

Chapter 2

Some Useful Ideas and Background Theories

2.1 Reid on Abstraction and Classification

There are some valuable arguments in Thomas Reid's 1785 essay 'Abstraction' (Reid 1785). [Reid was a nineteenth century Scottish 'common sense' philosopher and the essay is from his *Essays on the Intellectual Powers of Man*. Interestingly enough the librarian pioneer, Charles Cutter, was influenced by exactly this essay (Miksa 1983). Indeed, Reid himself had worked as a librarian. (Fraser 1898 p. 27).]

Once a child understands plurals, that, for example, he or she has *two* brothers or *two* sisters or *several* toys, he or she understands that there are *attributes* such that the same attribute can be possessed by different things. There is the one item, one of the toys, that has perhaps several different *attributes*, such as being red, chewed, being in the play pen, and being a toy, and another item, which also has several attributes, such as being blue, being in a person's hand, being desired, and also being a toy. And at least one of these attributes, 'being a toy', is possessed by two different things. Thus, the child sees, touches, and interacts with things individually, and understands that there are attributes which are possessed by, or are instantiated by, different individual things. There are two processes here: the distilling out of one property of interest (being a toy) from the many other properties that the each item individually possesses (being chewed, being blue), and the recognition that the property is common to more than one different subject.

This is the basis of classification, as Reid puts it

Our ability to distinguish and give names to the different attributes belonging to a single thing goes along with an ability to observe that many things have certain attributes in common while they differ in others. This enables us to put the countless hordes of individuals into a limited number of classes, which are called 'kinds' and 'sorts'—and in the scholastic language are called 'genera' and 'species'. (Reid 1785, p. 191)

[Notice that Reid, uses the words ‘kinds’ or ‘sorts’ as synonyms for ‘classes’. This usage is common, both then and now, and we will adopt it. The word ‘sort’ is particularly useful in classification in as much it is not overused and, unlike, for example, ‘class’ or ‘type’, does not carry the baggage of many other meanings.]

Attributes allow us to form classes, and these classes are often related one to another other as subclasses or superclasses; for example, the class of humans is a subclass of the class of animals. One of Reid’s examples is the class hierarchy

Animal
Man
Frenchman
Parisian

The animal class can be identified by the possession of the being-an-animal attribute. The ‘genus’ of animal can then be subdivided as a class by the application of further attributes to produce the ‘species’ of man. Then man, considered itself as a genus, can then be further divided in a similar way to produce the species of Frenchman. And so on. Items belonging to classes lower down the hierarchy inherit all the attributes from the superclasses (so, for example, a Parisian inherits the attributes of Frenchman, and if one of those attributes is ‘living in France’ then a Parisian lives in France).

So, classification results in a compression and codification of knowledge and meaning

It is by means of such extensive and comprehensive propositions that human knowledge is condensed, as it were, to a size suitable for the capacity of the human mind, adding greatly to its beauty without making it any less clear or any harder to absorb (Reid 1785, p. 193).

The classes, or the general words or labels for them, can be used either as subjects in sentences or propositions, or as predicates or straight attributes. So, for example, in ‘A Parisian lives at the center of the cultural world’ or ‘A Parisian is French’ the general word or general conception ‘Parisian’ is the subject of the sentences and the assertions are in terms of classes (or genus and species), and in ‘Suzette is a Parisian’ the ‘is a Parisian’ is a predicate being attributed to the individual person Suzette (who is an instance of the class or sort). So there is a kind of duality between the sort Parisian and the attribute ‘is a Parisian’ which generates it.

And, as Reid tells us, this ability to classify is possessed by almost everyone in every culture.

Such divisions and subdivisions of things into genera and species, with general names, are not confined to learned and polished languages; they are found in the languages of the most primitive human tribes. This tells us that the invention and the use of general words, to signify both the attributes that things have and the genera and species that they fall into, is not a subtle invention of philosophers but rather an operation that all men perform by the light of common sense. (Reid 1785, p. 192)

2.2 Ontologies, Gerrymanders, Natural Kinds and Taxonomies

These relations between classes, mentioned by Reid, result usually either from the way our world is, or from decisions we make and formulate in language. This is addressed by *ontology* and related topics.

The word ‘ontology’ is used in traditional philosophy and it means ‘what there is or what exists’. There is a modern use of ‘ontology’, within Computer Science and Artificial Intelligence, where it has a slightly different meaning and that is ‘conceptual scheme’ or ‘conceptualization’. ‘Ontology’ also gets used in database design where it amounts to ‘a description of the types or kinds of entities, and the properties or attributes, that are assumed to exist for the purposes of the database’.

These various uses of the word are not that different one from another and here it will be most useful to adopt a fairly modern style under which ontologies are generally small, local, and confined to particular areas or domains. So, as examples, if the focus of interest is with military personnel and armies, a suitable ontology might contain soldiers, ranks, squads, companies, brigades, divisions, weapons, and so forth; if the focus of interest is with cooking, an ontology might contain recipes, ingredients, proteins, flavoring, condiments, carbohydrates, fats, and so on.

There is an important distinction that can be made between ontologies that depend on us, depend on us humans, and those that do not. An ontology for the military of soldiers, ranks, squads, companies, etc. is a human creation. It is just conjured up to suit a purpose. It depends on us. It is *gerrymandered*. But not all ontologies are like this. There is a world we live in external to us, and there are things in that world, and things happening in that world. As examples, there are species of animals, such as tigers and lions and lizards, and there are chemical elements and compounds, such as hydrogen and oxygen and iron and rust. That there is a kind of animal that is a tiger does not depend on us, it has to do with the way the world is. Similarly, that there is the kind of substance iron, and a type of process in which some iron turns to rust over time, is a feature of the world. There is an evocative metaphorical phrase that comes into play here, and that is ‘carving nature at its joints’ (Hempel 1965). Nature is not a blooming buzzing confusion, on which we impose order; rather it consists of perfectly good things existing on their own, and going about their own business, without help from us. And if an ontology carves nature (or some of it) at its joints, the ontology fits what is there. An ontology that does this is an ontology of *natural kinds* for it deals with the kinds that are in nature (Reid calls these ‘natural substances’). Rust is a natural kind, whereas a company of soldiers is not.

Natural kinds feature in the causal structure of the world. And gerrymandered kinds do not. Here is what John Stuart Mill had to say in 1843 about the gerrymandered kind of white objects, as contrasted with some of the natural kinds of biology and chemistry

White things are not distinguished by any common properties, except whiteness; or if they are, it is only by such as are in some way connected with whiteness. But a hundred

generations have not exhausted the common properties of animals, of plants, of sulphur or phosphorus; nor do we suppose them to be exhaustible, but proceed to new observations and experiments, in the full confidence of discovering new properties, which were by no means implied by those we previously knew (Mill 1843) quoted from (Koslicki 2008, p. 791)

These additional common properties come from the causal structure of the world.

Of course, we do not have a direct infallible insight as to what it is in the world, as to what the natural kind ontologies are. We have to do science, to conjecture, and to propose theories to find out [see also (Haack 1999)]. For example, Mendeleev's 1869 familiar periodic table of chemical elements was a classification of chemical elements; it was an elaborate catalog. Mendeleev suggested on the basis of it that there were three missing elements that were yet to be discovered; and, indeed, the elements gallium, scandium, and germanium were later discovered. That this happened resulted from the fact that Mendeleev's classification in some sense locked on to what is actually there in the world. And, to contrast with our successes, there are plenty of examples where we have been mistaken. Prior to Lavoisier, about 1770, *phlogiston* was supposed to be a chemical element; so early eighteenth century chemical ontologies would have included phlogiston as a natural kind; but Lavoisier showed that there was no such element as phlogiston, hence the natural kind chemical ontology of the time needed revision. As Brian Ellis phrases it

Every distinct kind of chemical substance would appear to be an example of a natural kind, since the known kinds of chemical substances all exist independently of human knowledge and understanding, and the distinctions between them are all real and absolute. Of course, we could not have discovered the differences between the kinds of chemical substances without a lot of scientific investigation. But these differences were not invented by us, or chosen pragmatically to impose order on an otherwise amorphous mass of data. There is not continuous spectrum of chemical variety that we had somehow to categorize. The chemical world is just not like that. On the contrary, it gives every appearance of being a world made up of substances of chemically discrete kinds, each with its own distinctive chemical properties. To suppose otherwise is to make nonsense of the whole history of chemistry since Lavoisier. (Ellis 2008, p. 140)

[See also (Smith 2004).]

Another label for a natural kind hierarchical ontology is a 'taxonomy' (classically, 'taxonomy' means 'arrangement by law', the ancient Greek 'taxis' is 'arrangement' and 'nomos' is 'law'). The notion of natural kinds, or natural kind ontologies, can be extended from the empirical world to mathematics and logic. Mathematical entities typically have many properties that are not of our creation.

