1}^{n} s_i \cdot d \tag{5.4}$$

Here, s_i is the i-th child of c, and n is the number of children. The parameter β determines the weight one wants to assign to the children; the division by n normalizes over the number of children. This is an interesting alternative that deserves further exploration.

Retrieval by Sorting in the Hierarchy. Another way to make use of the concept hierarchy is the following. We can use a certain concept to find documents related to subtopics of this concept, i.e. concepts lower in the hierarchy. We can first retrieve a large set of documents for the concept, and then sorting those documents down to the leaves of the hierarchy. Such an algorithm can be recursive. It should be evaluated whether a document that is retrieved for a concept fits at the current concept in the hierarchy, or better fits at one of its children. If it fits here, a link must be created from this concept; if not, for the best child the same evaluation should start. This process can continue until the document arrives at a leaf concept. Sorting in this manner requires a measure to evaluate where a document fits best. Here, standard vector space similarity might be used, or another form of hierarchy-based similarity as described above.

5.5 Linking to the Web

In the previous section, we described experiments with (automatically) linking the concept hierarchy to the Handbook. We also want to have useful links to the Web, since there is a lot of material on the handbook's subject available on the Web. As discussed in Section 3.3.2, we distinguish between four types of such links: links to research groups, individual homepages, tools and publications. In the next subsection we discuss how to make links to the web, and for which type of links this process can be automated. After that, we discuss the automatic generation of programs used in the process.

5.5.1 The Entire Process

On the Web, there are many search engines that help people find the information they need. To avoid reinventing the wheel, we can use those to link the handbook to relevant material on the Web. For example, to create links to homepages, we may use the name of an important researcher in the Logic & Language field as a query to one or more search engines,[7] and use one of the returned results as a link to the home page.

There are several reasons why it is useful to have the links to the Web created automatically:

- The web is dynamic: new pages often appear, pages often move from one location to another, or disappear. Therefore, links to the web should be maintained continuously.
- Having a concept hierarchy of a substantial size has as a result that this maintenance should be done for many links.
- It is boring to check for new links and verify whether existing ones still exist. Still, it is important since readers want to have recent information and do not like to encounter broken links.

Therefore we want to create those links automatically. Here we discuss a method to do this. It will take four steps: first, a query will have to be formulated. Second, the query needs to be posed to the search engine(s) suited to the query. The third step consists of the extraction of the relevant information from the HTML page with the search results. The computer programs that perform the second and third step are called *wrappers*, and are specific for the search engines used. The last step is the processing of the extracted results. We will discuss the steps in the next paragraphs.

Query Formulation. To be able to use a search engine for the generation of links, one needs a query to pose to it. One can ask authors contributing to the concept hierarchy to provide such a query (without it being shown to the user). But it is be easier for the authors if the query is derived automatically from the description of the concept. For each type of link, we shortly discuss to what extent we think this is possible:

[7] We can use both general purpose search engines like Google (`http://www.google.com`, visited on January 5, 2001) and domain specific search engines like the home page search engine Homepageseek (`http://www.homepageseek.com`, visited on January 5, 2001).

Links to individual home pages. Only homepages of people mentioned explicitly in the description can be searched. Probably the only people mentioned in the description are the authors of referenced papers. The queries can be submitted to a specialized homepage searcher, like the mentioned Homepageseek.

Links to publications. Links to publications can be generated automatically, from the references that are mentioned in the description text of the concept. One can use the search engines of online publisher's sites to find the desired publication.

For both the links to individual homepages and the links to publications, the names should be recognized as such. This can be done with simple clues: e.g. names have a capital first letter, names of persons are mentioned first in a reference, etc.

Links to research groups. It is more difficult to generate a query for a research group from the description of the concept. In the current glossary of the Handbook, research groups are not mentioned at all. Therefore, the name of the research group should be provided by the author of the concept. There are no special purpose search engines for research groups, so the queries will have to be submitted to general purpose search engines.

Links to tools. To generate a query for a link to a tool is also difficult, since tools are not recognizable by their names or by mark-up properties. Authors should in this case also provide the query.

To conclude, the automatic formulation of a query seems to be possible for the links to publications and to homepages of researchers, but not for links to research groups and tools. Whether the query was obtained automatically or elicited from the author, it should posed to the search engine, which we discuss next.

Submitting the Query. The search engines should be addressed via their URL and the query must be translated from the internal format of the computer program to the format of the specific search engine. This task is performed, together with the information extraction from the HTML result page (discussed in the next paragraph) by so-called *wrappers*. Wrappers handle the source specific aspects of computer programs dealing with different types of information sources; every connected information source has its own wrapper.

Query translation is necessary since search engines have their own query syntax; e.g. for some search engines, the query must be enclosed between double quotes in order to do a phrase search, while for others a certain checkbox must be checked. In general, the query posing part of wrappers is made by hand. Most search engines have an interface with some checkboxes and a textfield where the query can be posed. Then, when the query is submitted to the search engine, it is converted to a URL that consists of the location of the script that executes the query, and arguments to this script. The person building the wrapper studies the resulting URLs and builds a program that can convert the query in the internal format directly to the before mentioned URL.

Information Extraction. As mentioned above, search engines return their search results in an HTML page for display by a Web browser. However, an HTML page does not give enough structure to the information to be used directly by a computer program. Therefore wrappers discard the HTML on the result page and return the actual search results in a structured format, e.g. a set of tuples of attribute/value pairs. The information extraction can sometimes be performed by simple algorithms, that only look at superficial properties of the page (e.g. 'after the third `<B>` tag there is an `<A>` tag containing a URL to be extracted'). More advanced algorithms define the wrapper in the framework of a certain class of grammars, while some researchers have also conceived wrappers that use natural language processing techniques, and are powerful enough to handle irregular sources.

To extract information from *result pages* of search engines, which generally have some kind of tabular format, natural language processing techniques are not necessary, and we can rely on the simpler types of wrapper. In the next subsection we describe an approach to generate such wrappers automatically. We will fist discuss what to do with the extracted search results.

Processing of the Results. The processing of the search results is very important. High recall and especially high precision is very desirable, as low precision makes the readers lose confidence in the electronic handbook's links. One way to get higher precision and recall than individual search engines is to combine the results of several different search engines.

This combination, or fusion, of information from different sources is not straightforward. For example, what to do if search engine A ranks document d high, and B ranks it low, but it is known that B gives better results in general? Overall retrieval effectiveness can be severely degraded if the responses from different sources are simply ordered by the rankings reported from the individual sources [Voo97]. Reason for this is that different sources use different ranking mechanisms that are not easily comparable.

To the processing of the results also belongs checking if the found URL is of the right type; it would be a bit silly if a link was typed as a link to a homepage, whereas it really is a link to a publication. This can be done by simple means, such as recognizing some keywords on the page. On homepages, words like 'personal', 'cv', 'publications', 'teaching' will occur, while pages of research groups will more often contain terms like 'institute', 'group', 'members', etc.

When the results have been processed, the link to the found page can be added to the concept. Depending on the time it takes to perform the entire process, it can be done either when someone wants to view the concept and its description, or in batch.

5.5.2 Wrapper Induction

As we promised above, we would go into more detail about automatically creating the information extraction part of the wrappers.

To automate the construction of those information extractors is very useful, since it is difficult to program them manually. It requires extensive analysis of result pages of search engines and takes a lot of time. It must also be done often, since search engines on the web frequently change the way in which they present the results, which usually has as a consequence that a new wrapper must be made.

Learning Wrappers. We have devised a method for learning wrappers that requires very little user input. It operates incrementally, starting from a *single* labeled item on a *single* result page. Our method uses an abstract grammar to represent the items encountered. The grammar is initially set to the labeled item. Next, the grammar is used to find a next item; to this end, an adaptation of the *string edit distance* is used. Every time a new item is encountered, the grammar is adjusted to cover this item as well. The process stops when no further items can be found.

It is a complex form of grammar induction, since the grammar should be induced from only *positive* examples (the items on the page). Furthermore, when inducing the grammar, the examples on which to induce must be found. The latter is done by our algorithm by making some assumptions about the structure of HTML result pages.

Experimental Results. We have tested our wrapper learning algorithm on the result pages of 22 search engines on the Web. For 16 of the 22 retrieval engines, the algorithm created a working wrapper; for another 2 the algorithm can be made to work with minor modifications. Allowing these modifications, we have a success rate of 82%, based on labeling only one item on only one result page. The average time needed to learn a working wrapper was only 7.8 seconds on a very modest machine.

Discussion. Our approach for automatically generating wrappers is very successful in the sense that its performance is good compared to other methods, while the user input is very limited. In recent years many researchers have investigated automatic wrapper generation. Most other approaches in the literature require substantially more user input. The only real competitor to our method is [HD98]; while also requiring a limited amount of user input, their method seems to handle differences in the order of relevant information better than ours. But since all the methods for wrapper generation have been tested on different data (see Section 5.6.3), further experimental evaluations and comparisons are needed.

The paper [CRdR00] provides references to the literature and presents our approach for generating the information extraction part of wrappers in more detail.

5.6 How to Evaluate

In Section 5.4 we discussed our experiments with link generation from the concept hierarchy to the handbook. The evaluation we performed was quite simple, but justified by the fact that the experiments were exploratory. In the next subsection, we first discuss alternatives for evaluation in an exploratory stage, to determine a weighting scheme. However, when more has been explored, it certainly makes sense to have a more thorough evaluation. Therefore, we also discuss two methods, due to Green and to Tebbutt to evaluate similarity links. Finally, in Section 5.6.3 we discuss the issue of evaluating wrappers.

5.6.1 Determining the Weighting Scheme.

For IR algorithms that find documents relevant to a query there are good test collections (e.g. the one used at the TREC conferences). Hence, one can easily find the best weighting scheme for the algorithm. However, since evaluation of hypertexts takes a lot of effort, one has to find an easier way to choose the right weighting scheme when IR is used to generate hypertexts. There are at least two possible solutions to this problem:

Ideal hypertext collection. A test collection with a number of optimal hypertexts based on a number of documents can be used to compare the weighting schemes. In that case, there should be a similarity (or distance) measure for the graphs underlying the hypertexts to be able to calculate the similarity between the optimal and the generated hypertext.

Clustering hypertexts. A way to decrease the number of hypertexts to be evaluated is the clustering of hypertexts. The hypertexts generated using the different weighting schemes can be clustered, so that only a representative hypertext from every cluster should be evaluated. Like the previous solution, this requires a similarity or distance measure between hypertexts.

5.6.2 Evaluating Similarity Links

In this subsection we will discuss two different methods to evaluate generated similarity links: one is due to Green, the other to Tebbutt.

Green. The similarity links can be evaluated in similar ways as the hierarchical relationship. It ranges from having users assess whether two documents that are linked by the link generation system are really about the same thing, to having users perform a real IR task.

Green tries to compare his link generation system based on lexical chains (see [Gre97] and Section 5.3.5) with a vector space IR system such as Allan's (see [All95] and Section 5.3.4). For practical reasons he does not use the SMART system used by Allan, but the system Managing Gigabytes (MG) [WMB94]. He generates links with MG using a method very similar to Allan's.