In a sense there is a third possibility between natural kinds and gerrymanders, and that is *conventional* or *constructed* ontologies. And these are *structured gerrymanders*. For example, in knowledge representation—a discipline of interest to Artificial Intelligence, to Database design and to Information Retrieval—the use of ontologies, especially taxonomies is widespread. A typical problem here might be how to model or represent Human or Employee Resources for a large company, and data of interest might include who the employees are, their salaries, their

qualifications, whether they are full or part-time, what their benefits are, and so on. And a suitable representational scheme might use a hierarchical taxonomy where every item is an employee, but then a subtype of these are full-time, another subtype part-time, and so on. Such a taxonomy is not a natural kind ontology—it is not a feature of the world we live in that part-time employees work less than, say, 25 h a week. The Database designers, with input from the company, simply decided that. The ontology is a human construction. It is a gerrymander. However, there is a lot more to it than there is to Mill's collection of white things. It has structure, and although it is a creation of humans, humans might not be immediately aware of all the properties and features it has. And it can be used for inference. For example, learning that John Smith is a full-time employee may well permit an inference to what his benefits are. And it could be that no one knows what John Smith's benefits are prior to this inference being made—knowledge is acquired from the ontology in conjunction with facts about the world.

When representing knowledge or information, a distinction might be made between items of knowledge that are specific or particular and items which are generalizations or are conventional or natural laws. So, that John Smith is a full-time employee is an item of information that is specific, that all full-time employees receive benefits is an item of information that is a generalization of convention. One system of organization puts the items of information that are specific into a database or similar structure, and the items of information that are general into an ontology or classification scheme. Then the classification scheme can be used with the facts to make further inferences. This approach can be used either with natural kind ontologies or with structured gerrymandered ontologies.

Reid alerted us to the advantage of using classification to compress human knowledge (that it allows us to absorb and retain that knowledge). But for this to happen the classifications need to be correct, or true, or realistic. The classifications must constitute knowledge. This is exactly the advantage and importance of natural kind ontologies. They are realistic, and they do embody knowledge about the world we live in.

2.3 Concrete Objects and Abstractions

There is an important distinction to be made between 'concrete' and 'abstract'. Many things are physical and concrete, that is, they are located in space and time. Consider diamonds, those forever gemstones. The Hope Diamond is a physical thing, and, at any particular time, it is in a particular place (usually the Smithsonian Museum in Washington). But things can also be abstract. Jane Austen's novel *Pride and Prejudice* certainly exists embodied in many concrete physical forms (as the actual manuscript she wrote, as the physical book in the local public library, etc.). But when a high school class reads *Pride and Prejudice*, as part of their syllabus and studies, they all read the same novel or book, namely *Pride and Prejudice*, but they very likely will be physically reading different

physical concrete copies, because each pupil will have his or her own copy. The ‘same’ book that they read is an abstraction.

It will be useful for us to spend a moment or two on the nature of concrete objects and abstractions and the distinction between the two.

The ancient Greeks drew a different twofold distinction among existing entities viz., a distinction between physical entities and mental entities. So papyrus copies of *Goatherd and the Palace Guards* were physical, but the novel itself, as discussed at dinner parties in ancient Greece, was, or would have been considered to be, mental or a mental idea (i.e. existing among the contents of someone’s consciousness or mind or, perhaps, in many peoples’ minds).

That, more or less, would have been the received philosophical view for 2000 years until the work of the late nineteenth century German philosopher, Gottlob Frege. Frege was not content with the twofold distinction between the physical and the mental, because, in this context, the ‘mental’ simply would not do the work it was supposed to. When, at that ancient Greek dinner, Aesop and Agatha were discussing *Goatherd and the Palace Guards*, they certainly would have thought that they were discussing the *same* thing. But if that thing was a mental idea, or just a mental idea, whose idea was it? Were they discussing Aesop’s mental consciousness? Or were they discussing Agatha’s mental consciousness? Or what? Clearly, what they were discussing was indeed the same thing, but it was not either of their individual mental consciousnesses; rather it was something that was common to those consciousnesses, it was what those consciousnesses were *of* or *about* (i.e. in this case, *Goatherd and the Palace Guards*), it was the content of those consciousnesses. Just as separate papyri can be instances relating *Goatherd and the Palace Guards* so too can separate psychological states share the commonality of referring to or depicting, in part, *Goatherd and the Palace Guards*. Frege invoked a ‘third realm’. And that third realm consisted of abstractions (which were not mental or psychological ideas). Frege’s view was later taken up by others; for example, the twentieth century philosopher Karl Popper distinguished between three worlds: the physical world, the mental world, and the world of abstract contents (Popper 1972). We also will use this same threefold distinction, but our focus will be with the abstract and the concrete (the notion of mental or psychological phenomena is worthy of considerable study and research, but it is not one of the topics of this book).

How is the concrete to be distinguished from the abstract? The concrete is located in space and in time; at a particular time, concrete items are in particular locations (as we noted earlier, on April 17, 2010, the Hope Diamond is some definite place, perhaps in the Smithsonian Museum). Concrete items are also causally efficacious; that is: they can be the causes of effects on other items, and other items can be the causes of effects on them; a ‘concrete’ physical copy of *Pride and Prejudice* can be dropped and it can knock over a cup of tea, and, in turn, the spilled tea can stain the pages of the copy of the book. In contrast, the abstract work *Pride and Prejudice* (which the High School class read) has no definite physical location, it is nowhere in particular, and it is not causally efficacious, it cannot knock over cups of tea nor can its pages be stained. In most of the cases we will be interested in, abstractions can be exemplified in concrete

objects. The abstract *Pride and Prejudice* has many concrete books and volumes that exemplify, or instantiate, it.

Returning for one moment to Reid, the child and toys. The individual toys are concrete and physical; but the attribute ‘being a toy’ is not concrete and physical, it—the attribute that is—is an abstraction. The attribute ‘being a toy’ does not exist in space and time and it cannot be causally efficacious. Reid counsels us that there is nothing mysterious here: the abstraction ‘being a toy’ exists in the sense that it is capable of being instantiated by the individual toys.

When we ascribe existence to [abstractions], it is not existence *in time or place* but existence *in some individual subject*; and all that *this* existence means is that they are truly attributes of such a subject. Their existence is merely predicability, i.e. the capacity to be attributed to a subject. The name ‘predicables’ that was given them in ancient philosophy is the one that best expresses their nature. (Reid 1785, p. 210)

Abstractions exist as possibilities (the possibility of being instantiated).

2.4 Entities, Properties and Statements

As language users, we have a way of viewing the world that is going to pervade the present work, it is going to reveal itself in how we think, in how we talk, in how we classify, in how we analyze logically, and in other places besides. Let us be explicit about it.

In broad terms, we see, and talk about, the world as though it consists of two kinds of items. There are *entities* (which are things, or objects, or individuals, or events, or processes), thus in the world, there are trees, tables, cars, people, clouds, rugby games, etc. We certainly can talk about these in language, and when we do so we use *nouns* or *noun phrases* to do so. And the second kind of item is true or false *statements* or *propositions*. One way statements arise is when we ascribe properties to the things. We might say, or think, or envisage, or know, or believe, that a particular tree is green or that people are kind or that the rugby final was exciting. Our ascriptions of properties to things constitute, or result in, statements or propositions. And when we talk about true or false statements or propositions in language, we use *sentences* to do so.

As hinted above, there is some ‘mortar’ or ‘construction material’ that allows entities to take part in statements: we described it as being ‘the ascription of properties to the things’. So, for example, we start with the rugby final and apply the ‘was exciting’ paste or paint to it and get an ascription or statement (that the rugby final was exciting). This glue transformation step is paralleled in language. There are nouns or noun phrases, and we apply to these the glue of verbs, or the copula ‘is’, often mediated by adverbs or adjectives, or other complexities, and sentences are result.

Michel Foucault (1970) used to talk about ‘the theory of the verb’, and of noun and verb. Primitives can utter sounds, ‘cat’, ‘tree’, ‘man’ and mean thereby some of the entities immediately to hand (‘mean’ in the sense of ‘designate’ or ‘refer to’ or ‘draw attention to’). These noise nouns give us some of the things in the present.

But once verbs are added, for example, the verb ‘springs’ to form ‘the cat springs’, affirmation is possible, and statements and propositions result. Adding the copula ‘is’, tenses, adjectives, adverbs, determiners, nouns, etc. to the verbs allows the propositions to be richer. But the overall conceptualization is still: things and property, or function, ‘mortal’, yield statements.

The Greek philosophers discussed substances (the things) and properties that the substances might have, and the resulting combinations, in thought or expression, of substances with properties, or subjects with predicates, were the statements or assertions or propositions. This basic distinction, and combination of the distinct kinds, is seen nowadays in many theories. Categorical and Generative Grammars of natural languages use it (Heim and Kratzer 1998; Partee et al. 1990). And in First Order Symbolic logic, to be visited later, there are just two kinds of basic expressions: ‘terms’, which name things, and ‘formulas or well-formed-formulas’, which are the logical representation, or names, of statements.