The task that is used for evaluation is question-answering, since it is well suited for the browsing methodology that links should support, and because it can be performed using *only* browsing. Although many hypertext applications include a search interface, those are not used in the experiments, since they introduce the problem of whether to attribute performance in the task to searching capabilities of the test persons or to the quality of the links.

An obvious way to compare both systems would be to generate two different hypertexts, and compare the test person's performance using both. However, this would require too many test persons. Green solves this problem by generating one hypertext containing links from both systems and counting the number of times the links from each system are followed. The links between newspaper articles are shown with the headline of the article that is the link's target. Although Green does not mention this, it is thus implicitly assumed that the headlines adequately represent the content of the articles. However, this can be questioned, and the user's decision of whether or not to follow a link may well depend on the attractiveness of the headline, instead of the system that generated the link. Furthermore, the order in which the links are shown may also influence users. Green showed that the links occurring first were chosen most often, but does not mention whether he equally spreaded the links generated by both systems over the pages with links.

This way of evaluating also introduces the question what to do with links generated by *both* systems. Green expects both systems to have many links in common, and simply excludes those links from the evaluation hypertext since 'By leaving them out, we test the differences between the methods rather than their similarities.' [Gre97, page 68]. This resulted in many useful links not being included, and small numbers of correct answers found by the subjects.

We think it is rather strange that Green expects the systems to have many links in common, because if Green already expected this, why would he bother to design a link generation system that has worse computational properties than vector space IR? Furthermore, if the systems would have this large amount of links in common, it is not unreasonable to expect the other links that were not common to both methods to be of the same quality.

Instead of leaving the common links out, Green could also have solved this problem by including them in the hypertext, and when a user followed such a link increment both counts, or not counting it at all. Then, the subjects would be able to answer more questions.

The main conclusion, finally, is that inter-article hyperlinks generated using lexical chains are not significantly better than links generated with MG. The intra-article links did not prove to be useful either. This conclusion is a conclusion that is often drawn when comparing several different IR systems (or even artificial intelligence systems): systems that make use of more knowledge about language structure and the meaning of words, do not, in general, show better performance. The reason for this is unclear, and the phenomenon is worth to be investigated by the IR community. What it tells us, is that it is good practice when trying to solve a problem to start with a simple solution, because there is a good chance that it will work.

Although the main conclusion is somewhat disappointing, the treatment of the collected data deserves praise. He uses statistical techniques such as standard regression, ANOVA analysis of the regression, and t-tests to establish the conclusion. He carefully selected dependent and independent variables to test their hypotheses and tried some combinations of those. Both the overall results and the results per question were analyzed.

Tebbutt. Tebbutt [Teb99] presents yet another way to evaluate similarity links generated by an IR system. The links were created in a large procedural manual (100 MB, about 41000 pages) of the Social Security Administration using an IR tool. The hypertext version of the manual consists of chunks of text corresponding with sections of the original manual, since the section is the principal content bearing unit of the manual. Using a section as a query and the other sections as documents to be retrieved, a link is created from a section to the 5 most similar sections. To every link generated, a *percentage similarity measure* is assigned, to give an estimate of the similarity between the source and the target of the link. First access to the manual is not given by a table of contents, but by a search engine (the same one that was used to create the links).

The evaluation was small scale, with only 6 users experienced in using the manual with a search engine, but without links. The task they had to perform was to gather information about items or tasks they encounter in their daily work with the manual. They were asked to describe and discuss their actions during the information gathering. The researcher was there to observe the participants and supply technical assistance with the system. After this, the participants were asked answer a number of questions about the system. All this was recorded on an audio tape.

Several interesting observations came about from the evaluation. With respect to the interface, the subjects found it more useful to have the similarity links at the top of every section. This way, when the section is not appropriate they can easily leave it. The percentage similarity measure was found rather confusing than useful; the ranking of the links alone provided enough information. The participants did not agree on the accuracy of the links. The number of five links per section was found rather small, and participants suggest not to have a fixed number of links, but a varying number based on the relevance, with a minimum of about five. Subjects also missed structural links and referential links: cross-references in the text were not supported by corresponding hyperlinks.

An evaluation like this is very useful, since it is not numerical, but tells us a lot about what users experience as positive or negative when using a hypertext with automatically generated links. Especially when trying new ideas, feedback like this is almost indispensable. For example, Tebbutt did not seem to have the idea to have the number of links depend on the similarity measure before the evaluation. However, a problem already mentioned in Section 3.4.2 is that satisfaction of hypertext users does not always correspond with their performance [Blu99], so one must be cautious with evaluations like this.

5.6.3 Evaluating Wrappers

In Section 5.5.2 we have concisely described our method to automatically generate wrappers, and discussed experimental results. We stated there that our method works for 16 of 22 search engines with which we tested the wrapper induction algorithm. For a wrapper to work, it means that it extracts the right attribute value for every attribute on the page. Even if only one attribute value is extracted incorrectly, a wrapper is said not to work. This may seem a bit rigid, but the purpose of wrappers is to mould unstructured data into a structured form that can be processed by computer programs, and errors in the structured form can cause the computer programs to break down, rendering the wrapper completely useless. For example, if a string instead of a numerical value is extracted from an HTML page, the computer program might want to perform something impossible like arithmetic on strings.

Although evaluation is as simple as checking whether a wrapper correctly extracts all attribute values from a page, there are some interesting issues about this subject:

Number of test pages. To evaluate the wrappers generated by our algorithm, we checked them with about a dozen of result pages from the search engine. By inspecting the generated wrapper, the type of result pages, and the extracted information we could decide whether the wrapper worked or not. This decision is based on the assumption that the variation between different result pages of a certain search engine is not unexpectedly high, and this assumption is fulfilled because the search engines generate their result pages automatically.

However, it is formally incorrect to decide that a wrapper works by just inspecting the wrapper and result pages of search engines, since we do not know anything about result pages that will be generated in future. But in a practical setting, this will do.

When to generate a new wrapper. When the wrapper seems to be correct on the basis of the evaluation on a number of result pages it is put in use and will be used until it does not work anymore. It does not work anymore when the structure of the HTML page for which it has been constructed changes in such a degree that the wrapper is unable to extract the right attributes. Whether it works or not, can be verified automatically by posing standard queries to search engines, extracting the attributes and comparing them to stored extracted attributes. However, the extracted attributes differ from the stored ones not only if the structure of the page has changed (which we want to detect) but also when the search engine returns other answers to the query than before. The latter does not mean that the wrapper does not work, so when differences are detected a human should always check whether the wrapper really works incorrectly.

However, the wrappers we generate do not allow for many changes in the structure of the result page, so most of them break down when the structure of the page changes, something that will be detected when posing the standard queries.

Standard test set. While the field of IR has the TREC conferences[8] with standard queries and document collections, unfortunately there is no such thing for the (newer and much smaller) field of wrapper generation. This makes a thorough comparison of the different approaches to wrapper generation impossible. If there were such a standard test set, one would be better able to find flaws in wrapper induction algorithms and find types of pages for which it is difficult to generate wrappers.

In literature all wrapper generation algorithms are evaluated by taking a number of sources and verifying whether the wrapper generation algorithm works for them; we were unable to find more sophisticated approaches.

5.7 Discussion and Conclusion

In this chapter we have discussed the automatic generation of links. We have started the chapter by explaining the usefulness of link generation, and introducing briefly two techniques used by methods for link generation: IR and classification. After that, we discussed several methods found in literature to generate links, some of which rely on classification techniques, others on IR techniques. After that we presented experiments we performed ourselves to link

[8] `http://trec.nist.gov/`, visited on January 5, 2001.

concepts in the hierarchy to handbook pages. The main conclusion of the experiments is that we think it is useful to support an editor with IR tools for the creation of links, but not to automate the task entirely. Assigning more weight to the name of a concept than to the descriptive text favored retrieval performance.

We also discussed the creation of links from the concept hierarchy to the WWW, and briefly explained the automatic generation of wrappers that can be used for this purpose. After that we discussed the issue of evaluating links, and evaluating the quality of wrappers.

We conclude this chapter with two challenging issues that we have not discussed so far.

Nonmonotonicity. A problem with using Information Retrieval techniques to generate semantic links is nonmonotonicity [Thi97]. When documents are added to the electronic handbook, the IR techniques will create links to them. However, this can cause previously generated links to disappear; if documents D and E were linked because E was in the top n of retrieved documents on query D, now this link may disappear because the new document F appears in the top n and pushes E out of it. When documents are removed from the handbook, links to them will disappear, which might cause the algorithm to create new links to already existing documents.

Nonmonotonicity makes the hypertext system unpredictable: whether there is a link between documents is not only dependent on the contents of both documents itself, but also on other documents. This unpredictability should be avoided since it may confuse readers. The main reason for having a link between two documents should be that they 'are documents about the same thing'. Although this aboutness is a subjective notion, it is not a phenomenon that changes over time, thus causing links to appear or disappear. One can avoid nonmonotonicity by not linking to the top n of retrieved documents, but by linking to all documents with a similarity value above a certain threshold. However, the problem with this approach is that such a threshold is hard to find, as discussed above, and the number of links becomes uncontrollable, causing some documents to have too much links to other documents, while others might have too little links. For example, at the time the threshold is set, every concept may be linked to 5 articles about the topic. However, maybe the hierarchy will later give access to a collection double as large, causing there to be 10 links from every concept.

Verifying the Handbook's Contents. IR techniques can be used for more purposes than only generating links. They can be used to check the correctness and completeness of the hierarchy (see Section 4.4) with respect to the material it gives access to. For example, if there are parts of the handbook that do not have sufficient similarity to any concept in the concept hierarchy, the hierarchy is not complete. On the other hand, when the hierarchy contains concepts that do not have sufficient similarity to any part of the handbook, the hierarchy is incorrect; parts of it are not treated by the handbook. Using the techniques of Allan (Section 5.3.4) to visualize relationships between documents, even more information about the relationship between the hierarchy and the handbook or two parts of the handbook can be obtained. This way, the construction of the hierarchy as discussed in Chapter 4 can also be supported by tools described in the current chapter.

IR techniques might also be used to check new contributions to the handbook text: as described above, they can be used to check the correctness and completeness of the hierarchy with respect to the new contributions. In addition to this, IR techniques can verify whether the structure of the contribution itself is good; for example, with Allan's algorithm we can verify whether a summary section of a contribution is indeed a good summary or not. More research is needed to find out whether other kinds of coherency checks on new contributions can be automated with the help of IR techniques.

Chapter 6

Conclusions and Future Work

In this chapter, we first summarize the thesis. After that, we will discuss the next steps for the L3 project. In Section 6.3 we describe the output we expect the L3 project to deliver.

6.1 Summary

In this thesis we have presented the initial results of the Logic & Language Links project, a project that deals with the electronic publication of scientific information that is traditionally found in a handbook.

In Chapter 2 the requirements of four user groups of the electronic handbook are described. We focus on the readers of the handbook and the authors contributing to it. Two important themes have arisen from those requirements: good *access* to the handbook and *navigation* through the handbook. Developing a concept hierarchy for navigation and access to the handbook is an enterprise that addresses those themes and most of the discussed requirements.