George Bealer describes this

What function does the subject/predicate distinction have? First, in speech the distinction shows up as follows. A subject expression is the kind of expression that functions to identify a thing about which something is to be said. A predicate expression, by contrast, functions to say something about things so identified. ...subjects fix the subject matter, and predicates (verbs) do the saying. Secondly, the subject/predicate distinction plays a role in syntax.... The definition of [an atomic] sentence is roughly this: subjects combine with predicates to form sentences.... Thirdly, the subject/predicate distinction plays the role in the construction of a natural economical semantics that tallies with the intuitive concept of meaning. (Bealer 1982, p. 86)

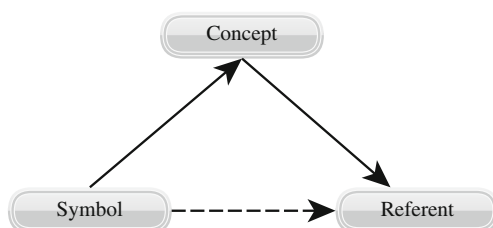
As we have just seen in the previous section, entities might be concrete or abstract. Within this classification, statements themselves are abstract, they do not have location. When a gardener asserts that the tree in the garden is green, he may say or write sentences to that effect on many different occasions—all of those sentences or utterances are concrete. But what he asserts—that the tree is green—is abstract. Of course, we can treat the assertions or statements as things, then they would be abstract things, and we can make further statements about them. We regularly do that. Imagine a court of law. When the accused protests that the insinuations of a witness are false, the accused is treating statements as things and then making further statements about those statements.

The distinction between entity and statement has immediate application in librarianship as it underpins the distinction between the retrieval of documents and the retrieval of information. A document is an entity, (usually a concrete entity); information is, or consists of, one or more statements (and they are abstract).

2.5 The Triangle of Meaning

Another distinction that will be useful is that arising from the ‘Triangle of Meaning’ (Fig. 2.1). The ‘Triangle of Meaning’ is a phrase originating with Ogden and Richards *The Meaning of Meaning* in the 1920s (Ogden and Richards 1972).

Fig. 2.1 The triangle of meaning



Shiyali Ranganathan (1937, p. 327), the pre-eminent modern theorist of librarianship, also used the distinction in 1937; he called the Symbol Vertex the ‘verbal plane’ and the Concept Vertex the ‘idea plane’. Actually, the distinction goes back at least to Aristotle (*De Interpretatione*) and it was discussed by the aforementioned early twentieth century German philosopher Frege (Tichy 1988).

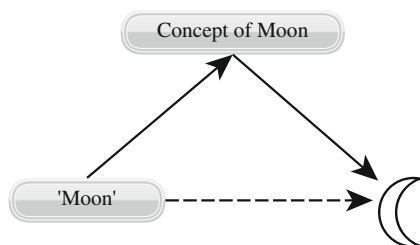
Consider written or spoken words; for example, the word ‘moon’. What does ‘moon’ mean? The word ‘moon’ does not and cannot mean that actual physical object in the night sky, for the following reason. Suppose the moon disappeared or was destroyed (perhaps one of those NASA water seeking rockets blows it up). If the word ‘moon’ meant that physical object, then, when the physical object disappeared, the meaning of ‘moon’ would disappear as well; the word ‘moon’ would not have a meaning. That is obviously wrong. The word ‘moon’ would still have a meaning; it is just that the word would not refer to anything if there were no moon. We see much the same with words like ‘unicorn’; ‘unicorn’ is a perfectly good word, and it has a perfectly good meaning; even though no unicorns exist.

To understand what ‘moon’ means, we need the notion of a concept (Fig. 2.2). The Triangle of Meaning makes distinctions, first between a concept and an expression or symbol or sign that names or identifies the concept, and then between the concept and the things it applies to or refers to. So, there might be the word ‘horse’, the concept of horse, and those particular delightful creatures which fall under that concept, for example, Secretariat, Sea Biscuit, Little Sorel, Trigger, Silver, Black Beauty, etc. The terminology commonly used here is: a property, like being a horse, is an *intension* or *connotation*, and those things in the world which satisfy a property, the horses, say, form the *extension* or *denotation* of that property.

So, what is a concept? In this context, the word ‘concept’ gets used in pretty well the same way as in ordinary speech and life, and that amounts roughly to ‘general notion’ or ‘general idea’ or even ‘meaning’. Many describe concepts as being mental or mental constructions; however, we regard them as abstractions or abstract objects (in the standard Fregean third realm).

A concept can have, and, in fact, usually does have, more than one word or phrase that names it. That occurs in the case of *synonyms*. The words ‘butterfly’ and ‘rhopalocera’ are synonyms, they have the same meaning—they are different words but they identify the same underlying concept. It also occurs with translations across languages; the French ‘papillon’, Japanese ‘choo choo’, German ‘schmetterling’, Welsh ‘pili pala’ are also all words or phrases that identify the butterfly-concept.

Fig. 2.2 The triangle of meaning for ‘moon’



Also, and also unfortunately, some words for concepts are *homographs*. An individual homograph, on its own, is ambiguous in that it can name different concepts. Consider the word ‘bank’. It is a homograph. There is

‘bank’, the noun, meaning ‘the place that looks after money’
 ‘bank’, the noun, meaning ‘the place at the side of a river’
 ‘bank’, the verb, meaning ‘to lean over’
 ‘bank’, the verb, meaning ‘to rely upon’
 etc.

So, there are banks to rivers, banks with cash in them, and banks that airplane aerobatics teams do. The one label ‘bank’ is being used to label three or more entirely different concepts.

Two different concepts, two different intensions, can have the same extension (when this occurs, that they have the same extension does not mean that they are the same concept). Imagine this. Say everything in our world that was red was also round, and also that everything that was round was red. So there are two concepts here ‘red’ and ‘round’, and the red things are {a, b, c, etc. say} and the round things are also {a, b, c, etc. say}. But red is a color, and round is a shape, and it just so happens that in our world their extensions are the same.

How do we apply an intension to get the extension? How do we determine just what a particular intension is? These are not easy questions to answer. It is not even easy to understand quite what is being sought. Are we doing logic here? Or psychology? Or methodology? Certainly, even young children can apply simple and basic concepts pretty well. Then the interest becomes how is a person to manage with learning and applying a new and unfamiliar concept, say the concept of a rose. There are theories [including Rosch’s Prototype Theory and the Classical Theory of Categories (Rosch 1973, 1975, 1978; Rosch and Mervis 1975; Rosch et al. 1976; Jacob 2004; Taylor 2004)]. We will just take it as an unanalyzed datum that we can work with concepts and their intensions and extensions.

A point to be aware of is that the Symbol Vertex is relatively accessible to computers, and that is where the strengths of computers lie. In contrast, the Concept Vertex plays to the abilities and strengths of human beings. So, for example, if we are analyzing a book, questions like, how many words are there in the book?, how many occurrence are there of the word ‘dog’?, and how many concordances are there of the words ‘dog’ and ‘owner’ in the same paragraphs? are

all Symbol Vertex questions and are trivial for computers to answer. In contrast, a question like ‘Is the book about betrayal and unrequited love?’ is a Concept Vertex question (in particular, it is certainly *not* asking whether the words ‘betrayal’, ‘unrequited’, and ‘love’ occur in the book). Humans can answer it, but here computers are in the realm of guesswork.

What we will find is that the important questions of information organization and retrieval involve the Concept Vertex. They need humans if they are to be answered. But the volume of information is such that humans cannot keep up with organizational questions and tasks. Thus the global task for organization of information is to try to get computers to do a task which they are not well suited for doing.

2.6 Identifier Mappings

We have an interest in the relationship between names, labels, or identifiers and those items that the identifiers identify. We saw with the Triangle of Meaning the issue of synonyms and homographs. With synonyms, several different labels pick out the same concept, so the relationship between synonyms and their concepts is many-to-one. With homographs, the same label names several different concepts, so the relationship between the name and the items named is one-to-many. And, for the third category, there are nouns which are neither synonyms nor homographs, and the relation between them and the concepts they name is one-to-one.

The many-to-one mapping kind is common in information retrieval. We see it with aliases, pen-names, and the like. The author Samuel Clemens wrote under the pen-name of Mark Twain; so the names Samuel Clemens and Mark Twain each named exactly one person, indeed the same person, but the person that they named had, so-to-speak, two, or maybe more, names. There are also what might be characterized as spelling variations of the same name. Works in the Library of Congress have about 50 versions of the name for the 2011 President of Libya, including

Qaddafi, Muammar
Gadhafi, Mo ammar
Kaddafi, Muammar
Qadhafi, Muammar
El Kadhafi, Moammar

[See (Mann 1993, p. 26).] The familiar URLs (Uniform Resource Locators) on the Web, for example,

<http://www.msn.com>

are a many-to-one scheme. It could be, and often is, the case that two different URLs name the same page. Moreover, it is often desirable to do exactly that. ‘www.msn.com’ is the address for the home page of MSN (News). But MSN will have bought a number of similar names, including perhaps ‘www.msn.org’ and use all of them as names of that same home page. The reason is that Users might

not know the exact address and so MSN wants to help them with close variants. (There are also other good reasons for allowing many-to-one naming. For example, different contexts, that is different audiences, experts as contrasted with children, or different languages, French or English, may invite the use of different names for the same page or resource such as a list of addresses and phone numbers.)