Chapter 3 continues with a description of the concept hierarchy. The hierarchy is based upon concepts, represented by a synset, a set of synonymous terms. The concepts can also carry a gloss, and a more extended description like entries in the glossary of the Handbook of Logic & Language or the Bluffer's Guide to Computational Semantics. It is important that the concepts carry bibliographical metadata, to determine their author, creation date, etc. The concepts in our hierarchy are *monothetic*, i.e. each concept is about one well-defined topic. The idea is that the concepts of the hierarchy form a concise and abstract mirror of the handbook's contents.

The main structure of the hierarchy is formed by the *related subtopic* relation (and its inverse). The subtopic relationship should form a directed acyclic graph together with the concepts. The other relationships between the concepts in the hierarchy are *similarity*, *antonymy*, *sibling* and *other meanings*. Thus, we explicitly assign a type to the links; we also discuss the advantages of this typing. Besides the links between the concepts, there are several types of links that connect the hierarchy to the handbook and the WWW.

The hierarchy must be displayed to the reader. We present several ways to do this and motivate our decision to use a Yahoo!-like visualization. We also discuss the issue of evaluating user interfaces.

Chapters 4 and 5 deal with tools to support the authors and constructors of the hierarchy in building and maintaining it and linking it to the Web. There are many such tools, but at the moment we do not think they are good enough to employ autonomously, without human supervision. However, this is a goal to strive after.

Chapter 4 describes the construction of the concept hierarchy. We first describe the material on which we base the concept hierarchy: the glossary of the Handbook of Logic & Language and the Bluffer's Guide to Computational Semantics. After that we discuss the current, prototypical, implementation of the concept hierarchy. Following this, we describe some methods to automatically construct concept hierarchies, and how to apply them in the L3 project. The last part of the chapter deals with evaluation of the quality of the constructed hierarchies. The evaluation can be performed with subjects that judge the hierarchical relationships or perform information seeking tasks, but also with the help of metrics providing information about the hypertext structure.

Finally, Chapter 5 deals with the creation of links from the concept hierarchy to the handbook and to the WWW. We describe several methods from literature for the creation of those links, which are based on Information Retrieval and text classification techniques. We also report experiments with link generation using IR techniques that we performed ourselves, from which we conclude that at present we should not automate this task entirely, but that the editor of the hierarchy can use IR tools to support the link generation task. Furthermore, such tools should make use of the structure of the concepts in the hierarchy; we found that if the names of concepts in the hierarchy are weighted more heavily than the descriptions of the concepts, retrieval performance improves. Further research in this direction can concentrate on using more of the structure of the (concepts in the) hierarchy to find good links.

We also discussed the generation of links from the concept hierarchy to the WWW, and the automatic construction of wrappers, software programs that are used for this purpose. An important issue in this respect is the evaluation of generated links and wrappers. We finished this chapter with the interesting issues of non-monotonicity of automatically linked hypertext, and automatically assessing the coherence of new handbook contributions using IR techniques.

6.2 Next Steps

The next steps for the L3 project fall under four themes: pilot implementation, involving the ILLC community with the construction of the hierarchy, further research in tools for supporting the authors, and evaluation. We will discuss them below.

Pilot Implementation. The current prototypical implementation that we described in Section 4.3 has been useful for experimenting with the ideas we had. However, at present we need to involve more people with the construction of the concept hierarchy. This requires the following:

- Example concepts provided by the people involved with the L3 project.

- Guidelines for authors who are invited to contribute. Together with the examples those guidelines can help the authors to write in a coherent way, thus achieving some uniformity in the structure of contributions written by different authors.
- A better implementation of the hierarchy using a database system that is connected to the web. This way, people on different locations can inspect the concept hierarchy. There should also be facilities for the contributing authors to add their contributions via the Web through an easy to use interface and using a familiar format like LaTeX. Those contributions should be stored in a structured way that makes it possible to identify the different ingredients of the concepts and links stored. Storing as much information as possible separately enables retrieval by querying on such things as author, concept name, links leaving the concept and links towards the concept.

Involving the ILLC Community. The construction of a pilot implementation is not a very interesting research question in its own right (it rather is a software engineering problem), but it is very important since it is a precondition for involving more people from ILLC with the construction of the concept hierarchy. If this involvement via electronic means is a success, we can widen our scope and try to involve, through the contacts of ILLC, researchers from outside ILLC in the discipline of the Handbook. This will result in a very well structured concept hierarchy mapping out an entire scientific area, with a lot of high quality links to relevant material about its contents.

It will be interesting to see whether it is possible to have the hierarchy grow and improve without the presence of an editor that makes the important decisions, but with an entire community working simultaneously and in parallel on the structure of such a hierarchy — thus getting an overview of the scientific discipline that is widely agreed upon.

The resulting concept hierarchy will first give access to the contents of the Handbook. However, it might turn out that the structure of the concept hierarchy will differ very substantially from the Handbook's structure and constitute an improvement over it. In this case, the hierarchy might serve as a blueprint for the rewriting of existing chapters, or indicate the need for addition of new chapters.

The advantages of working on such a hierarchy via electronic means are numerous. People on different geographic locations, working at different times, can still work together on the construction of the hierarchy and monitor each other's work. A physical meeting of an entire scientific community to work on such a structuring of the discipline might practically be impossible, while a 'virtual meeting' via additions to the hierarchy and reactions to each other's concepts is well possible.