We are going to assume the every Information Object (IO) has, or can be given, at least one name or identifier that uniquely identifies it. So, conceptually, there is IO #1, IO #2, IO # 3 etc. This assumption lets us know which IO we are talking about. We know this assumption can be met. There is a generalization of the URL naming system and that is the URI (Uniform Resource Identifier) naming system—the generalization is that URIs are suitable for naming anything, any ‘resource’ (and anything includes web pages). There are enough URIs for all resources. URI naming is many-to-one. (There is a subtlety that should be mentioned. An URL names a web page or file, and web pages can change. For example, the URL <http://www.msn.com> is a news page and likely yesterday’s news is different from today’s. So URLs are naming ‘containers’ yet we may have reason to name ‘content’ so that yesterday’s news is a different IO to the IO which is today’s news. There are complementary naming schemes to the URI system, such as Digital Object Identifiers (DOIs) and the Handle System, which can do exactly this. So all IOs, containers and content, can have their own names, as assumed.)

At a theoretical level, one-to-one schemes seem the cleanest and most incisive. But, actually, for the purpose of naming IOs, many-to-one schemes seem more useful and more in tune with practical reality.

With IOs, there is always the question of granularity or what constitutes a ‘documentary unit’ (Anderson 1997). For some purposes, a book as a whole might be an IO, for other purposes, Chapters, Sections, Pages, Paragraphs, and even Sentences, in that very same book might all be IOs. However, whatever the IOs are, we are going to assume we have names for them, that we can identify them.

(Even one-to-many schemes can have their uses, though perhaps not in information retrieval. There is the apocryphal senior academic who was proud of the fact that he knew and remembered the names of all the students that he had taught during his long career of 40 years or more—all the male students were ‘Mate’ and the females ‘Honey’.)

2.7 Graph Theory

Graph theory will be useful both for discussing the links and structure that underlie the World Wide Web (and citation webs) and for providing the theoretical basis for trees and hierarchies which are the foundations of classification systems. They also can be used, in a tricked out way, via Resource Description Framework (RDF), for representing statements in logic, and then using those statements in the Web, particularly the Semantic Web.

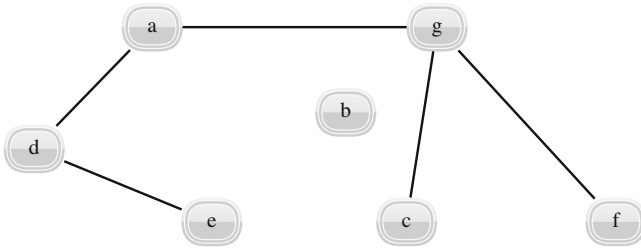
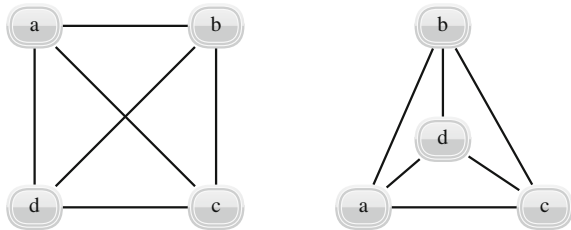


Fig. 2.3 An example of a graph

Fig. 2.4 Two visually different presentations of the same underlying graph



Graphs consist of two things: nodes (other words for these: ‘vertices’, ‘points’) and links (other words for these: ‘edges’, ‘lines’, ‘ties’). In applications, nodes typically model individual people, countries, businesses, web pages, genes, amino acids, etc. Links typically model friendships, web hyperlinks, telephone lines, communications, etc. A node can be thought of as being a dot or a point, and a link as being a line or an edge which runs from a node to a node. So here is a graph (Fig. 2.3).

The *size* of a graph is the number of nodes in it. This graph is of size 7. Graphs can have the same size but different link structure. The graphs of interest to us will all be *finite* graphs; at most they will have finitely many nodes in them. The *degree* of a node is the number of links that connect to it. So, in the example, three of the nodes have degree 1, but nodes a and d have degree 2, g has degree 3, and b degree 0.

How the links are drawn is irrelevant (they can be drawn as curves or straight lines, and the links can cross each other without that meaning anything), and where the nodes are located is also irrelevant—all that matters is which nodes are connected by which links. These two graphs are actually the same graph (Fig. 2.4).

A *walk* is an alternating sequence of nodes and links, which starts and ends with a node. So, in first example graph, $a \rightarrow e \rightarrow b$ is a walk. A *path* is a walk with no repeated nodes. The *length* of a walk is the number of links included in it. Two nodes are *adjacent* if there is a link between them. A graph has a *cycle*, or is *cyclic*, if there is a walk of non-zero length that starts and ends on the same node (otherwise the graph is said to be *acyclic*). A graph is *connected* if there is a path between every two nodes in it. The example graph is connected.

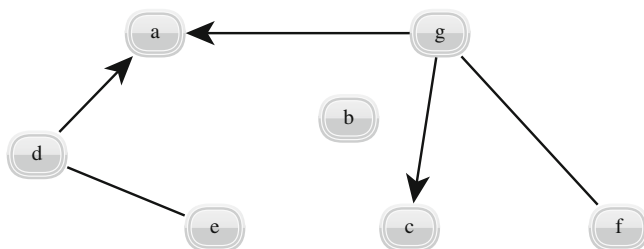


Fig. 2.5 An example graph with some directed links and some links which are not directed

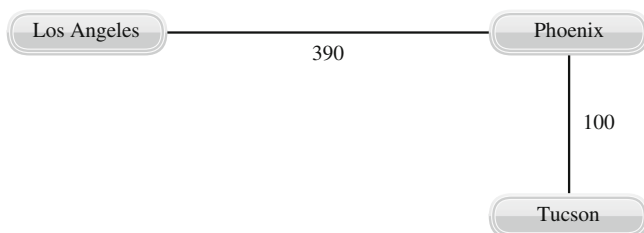
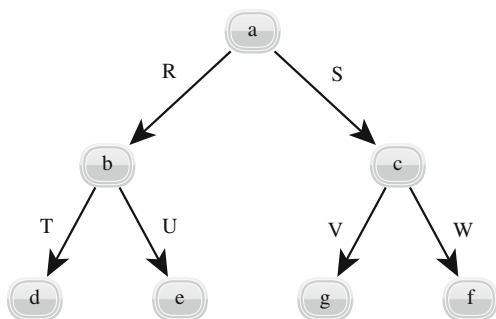


Fig. 2.6 A schematic road map shown as a fragment of a labeled graph

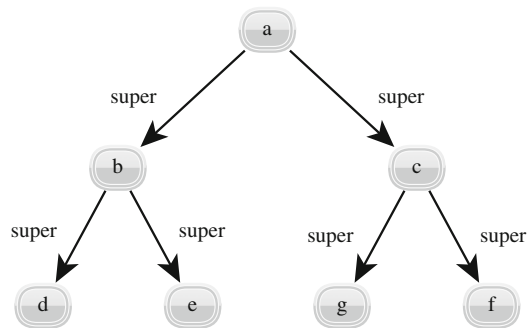
Fig. 2.7 An example graph which is labeled and directed



Links can be *directed*. The following graph has some directed links (from d to a, from g to a, and from g to c—shown by arrows) and some links which are not directed (Fig. 2.5). Having directed links invites an update of some of the notions. *Indegree* is the number of links coming into a node and *outdegree* is the number of links going out. And the notions of path and walk can also be revised depending on whether the traveling is required to follow the direction of the links. Then, for example, a graph would have a *directed cycle* if there is a walk of non-zero length that starts and ends on the same node and which follows the direction of the links.

Links can also be *labeled*. This just means that the links have labels on them. This is useful, for example, with schematic road maps. The labels could represent the distances between the places that are the nodes on the graph (Fig. 2.6).

Fig. 2.8 A labeled directed graph depicting a organization chart



2.7.1 Triple Stores

Of particular interest to us will be graphs that are both directed and labeled. Here is a generic diagram (Fig. 2.7). This might be a fragment of an ‘org chart (organization chart)’ in an institution. In which case, the labels might all be the same, and they might represent the ‘supervises’ relation. Thus (Fig. 2.8). There is flexibility here over what a labeled directed graph can do (pretty well everything!).