The above will result in a well structured concept hierarchy that might even function as a blueprint for the Handbook's contents. This will be a very interesting result. If all the decisions made in the construction process are well documented, and characteristics of the Logic and Language community are made explicit, the current enterprise can serve as an example for the the writing of handbooks about other scientific disciplines.

Development and Application of Supporting Tools. In Chapters 4 and 5 we discussed the use of tools to support the people building the concept hierarchy. Part of it was based on the literature, but we also presented the results of our own experiments with the generation of links from the hierarchy to the handbook and the generation of wrappers.

With respect to the tools to build concept hierarchies (Chapter 4), we are now at a stage where we can apply them. Those methods need as input a collection of documents. Of course we can use the Handbook text for this purpose, but we suppose that using more documents will result in a better hierarchy. To verify this, we should collect more documents on the field of Logic and Language. For this purpose we can use papers on the ILLC ftp server[1] and available material from researchers' homepages and publisher's sites. It will be interesting to see in what respect the generated hierarchies differ from the hand-crafted one, and to what degree they can contribute to the extension of the latter.

With respect to the tools for link generation, further work can be done in several directions:

- To enhance precision, we can make more use of the structure of the concepts, for instance by using different weightings for different parts of the concepts and employing phrase-based IR techniques when querying on the name of the concept.

- We can make more use of the structure of the concept hierarchy, by sorting relevant documents through the hierarchy, and employing similarity measures that exploit its structure.

- We can try to find a set of variables that are predictive of the usefulness of a generated link and find a threshold function to automatically distinguish useful links from useless ones.

A less technical and more fundamental issue that should be investigated with respect to the generation of links is the ideal relation between the granularity of a link's source and its target. For example, a link from a concept at the lowest level in the hierarchy to an entire handbook chapter will be far from ideal, since it is a highly specialized concept to which one would never devote an entire chapter.

Zooming out of the individual tools, guidelines should be developed for the integration of the results of several tools that the editor can use. A feature of the tools for the construction and linking of the hierarchy is their domain independence, due to which they can be applied to other disciplines than that of the Handbook.

Evaluation. A theme that occurs throughout Chapters 3, 4 and 5 is evaluation. It is an important theme, since the construction of a concept hierarchy only makes sense when people find it practical to use. At present, we are not yet in a stage where we can start evaluating. Still, we would like to mention it here since we think it is an important and interesting aspect of research about electronic publishing.

[1] `ftp://ftp.wins.uva.nl/pub/theory/illc/`, visited on January 5, 2001.

Usually, some sort of *empirical evaluation* is described in literature: what do users think of the generated hypertext, and how well do they perform when they use it? However, as Chapters 3, 4 and 5 show, empirical evaluation in this research area has been a bit poor. The reason for this might be the efforts involved with finding test persons, designing a good test, and interpreting the obtained data. Besides the big effort, many researchers in the area of computer science may not possess the required knowledge about statistical methods and design of experiments.

On the other hand, in Section 4.5.2 we also discussed the *objective evaluation* of hypertext using metrics about its structure and the paths followed through the hypertext. When one has the right metrics at one's disposal, this is a much easier way of evaluation, especially if those metrics can be related to empirical data about user performance, like in McEneaney's work.

In that case, when enough empirical evaluation has been performed, this can be replaced by the simple calculation of some metrics. Further research is needed in this respect: there are not many such metrics yet, so others can be invented, and more empirical data should be gathered with hypertexts; finally, the relation between both should be investigated. Unfortunately, the ease of use of a hypertext does not solely depend on the structure of the underlying graph, so to ensure a that a hypertext is practical to use, empirical evaluation can probably never be omitted.

6.3 Output of the L3 project

If the research in the L3 project continues in the directions described in the previous section, the output of the project will consist of three closely interconnected pieces of work: a concept hierarchy for the Logic and Language domain, domain independent algorithms supporting the construction and maintenance of such a hierarchy and a methodology for building concept hierarchies in other scientific areas.

Logic and Language Hierarchy. This will be the most tangible result of the L3 project. The entire scientific discipline will be summarized in this hierarchy, and the relationship between several parts of the discipline will be made clear. The implementation of the hierarchy will be available on the WWW, and will be evaluated thoroughly.

Domain Independent Algorithms for Construction and Maintenance. The further development of such algorithms that can make use of the knowledge present in concept hierarchies might result in tools that can support the maintenance of the hierarchy, or even automate it entirely. Since widely applicable methodology will be used (e.g. IR algorithms), such tools for hierarchy construction and automatic link generation can be applied to other disciplines as well.

Since this is not a software engineering project, one should not expect that the project will deliver one integrated piece of software for all tasks involving the hierarchy. However, there should be guidelines for the use of the tools and how to handle and integrate their output.

Methodology for Hierarchy Building. Although the hierarchy itself will be an important proof of the fact that a hierarchy can be built by involving a scientific community, maybe more important is the way by which this achieved. Good documentation of the choices we have made in the process will result in a methodology applicable to other disciplines as well.

Bibliography

[All95] James Allan. *Automatic Hypertext Construction*. PhD thesis, Cornell University, January 1995.

[All97] James Allan. Building hypertext using information retrieval. *Information Processing & Management*, 33(2):145–159, 1997.

[Blu99] William James Blustein. *Hypertext Versions of Journal Articles: Computer-aided linking and realistic human-based evaluation*. PhD thesis, The University of Western Ontario, London, Ontario, March 1999.

[Bra90] Ivan Bratko. *Prolog Programming for Artificial Intelligence*. Addison Wesley, second edition, 1990.

[BRS92] R.A. Botafogo, E. Rivlin, and B. Shneiderman. Structural analysis of hypertexts: Identifying hierarchies and useful metrics. *ACM Transactions on Information Systems*, 10(2):142–180, 1992.

[BYRN99] Ricardo Baeza-Yates and Berthier Ribeiro-Neto, editors. *Modern Information Retrieval*. ACM Press, 1999.

[Can00] Marcel Canoy, editor. *Publishers: caught in the Web?* CPB Netherlands Bureau for Economic Policy Analysis, The Hague, The Netherlands, February 2000. Available for download at `http://www.cpb.nl/nl/pub/pubs/werkdoc_119/overview` (visited on January 5, 2001).

[Con87] Jeff Conklin. Hypertext: An introduction and survey. *IEEE Computer*, pages 17–41, September 1987.

[CRdR00] Boris Chidlovskii, Jon Ragetli, and Maarten de Rijke. Wrapper generation via grammar induction. In Ramon López de Mántaras and Enric Plaza, editors, *Proceedings ECML 2000*, number 1810 in LNAI, pages 96–108. Springer, 2000.

[CS94] Richard Chimera and Ben Shneiderman. An exploratory evaluation of three interfaces for browsing large hierarchical tables of contents. *ACM Transactions on Information Systems*, 12(4):383–406, October 1994.

[DD97] Harvey M. Deitel and Paul J. Deitel. *Java How to Program*. Prentice Hall, second edition, 1997.

[DeR89] Steven J. DeRose. Expanding the notion of links. In *Proceedings ACM Hypertext '89*. ACM, 1989.

[Fel98] Christiane Fellbaum, editor. *WordNet, an Electronic Lexical Database*. MIT Press, 1998.

[FLGD87] G.W. Furnas, T.K. Landauer, L.M. Gomez, and S.T. Dumais. The vocabulary problem in human-system communication. *Communications of the ACM*, 30(11):964–971, November 1987.

[Fra96] L.R.E. Fracas. The bluffer's guide to computational semantics, January 1996.

[GR99] Michel Goossens and Sebastian Rahtz. *The LaTeX Web Companion*. Addison Wesley Longman, Inc., 1999.

[Gre97] Stephen J. Green. *Automatically generating hypertext by computing semantic similarity*. PhD thesis, University of Toronto, Computer Systems Research Group, 1997. Available as Technical Report number 366.

[Gro97] Willem Groeneveld. Logic and language: A glossary. In van Benthem and ter Meulen [vBtM97], pages 1179–1213.

[Har00] Frédérique Harmsze. *A modular structure for scientific articles in an electronic environment*. PhD thesis, Universiteit van Amsterdam, February 2000.

[HD98] C.-N. Hsu and M.-T. Dung. Generating finite-state transducers for semistructured data extraction from the web. *Information Systems*, 23(8):521–538, 1998.

[Hea99] Marti A. Hearst. User interfaces and visualization. In Ricardo Baeza-Yates and Berthier Ribeiro-Neto, editors, *Modern Information Retrieval*, chapter 10, pages 257–323. ACM Press, 1999.

[HK97] Marti A. Hearst and Chandu Karadi. Cat-a-cone: An interactive interface for specifying searches and viewing retrieval results using a large category hierarchy. In *Proc. of the 20th Annual International ACM SIGIR Conference*, pages 246–255, 1997.

[Jas99] Jan Jaspars. Johan's worlds. In Jelle Gerbrandy, Maarten Marx, Maarten de Rijke, and Yde Venema, editors, *JFAK. Essays Dedicated to Johan van Benthem on the Occasion of his 50th Birthday*, Vossiuspers. Amsterdam University Press, Amsterdam, 1999. Published on CD-ROM.

[KH00] Joost G. Kircz and Frédérique Harmsze. Modular scenarios in the electronic age. In *Proceedings Conferentie Informatiewetenschap*, 2000.

[KZ97] Marcin Kaszkiel and Justin Zobel. Passage retrieval revisited. In *Proceedings SIGIR 97*, pages 178–185, 1997.

[LF99] Yannis Labrou and Tim Finin. Yahoo! as an ontology — using yahoo! categories to describe documents. In *CIKM '99. Proceedings of the Eighth International Conference on Knowledge and Information Management*, pages 180–187. ACM, 1999.

[McE99] John E. McEneaney. Visualizing and assessing navigation in hypertext. In *Proceedings of the tenth ACM Conference on Hypertext and hypermedia: returning to our diverse roots*, pages 61–70, 1999.

[MNRS00] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000.

[Mue97] Victoria Lynn Muehleisen. *Antonymy and Semantic Range in English*. PhD thesis, Northwestern University, June 1997. Available on the Web at: `http://www4.justnet.ne.jp/~vicky/dissertation/` (visited on January 5, 2001).

[NMWP99] C.G. Nevill-Manning, I.H. Witten, and G.W. Paynter. Lexically-generated subject hierarchies for browsing large collections. *International Journal of Digital Libraries*, 2(3):111–123, 1999.

[Par91] H. Van Dyke Parunak. Ordering the information graph. In Emily Berk and Joseph Devlin, editors, *Hypertext / Hypermedia Handbook*, chapter 20, pages 299–325. McGraw-Hill Publishing Company, 1991.

[Rét99] J.-H. Réty. Structure analysis for hypertext with conditional linkage. In *Proceedings of ACM Hypertext '99*, Darmstadt, 1999.

[SBS98] T. Sumner and S. Buckingham Shum. From documents to discourse: Shifting conceptions of scholarly publishing. In *Proceedings CHI 98: Human Factors in Computing Systems*, 1998.

[SC97] Randal L. Schwartz and Tom Christiansen. *Learning Perl*. O'Reilly & Associates, Inc., second edition, 1997.