There are many ways of encoding graphs into computers—the use of matrices is common and widespread. But of particular interest to us are *triple stores*. One way of representing two nodes and a labeled directed link between them, is just to write it as an ordered triple;

$\langle \text{firstnode}, \text{label}, \text{secondnode} \rangle$

so $\langle a, \text{super}, b \rangle$ represents the top left supervision labeled link. Then the entire of the above org chart is

$\langle a, \text{super}, b \rangle,$
 $\langle b, \text{super}, d \rangle,$
 $\langle b, \text{super}, e \rangle,$
 $\langle a, \text{super}, c \rangle,$
 $\langle c, \text{super}, f \rangle,$
 $\langle f, \text{super}, g \rangle$

This is a *triple store*. Its contents are triples, and they represent an ordered labeled graph. The order of items within a triple matters, $\langle a, \text{super}, b \rangle$ is different from $\langle b, \text{super}, a \rangle$. But the order of the triples themselves within the store does not matter, it is irrelevant. For example,

$\langle c, \text{super}, f \rangle,$
 $\langle b, \text{super}, d \rangle,$
 $\langle b, \text{super}, e \rangle,$
 $\langle a, \text{super}, b \rangle,$
 $\langle a, \text{super}, c \rangle,$
 $\langle f, \text{super}, g \rangle$

is the same graph. Duplicate triples are also irrelevant and extra ones should be discarded (within graph theory, you can have a labeled link between nodes, but not a second identical labeled link between the same two nodes). In the general case, a graph can have nodes that have no links to or from them (this is not common in our applications). A triple store would need to make an ad hoc convention to accommodate these (say `<a, noLinks, a>`).

Triple stores would probably not ordinarily be used if the label or relation was absolutely identical throughout the graph, simply because pairs could be used instead of triples. Usually a triple store, and its graph, would have a mix of relations. For example, with an investigation of communication channels within an organization there might be an interest both in who supervised whom and who was friends with whom (Fig. 2.9).

And that would be

```
<c, super, f>,
<b, super, d>,
<b, super, e>,
<a, super, b>,
<a, super, c>,
<f, super, g>,
<a, friend, d>,
<b, friend, c>,
<d, friend, e>,
<e, friend, d>
```

For every labeled directed graph there is a triple store that captures it entirely. But also this can be used the other way around. If there is some data from elsewhere that is in the form of a triple store [and all data can be cast that way] that triple store can be regarded as a generator of a directed labeled graph.

A number of large triple stores have been implemented (W3C 2010); for example, *Open Link Virtuoso*, 2010, a single triple store, contains more than 15 billion triples.

2.8 Elementary Data Structures and Their Algorithms

One field of study in Computer Science is Data Structures, or Abstract Data Structures. In this, certain mathematical structures (lists, stacks, arrays, queues, dequeues, trees, and others) are defined in terms of objects and operations on those objects. The operations typically include inserting, deleting, updating or retrieving the objects to and from the relevant data structures. Then the computer science discussion moves to algorithms, and languages, to support the structures and operations. This has proved to be a particularly fruitful style of analysis. The abstract data structures are set at a level above the implementing algorithms and computer programming languages. They are a commonality that ranges across many working implementations and programs.

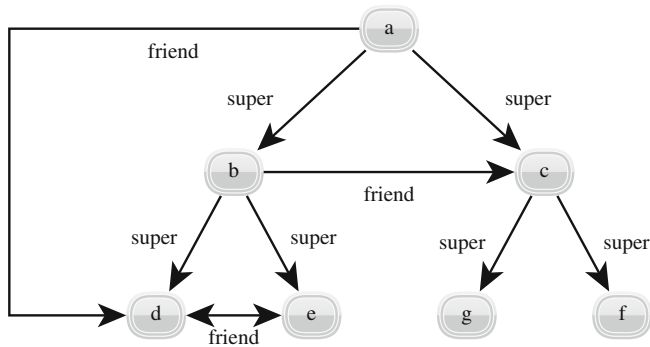


Fig. 2.9 A labeled directed graph depicting both supervision and friendship relations

Something similar will prove useful here. We too are interested in certain Abstract Data Structures. But for us the operations to focus on generally are not those of inserting, deleting and updating objects, rather they are often to do with accessing the objects in the data structures. Now, ‘accessing’ an object may simply be looking at it, so that presupposes a displayed instantiation or implementation of the abstract data structure. ‘Accessing’ may be a little more complicated than plain looking, it may require a fancy front-end or a retrieval tool or the right moves on the part of a human agent. Access is partly to do with the structures, partly to do with algorithms and partly to do with human cognitive abilities.

2.8.1 Lists and Strings

We are all familiar with lists (shopping lists, bullet point lists, numbered item lists in html and web pages etc.). Considered in the abstract, a list is a sequence, or total order, of items, where the items have a position (first item, second item, and so on; and each item, except the first and last, has a previous or predecessor item and a next or successor item).

There are differences in the convenience of manipulations between human produced lists, say a written list on a sheet of paper, and lists within computers. With a human list, it is easy to grow it by writing items on the end; it is not so easy to shrink it by taking items out (but that can be done by erasing items or striking through them); it is especially difficult to insert into the middle of a list, for that requires either squeezing items in or leaving gaps in the initial list to take the few new items (and that needs guesses as to where the gaps should be). With lists within a computer, growing, shrinking, and insertion anywhere are readily available.

As a special example of a list: lists—or sequences—of characters are known as ‘strings’ of characters so ordinary words like ‘Hello’ and ‘today’ are regarded as strings, as are nonsense strings like ‘bod!12H’.

2.8.2 *The Alphabetical List: The ‘Wheel’ of Information Retrieval*

A list of words or strings, ordered, or sorted, alphabetically (i.e. in so-called ‘filing order’) has wonderful properties. What can be done with alphabetical lists by humans is truly astonishing [And all of us would be indeed be astonished, but for the fact that we are so familiar with it.]

Say we have list of words,

Banana,
Cabbage,
Apple,
Cauliflower,
Date,
...

And we are interested in whether a supplied word, or key, say ‘Plum’ is in the list. There are really only two ways to find out; to start at the beginning and look through the list forward, or to start at the end and look through the list backward. How well we will do with this will depend on how large the list is, and where ‘Plum’ is (if, indeed, it is there at all). The list certainly has an order, ‘Banana’ is first, ‘Cabbage’ second, and so on; but the order it has is independent of the values or the words taking those positions (i.e. ‘Plum’) could be anywhere.

Consider now the same words, but in an alphabetical list.

Apple,
Banana,
Cabbage,
Cauliflower,
Date, ...

and the same problem of finding ‘Plum’. This time, we do not have to start at the beginning or end and do a total comprehensive search. Instead we can jump in the middle somewhere and, with a quick scan back and forth, find ‘Plum’ if it is there. The list has an order, ‘Apple’ is first, ‘Banana’ second, and so on; and the *absolute* position an item has is still independent of the values of the words taking those positions (i.e. ‘Plum’) could be anywhere. But the *relative* positions of entries *does* depend on the values of the items ‘Plum’ is always going to appear later than ‘Cabbage’ and earlier than ‘Quince’, if all those words are there. And that relative position feature makes all the difference.

For retrieval, an alphabetical list of words is something special. Imagine paper copy volumes of a New York telephone book of 7 million entries. If the names in the book are in alphabetical order, even a child can do name retrieval in moments. If the names in it are there in random order, basically they are not findable by ordinary human search.

Alphabetical order is magic, as Sherman Kuhn phrases it,

ALPHABETIZATION seems as indispensable to twentieth century life as the wheel; and, like the wheel, it was invented by some unknown genius of remote antiquity, who probably never realized the importance of his gift to mankind. Today, libraries, telephone communications, much of commerce and industry, even the lower levels of government, would be paralyzed if the art of alphabetization, and the tools which it has produced, were suddenly lost. (Kuhn 1972, p. 300)

Historically, lists of words in alphabetical order were slow to appear. There are three reasons for this: it was unclear what alphabetical order itself actually was; second, until the advent of association lists, there was no imperative to produce alphabetical lists; and, finally, it is hard to produce such lists by hand, without physical paper, or something similar, being readily available. Alphabets themselves, and alphabetical order on words were often ill-defined. In many of the alphabets of history, new letters appeared, old letters disappeared, and the order was sometimes changed (although change of order was not common). Then, even if alphabetical order by the first letters of words was understood, that this needed to be used in conjunction with alphabetical order on the second, third, and further letters of the words (to produce a total order in the list itself) was just beyond comprehension. Also, the need for alphabetical lists was unclear. What really shows the advantages of alphabetical lists are ‘association lists’ (to be discussed shortly) and historically these are tied to indexes, which are also relatively recent. Indexes themselves are connected with pagination of printed books, and printed books are the outcome of the Gutenberg’s printing press, circa 1450CE. Finally, actually producing an alphabetical list is difficult. Say you were given a random list of a 100 words, written on a page, and asked to produce a list of the same words that was in alphabetical order. How would you do it? You could scan through, find what you thought was the first alphabetical entry, write that down in a new list, cross it out of the old list, and repeat another 99 times. That technique would be error prone. It might work for a hundred words, but it would be under stress for a 1000 words. The right way to do this by hand is to write each of the words on its own separate piece of paper or card, and then sort those cards by your favorite sorting method [quicksort, bubble sort etc. (Sedgewick 1988)] then write the result of the sorted cards back to another sheet of paper. But paper itself (i.e. cheap and plentiful material to write on) was not produced mechanically until about 1200 (and not that many people could write). So, the technique of writing onto cards and sorting those cards was not really even a possibility until the fourteenth century or so. [By 1545, Conrad Gessner was able to describe a cut-and-paste technique in his *Bibliotheca Universalis* (Wellisch 1981; Blair 2003).] It would have been easy to produce short alphabetical lists, say with 5 or 10 entries, but short alphabetical lists do not really have any advantages over short random lists, for they are both easily scanned and accessed by eye. Shermann Kuhn describes early efforts

There is no steady advance from no alphabetization to first letter to second and on to full alphabetization. Glossaries organized by categories appear side by side in time with glossaries in some type of alphabetic order, and chronological ledgers and records side by side with ones which are in alphabetic order or no order at all. Full alphabetization is discovered, then abandoned for simpler systems, recovered or rediscovered, abandoned again, recovered again (Kuhn 1972, p. 302).