[SC99] Mark Sanderson and Bruce Croft. Deriving concept hierarchies from text. In *Proceedings ACM SIGIR Conference '99*, pages 206–213, 1999.

[SFC98] P.D. Stotts, R. Furuta, and C.R. Cabarrus. Hyperdocuments as automata: Verification of trace-based browsing properties by model checking. *ACM Transactions on Information Systems*, 16(1):1–30, 1998.

[Teb99] John Tebbutt. User evaluation of automatically generated semantic hypertext links in a heavily used procedural manual. *Information Processing & Management*, 35(1):1–18, 1999.

[Thi95] Paul Thistlewaite. Automatic construction of open webs using derived link patterns. In Maristella Agosti and James Allan, editors, *IR and Automatic Construction of Hypermedia: A Research Workshop*, July 13, 1995.

[Thi97] Paul Thistlewaite. Automatic construction and management of large open webs. *Information Processing & Management*, 33(2):161–173, 1997.

[Tri83] Randall H. Trigg. *A network-based approach to text handling for the online scientific community*. PhD thesis, University of Maryland, November 1983. The link taxonomy is given in chapter four which is available on `http://www.parc.xerox.com/spl/members/trigg/thesis/thesis-chap4.html` (visited on January 5, 2001).

[vBtM97] Johan van Benthem and Alice ter Meulen, editors. *Handbook of Logic and Language*. Elsevier Science B.V., 1997.

[vE98] Jan van Eijck. Logic and Language Links (LoLaLi) — finalized plan, November 1998.

[vH94] Eric van Herwijnen. *Practical SGML*. Kluwer Academic Publishers, 1994.

[VH97] E.M. Voorhees and D.K. Harman, editors. *The Sixth Text REtrieval Conference (TREC 6)*. Department of Commerce, National Institute of Standards and Technology, 1997. NIST Special Publication 500-240. Available on the Web at `http://trec.nist.gov/pubs/trec6/t6_proceedings.html` (visited on January 5, 2001).

[Voo97] E.M. Voorhees. Database merging strategies for searching public and private collections. In *ACM SIGIR '97 Workshop on Networked Information Retrieval*, 1997.

[War99] Colin Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann Interactive Technologies Series. Morgan Kaufmann, San Francisco, CA, 1999.

[WMB94] Ian H. Witten, Alistair Moffat, and Timothy C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Van Nostrand Reinhold, 1994.

[YL99] Yiming Yang and Xin Liu. A re-examination of text categorization methods. In *Proceedings SIGIR '99*, 1999.

[ZGG+99] Xiaolan Zhu, Susan Gauch, Lutz Gerhard, Nicholas Kral, and Alexander Pretschner. Ontology-based web site mapping for information exploration. In *CIKM '99. Proceedings of the Eighth International Conference on Knowledge and Information Management*, pages 188–194. ACM, 1999.

Appendix A

Glossary

Browser. Computer program for the navigation through hypertext. Often used browsers for the World Wide Web are Netscape and Internet Explorer.

Concept hierarchy. Topic hierarchy, in which topics can also bear an explanatory description.

DAG. Directed Acyclic Graph. A graph whose edges are ordered pairs of vertices (nodes), and in which there is no path that starts and ends in the same vertex.

HTML. Hypertext Markup Language. Language to specify lay-out and hyperlinks in documents on the Web.

Hyperlink. Connection between two chunks of text in a hypertext system, allowing users of the system to go from the first document (source) to the other document (target or destination).

Hypertext. Nonlinear text, text that can be read in a different order than in sequence, and thus cannot conveniently be printed on a conventional page. An example of a (very large) hypertext is the WWW. An introduction dating form the pre-web era, and giving insight in using hypertext for more purposes than only reading, is [Con87].

Java. Object-oriented programming language, very suitable for programming for the Web. A good introduction is [DD97].

Perl. Practical Extraction and Report Language. Programming language, often used for manipulating text files and generating HTML pages for the Web. A good introduction is [SC97].

PDF. Portable Document Format. It is a universal file format that preserves all of the fonts, formatting, colors, and graphics of any source document, regardless of the application and platform used to create it. It has been developed by Adobe; PDF files can be viewed with the program Acrobat Reader.

Portal site. Web site that that gives access to a lot of information on the WWW, and often includes a search engine. Many portal sites focus on one particular topic. A typical general portal site is for example `http://www.startpagina.nl` (visited on January 5, 2001), whereas its daughters are sites focusing on particular topics (e.g. `http://vakantie.pagina.nl` (visited on January 5, 2001)).

Search engine. Program, usually accessible via the WWW, that can be queried by user to find information on the WWW, relevant to his query. An example is Google, on `http://www.google.com/` (visited on January 5, 2001).

SGML. Standard Generalized Markup Language. A language for document representation that formalizes markup and frees it of system and processing dependencies. SGML documents have a rigorously described structure that may be analyzed by computers and easily understood by humans. A good textbook on SGML is [vH94].

Topic hierarchy. Hierarchical representation of some domain, where every topic except the root is connected to another topic via a subtopic relationship.

URL. Uniform Resource Locator. A URL is used to address any Internet resource, including web pages and ftp servers.

WWW. World Wide Web. A popular link-based hypertext system running on the Internet using a client-server architecture.

XML. Extensible Markup Language. Subset of SGML, that is easier to learn, use, and implement than SGML. It is becoming the successor of HTML as markup language for the WWW, already supported by the latest releases of the popular Web browsers.

Yahoo! Large category hierarchy on the WWW, where every category contains a number of links to web pages about this category. URL: `http:www.yahoo.com` (visited on January 5, 2001).