In sum, it took until about 1300CE before people were aware of what it was to put a list of words into alphabetical order (Daly 1967), and it is not until 17-1800CE that such lists are common.

2.8.3 Association Lists

An association list contains pairs of items, not single items. It consists of a list of key-value pairs, so, for example, it might consist of

Smith 471,
Jones 134,
Mercy 32,
etc.

The ‘Smith’, ‘Jones’, ‘Mercy’, are the *keys*, and ‘471’, ‘134’, ‘32’, the *values*. And, conceptually and algorithmically, the list is required to have the ability to return the value of a key-value pair when it is supplied with the key. So, asking the question ‘(value of) Jones?’ should result in ‘134’. To do this, each of the keys needs to be unique (i.e. ‘Jones’ cannot be there more than once because then the relevant value for the key ‘Jones’ might be ambiguous). (Well, there is a slight qualification to this. Most implementations of association lists do permit duplicate entries. However, the retrieval algorithm is designed to scan the list in order from the beginning and to return the *first* value encountered for any particular input key. And ‘first’ is unique. Essentially the duplicate entries are junk on the end.) That the keys need to be unique means that any homographs should be disambiguated if there is any plan to use them as keys; so, instead of having two keys with the value ‘bank’, there should be one key perhaps with the value ‘bank (money)’ and another with the value ‘bank (river)’.

An association list can be thought of as a table with two columns (provided it is remembered that the keys in the key column have to be different one from another so that they can act as unique identifiers).

Within the theory of Data Structures, association lists are not usually deemed to have any order—particular key-value pairs can be anywhere in the list. But, for us, it is valuable to think of a certain class of association lists as being *sorted association lists*. Of particular significance are ordered association lists where the list is sorted alphabetically by key, thus

Jones 134,
Mercy 32,
Smith 471.

And this give us the familiar book indexes. So, for example, the index fragment

absurdity, rule of 137
accessibility relation 166

adequate: for truth-functional logic 44;
sets of connectives vii, 44ff
algorithm 17
alpha (α) sentence 49
alternative denial 44
ancestral trees vii, 6, 33ff.

is just a sorted association list. Another point worth remarking on is that the values in a book index are really pointers or references or directions to what is wanted. For example, in the above index fragment ‘algorithm’ has the value 17, but it is not really the number 17 that you are after, rather it is a particular page in the book, page 17 in fact, and the 17 is just a stepping stone to get there. The alphabetical Index is a microscope into the text itself. Other alphabetical association lists include telephone directories, library catalogs, the standard bureaucratic ‘papered’ office with its filing cabinets, and so forth. It is hard to over emphasize the importance of the alphabetical list and the alphabetical association list.

Alphabetical, or filing, order is certainly not the only useful order that can be used with ordered association lists. There are other possibilities. For example, a ‘To do’ list might usefully be produced with the entries in the order that the tasks have to be done, similarly for Schedules and Agendas. Day Planners would be in the order of the days; TV Guides would be in temporal order (although a second index of programs, inverted, and in alphabetical order, would help in finding, say, when Bonanza was on without having to search the time slots). Lists of finishers in Olympic events, or Marathons, would often need to be in the order the participants finish, not in alphabetical order (although to find where a particular competitor finished among the 35,000 participants in the Chicago Marathon a supplementary alphabetical index would be a distinct help). Often computers return lists in order of relevance or importance—that ordering can be valuable too.

2.8.4 Filing Order and Key Syntax

Alphabetical order is often known as ‘filing order’ (for obvious reasons). Actually defining a filing order requires some decisions, discipline and conventions. With a computer, there will an order defined on each character in the available character set in the programming language; so, for example, ‘a’ will come before ‘b’; and this total order applies to every character including all the punctuation and special characters. Unicode (‘... the most widely adopted software standard in the world...’) has a representation for 107,000 characters, which basically amounts to most every character in most every language, and it defines a total order for those characters (Unicode 1991–2010). This means that ‘alphabetical order’ is immediately available for all strings i.e. for all words (for example, place all strings of length one before all strings of length two etc. then order strings of the same length by character comparison of characters in the same position in the different strings). So, this is trivial and uninteresting intellectually.

Outside a computer, the task is more challenging. The reason is: the order needs to meet the expectations of the human Users. For example, the computer order of the last paragraph could produce an ordered list like

A,
B,
Ab.

And nobody would expect that as an alphabetical order (for example, in a phone book). They would be expecting all the A strings together, all the B strings together and so on. Thus

A,
Ab,
B.

Of course, computers do not have to produce the original computer ‘internal’ order that was described. But the programmers writing the software need to know what is wanted.

So, what do, or should, users expect? Actually, there are different schemes and different books of rules for those schemes. Here are some of the choices.

The main decision is: word-by-word or letter-by-letter. What turns on this is, for example, whether ‘New York’ files before ‘Newark’ or after it. Word-by-word is the more common and that would place ‘New York’ before ‘Newark’. Letter-by-letter places ‘New York’ after ‘Newark’. Here is some explanation. The blank character or space ‘ ’ is a character like any other. So the string ‘New York’ is actually the string ‘New’ followed by the blank character, followed by the string ‘York’. In word-by-word order, the blank character is left in, and the blank character is considered to come early in the alphabet, in fact, before any of the other real writable or printable letters. So, when ‘New York’ is compared with ‘Newark’ letter-by-letter left to right, they are the same for ‘N’, ‘e’, ‘w’ but then blank is compared with ‘a’ and blank comes earlier, so ‘New York’ files before ‘Newark’. Letter-by-letter is different. For the purposes of this comparison, the blanks or spaces are discarded completely, so what gets compared is ‘NewYork’ (note, no space) and ‘Newark’ and this time the fourth letter ‘Y’ comes after ‘a’ and so the order is ‘Newark’ before ‘New York’. Similar considerations occur for hyphens and some other punctuation (e.g. how to fit ‘New-York’ in). Generally word-by-word order is the conventionally preferred form. There is an argument in favor of letter-by-letter order. Compound nouns from double stems can often be written as one word, as two words, or as a hyphenated word (‘particle board’, ‘particle-board’, and ‘particleboard’). If all three forms ended up in the same index, dictionary, or file, then letter-by-letter order would put them next to each other (which is what you would want), word-by-word order would scatter them (which is what you do not want).

Word-by-word has one terrific advantage—if employed intelligently it can be used to give fine structure to the alphabetical lists or alphabetical association lists, that is: it can help determine where the key, or key-value pairs occur, and what is proximate to what. If an alphabetical list has just single word keys, then it is just an alphabetical list. If it has two word keys, it is not ‘just’ an alphabetical list, it is in effect embedded alphabetical lists nested to a depth of two. If there are three word

keys, there can be nesting to depth three, and so on. For example, say there are, perhaps as subjects for study, small, medium, and large animals which are fish, mammals, and crustaceans; if two word keys are used for these then placing the sizes first in the syntax of the keys gives, using word-by-word filing

```

large crustaceans
large fish
large mammals
medium crustaceans
medium fish
medium mammals
small crustaceans
small fish
small mammals

```

or, to use a better visual display,

```

large
  crustaceans
  fish
  mammals
medium
  crustaceans
  fish
  mammals
small
  crustaceans
  fish
  mammals

```

but placing the animal type before the size, in the key, gives

```

crustaceans
  large
  medium
  small
fish
  large
  medium
  small
mammals
  large
  medium
  small

```

so, there is grouping or fine structure here that can be designed and controlled. The ‘trick’ is to use multiple word keys. This technique could be used, for example, with a book, calling the keys ‘Chapter1 Section1’ etc., and then an alphabetical list ordered word-by-word would produce a structure similar to a Table of Contents. This fine structure is a matter of *key syntax* (coupled with word-by-word filing order).

There are other decisions. Typically, with most systems, numbers precede letters and English language precedes other languages. From then on it can get pretty complicated, with strings consisting of a mix of languages, with other punctuation (quotes, apostrophes, and dashes etc.) and hyphenated words. There also can be such subtleties as ‘nonfiling characters’. The librarian data carrier, the MARC record, has a field to set the ‘nonfiling characters’, and this is the number of leading characters from a string to ignore when filing; it typically has the value 2, 3, or 4; what this is able to do, for example, is to remove the leading ‘A’s, ‘An’s, and ‘The’s from titles when considering filing order; so a title like ‘The Pilgrim’s Progress’, with ‘nonfiling characters’ set to 4, is filed with other titles beginning with ‘P’ (from ‘Pilgrim’) and not with titles beginning with ‘T’ (from ‘The’). And plurals can be a problem: without special consideration, ‘mate’ likely will not file next to ‘mates’, they could have ‘maternity’ in between them. Folk that produce artifacts that depend on filing order, for example, phone books, spend some considerable time and effort specifying their rules. Unfortunately, there is no universal filing order used by everybody everywhere. As Jessica Milstead writes

Nearly all library catalogues and telephone books, and most book indexes, use word-by-word arrangement. Encyclopedia indexes are frequently arranged word-by-word but the text is usually arranged letter-by-letter. Dictionaries are nearly universally arranged letter-by-letter. Abstracting and indexing services vary, with firm exponents of both means of arrangement being easy to find. Thesauri, and the services which use them, are frequently arranged letter-by-letter (Milstead 1984, p. 48).

There is an entirely separate, and slightly subtle, issue about alphabetical lists. An alphabetical association list associates strings with values. But much of the time, in practice, that association is used merely as a component in a wider association between something else, perhaps a person, a thing, or a concept, and a value. Think again about phone books. They associate strings with numbers, phone numbers. But the User is not trying to find the phone number of a *string*. The User is trying to find the phone number of a person or business. There is a gap here. There are persons, institutions, businesses, etc., and these have names or labels, which are transformed into string keys, which represent or identify them. But there needs to be a controlled conventionality at the point of insertion of those strings into the list or else the User will not know what string represents the item sought. Say there is the person John Jones and another person Jane Smith; if the string key for one of them is the form <last name>, <first name> i.e. ‘Jones, John’ and the other in the form <first name> <last name> i.e. ‘Jane Smith’, we start to be in trouble. If there is not suitable discipline here, the alphabetical list, while perfect on filing order string retrieval, will not work as a retrieval device for what we want.

2.8.5 Precoordination and Postcoordination

There is the consideration of *coordination* or *coordinate indexing*, in particular *precoordination* versus *postcoordination*. These ideas really come from Mortimer Taube, in the 1950s (Taube and Thompson 1951; Taube 1961, 1953, 1951), and they appear in a number of different settings in librarianship and with association lists. [Dagobert Soergel (1974) suggests the alternative terminology of ‘precombination’ for ‘precoordination’ and ‘postcombination’ for ‘postcoordination’ because those labels are more accurate to, and evocative of, the underlying operations. He is exactly right. Unfortunately the field of information science has ignored this excellent suggestion.]

An example from the standard use of Web Search Engines, especially Boolean search, can illustrate precoordination and postcoordination. The keys here are the text that is typed into the Search box, and the values are the lists of pages that are returned. Say our interest is in the astronomy and cosmology of stars; and, in reality, stars include ‘white dwarfs’, ‘red dwarfs’, ‘yellow dwarfs’, ‘brown dwarfs’, ‘red giants’, ‘blue giants’ and a variety of others. Initially, we might type in a series of one word keys such as

```
dwarf
giant
```

and the Search Engine will produce lists of web pages as results. What is going on behind the scenes here is that the Search Engine will have indexed in advance as much of the Web as it can. And it will do so for keys that people are likely to ask for (and that certainly would include ‘dwarf’ and ‘giant’). It will look up the relevant pages in its indexes and return them. However, in these queries, likely there will be a problem with homographs. Dwarf stars are not the only kind of dwarves there are, there are also dwarf dwarves (like the ones that accompany Snow White). And giant stars are not the only kind of giants there are; there is Goliath, a baseball team, and a film with Elizabeth Taylor as an acting lead. So the pages returned to us may contain irrelevant material. No doubt we could tidy up our request by moving to two words, and by asking about

```
dwarf star
giant star
```

What the Search Engine does behind the scenes will be similar to the first time around but it may or may not be exactly the same. In its own internal indexes, the Engine may have indexed in advance at least some two word keys, including perhaps ‘dwarf star’ and ‘giant star’. If so, it would just look up the relevant values and return them. This technique is that of *precoordination* or *precombination* (so-called because the words in the composite key are put together into the complex prior to the indexing and the determination of values for that key). However, it could be that the Engine works in a different way. It might *not* have internal indexes for two word keys, or it might *not*

have values for the keys ‘dwarf star’ and ‘giant star’, in which case, it would proceed using what it does have which, we may suppose, includes entries for the single word keys ‘dwarf’ and, separately, ‘star’; it would find all the values for ‘dwarf’, say {a, b, c, d, e} and all the values for ‘star’, say {d, e, f, g} and form the set theoretical intersection of the two sets of values to get {d, e} as the sought for result. This is to perform Boolean, or set theoretic, operations on the results of index retrieval after the fact. This is to *postcoordinate* the keys and values (Taube and Thompson 1951). There is no indexed entry for ‘dwarf star’ until the query is made, the two component words are post combined into the composite key. With real search engines, likely strings such as ‘dwarf star’ and ‘giant star’ would be precoordinated. However, we could ask about three word strings such as

white dwarf star
red giant star

and so on. There are so many possibilities as the strings get longer that eventually any engine would be swamped (if there are 40,000 indexed word, i.e. strings of length 1, there will be approximately $(40,000 \times 40,000)$ strings of length 2, and approximately $(40,000 \times 40,000 \times 40,000)$ strings of length 3, and so on). Eventually there has to be some postcoordination (or plain abandonment of the task). [Postcoordination, with search engines, is exactly the same as Boolean AND searches i.e. a postcoordinate search for ‘dwarf star’ effectively is a Boolean search for (dwarf AND star).]

Precoordination will always give better precision and recall, because it pays attention to the order of words in composite strings. For example, precoordination can separate index values for ‘Venetian blind’ from those for ‘blind Venetian’ whereas postcoordination on ‘Venetian’ and ‘blind’ will not do that. However, longer composite key strings face two problems. As shown above, the combinatory explosion means that eventually there are just too many keys. And, secondly, it can be difficult for the User to find the composite keys in an index, especially if the index is on paper. Imagine the engineers to the Three Mile Island nuclear accident wondering whether the proper string index entry to the manual was ‘significance of alarms to incipient uncontrolled thermonuclear runaway’, or ‘alarms, significance to incipient uncontrolled thermonuclear runaway’, or ‘incipient uncontrolled thermonuclear runaway, significance of alarms’, or what. This issue of controlled conventionality with the form of the string keys can be considered as another concern with key syntax.

There is, or was, an area of research, *string indexing*, given to the question of how to present multiword keys like ‘significance of alarms to incipient uncontrolled thermonuclear runaway’ so that they can be found (Craven 1986; Svenonius 1978). Basically, before the computer, there were three ways. The key could be put in the index just once. Then there would be so-called ‘citation order’ questions as to what the order of those multiword keys should be (e.g. natural word order, nouns then verbs or qualifiers, concrete to abstract, etc.). Second, variants of the key could be put in to the index as additional entries. So, for example, all of ‘significance of alarms to incipient uncontrolled thermonuclear

runaway’, ‘alarms, significance to incipient uncontrolled thermonuclear runaway’, and ‘incipient uncontrolled thermonuclear runaway, significance of alarms’, and many others, might be there. This makes the index much larger and creates noise. For example, Timothy Craven reports that one of the authority lists of the PRECIS system has a 100 entries starting with the word ‘acquisition’ (Craven 1986, p. 101). Such an index is not easy to use. Third, parts of the key could be put in to the index as additional entries. So, for example, not only would there be ‘significance of alarms to incipient uncontrolled thermonuclear runaway’ as an entire key but also there might be entries for fragments of it such as ‘uncontrolled thermonuclear runaway’, ‘alarms’, etc. This also makes the index much larger and creates noise.

This kind of string indexing research, and its resultant proposals and technologies, does not have the urgency it once had. It used to be that published indexes were to be on paper. But nowadays the indexes are electronic, or available in electronic or digital formats. And string-in-string search is available for the keywords or phrases within the actual keys of any indexes. So, for example, if ‘significance of alarms to incipient uncontrolled thermonuclear runaway’ was an index entry, there would be no need whatsoever to have any parts of it as additional entries in the index simply because the User could do a search for, say, ‘uncontrolled thermonuclear runaway’ and the software (string-in-string search) would find all the keys containing that phrase.

The Library of Congress Subject Headings (LCSH) are an interesting example. To most intents and purposes, there are 320,000 of these ‘headings’ (or keys) and they are precoordinated. Effectively, these headings can be used as keys in subject indexes and a recent Library of Congress Review concludes about the precoordination of their headings

... pre-coordinated strings provide context, which is needed for “disambiguation, suggestibility, and precision” and browsability. Pre-coordinated strings have a sophisticated syntax that can express concepts better than single words, yet also can be faceted by systems to group topics into categories for post-coordinated displays when desirable. Post-coordinated terms have serious limitations for recall, precision, understanding, and relevance ranking. (Cataloging 2007, p. 1)

All of which is correct, but it does not tell the whole story. We would expect that ordinary people would not be that accomplished at finding the headings suited to their needs among the 320,000. And, indeed, a few years ago, when the lists of LCSH entries used to be principally on paper, the headings were not that useful to Users. However, the LCSH list is now available electronically. But, as Lois Mai Chan points out in the same report, Users usually do not search LCSH lists using the actual LCSH string headings. Instead, Users often will keyword search, and that search will search among the headings. So, for example, ‘Tea making paraphernalia in art’ is an actual LCSH precoordinated heading i.e. a potential index key. This heading, and its associated heading-values indexes, certainly facilitates precision and recall for those searching for IOs on tea making paraphernalia in art. However, without the use of computers, many Users would not be able to find the

actual heading because likely they would not know whether the heading string was ‘Tea making paraphernalia in art’ or ‘Art, tea making paraphernalia in’ or ‘Tea pots depicted in art’ or what. With computers, many Users would be able to reach the heading string. But they would not do so by searching for ‘Tea making paraphernalia in art’, because of the reasons given above i.e. they do not know the requisite heading string, instead they would search for keyword combinations like ‘tea’ AND ‘art’ among the headings. And the keyword search would narrow to the sought for heading. So the process really would be a postcoordinated search among precoordinated headings. The precoordination takes place behind the scenes, largely hidden from the User.

When Taube first devised postcoordination, he envisaged a uniterm system where the keys were single words (i.e. strings of word length one) and the postcoordination was carried out on those single words. That kind of system will suffer from precision and recall problems. It will be improved by having some precoordinated keys which consisted of two or three word strings (i.e. strings of word length two or three, for example, ‘school library’, ‘library school’, ‘birth control’, etc.). However, longer and longer multi-word keys introduce problems of their own.

In sum, usually some precoordination will be good, but it should be limited and possibly supplemented with some postcoordination.

2.8.6 Hashing and Key Jumping

‘Hashing’ is an algorithmic technique devised in the 1950s, to help with, and speed up, the finding of items in a data structure. It is an intelligent generalization of alphabetical lists or ordinary everyday office filing. Say we have a school, with children who bring their lunches, and a rack of 26 pigeonholes or cubbies that are going to hold those lunches. We could let the children put their lunches anywhere, in which case the lunchroom helper might need to look through all 26 cubbies to find a particular lunch, say Charley’s lunch. Alternatively, we could label the cubbies alphabetically ‘A’–‘Z’ and use a system where the children put their lunches in the appropriate cubby suggested by their name. With this, a lunchroom helper can find Charley’s lunch without having to look through all the cubbies—the helper can jump to the right place merely by doing a calculation on the key ‘Charley’ and using that to get the correct cubby. Full blown hashing is a little more elaborate than this, but the basic idea is the same; namely, a calculation can be done on what you are looking for, its key or entry or string value, and that will tell you where it is likely to be and where you are likely to find it.

Hashing may not sound terribly important with lunches and cubbies. But if we scale up the problem to, say, a few 100 million people and their social security records keyed by social security number, hashing can make the difference between data processing being viable or not.

In traditional data structure terms, hashing is slightly different to retrieving from an alphabetically ordered association list. A list can be of any size, and it can grow

and shrink; a particular item can have any absolute position; there is no interest in determining absolute position, and alphabetical retrieval uses the key in conjunction, with other keys and the relative values of the keys themselves. Hashing is normally done in the context of arrays, which are fixed size data structures (cubbies of fixed number), and a calculation on the key is used to determine the absolute position in the array. So, there are differences, but both use information available from the key itself to suggest where to find it.

There are sufficient similarities to accessing an alphabetical list and using hashing that it would be convenient to have a label to cover both. Let us coin and use the term ‘key jumping’ to cover the use of features of a key as a heuristic to discover where it is in a data structure.

2.8.7 Metadata

Returning for one moment to IOs, in addition to one of more identifiers (i.e. names or labels), each IO will have many properties of interest for information retrieval. Almost all of these properties can be conceived of as *metadata* about the IO or IOs. Say we have a copy of the book *Geography for Younger Readers* by ‘Jane Smith’. This book is an Information Object; and, indeed, it contains information or data; for example, data about the longest rivers, the highest mountains, etc. In addition to the data that the book contains or conveys, there is other data, which is data about the book itself—this is the metadata. For example, the author’s name, the publisher’s name, the publication date, the edition, the subject matter, and the like—all of these items are metadata.

Conceptually, metadata is merely an association list of key-value (or field-value) pairs for each individual IO. So, for example, metadata for a particular IO, say IO with ID# = 97, might be

Field (Key)	Value
Author	Smith, Jane
Title	Geography for Younger Readers
Date	2011
Subject	Geography

Metadata is infrastructure for the organization of IOs. How do you arrange or organize IOs? You give those information objects metadata, preferably using metadata fields whose values can be determined easily (i.e. by viable algorithm on the information objects).

There is the question of whether the number of metadata fields that there are within a particular conventional system is fixed or whether the number of fields is arbitrarily extensible. Providing for, and accommodating, arbitrary extensions to

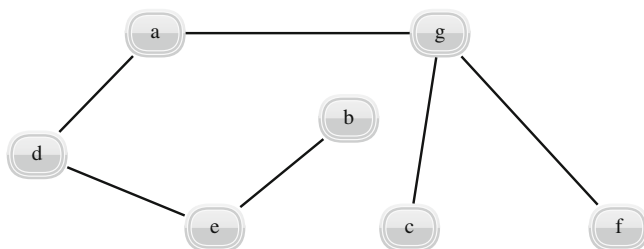


Fig. 2.10 A free tree

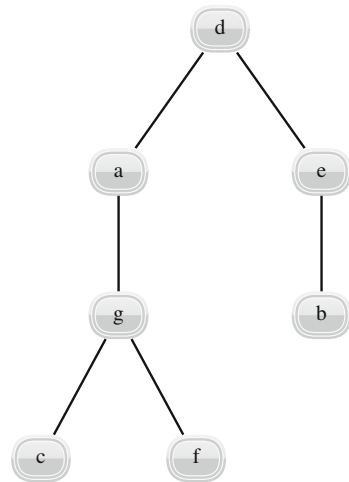
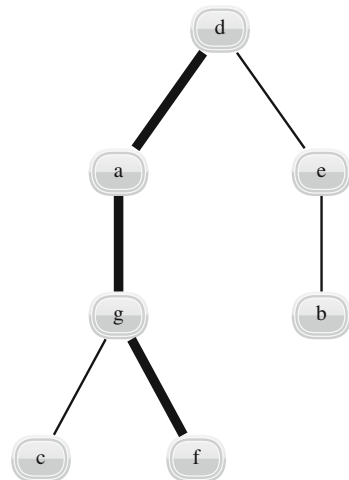
metadata, sometimes called ‘extended attributes’, is a desirable end in itself (Siracusa 2001, 2005). Usually with a pure association list, the actual number of entries, the number of key-value pairs is not limited (just as an ordinary shopping list can have as many entries as needed to suit the task at hand). But in practice many metadata standards or schemas have a fixed number of fields. For example, the ordinary original Dublin Core metadata standard had 13 fields in it. It is better if the number of metadata fields is *not* fixed i.e. that we can add more fields if we wish to. The reason for this is that the metadata fields are used to organize the IOs, but we cannot envisage now all the different ways that folk will want to organize their IOs, so we cannot know now, once and for all, just what fields to have. A metadata scheme should allow augmentation. A single item of metadata, or a metadata field, is an annotation. And a new grouping amounts to a new annotation. There is no reason to limit annotations.

2.8.8 Trees, Hierarchies, Forests and DAGs

We will have a special interest in what is known as *rooted directed trees* and *directed acyclic graphs*. We will often call the former just *trees* and the latter *DAGs*. [For the connection between trees and information retrieval see, for example, (Salton 1962).]

A *free tree* is a connected graph, which has only one path between every two nodes (i.e. there are no ‘cycles’). For example, (Fig. 2.10) is a free tree (you can walk from node to node (so it is connected) but you cannot walk around in a circle (so there are no cycles)). A free tree can be converted into a *rooted tree*, by choosing one of the tree’s nodes as being the ‘root’. Then it is common to draw it with the root at the top, and the other nodes appearing below. So if the node d were to be chosen as the root of a rooted tree from the first diagram, the tree might be re-drawn (Fig. 2.11). With a rooted tree, there are nodes which do not have other nodes below them. These are *leaves*. So, in the above tree, ‘a’ is the root, and d, f and g are leaves. With any leaf, there is exactly one path from the root to it, such a path is a *branch* (Fig. 2.12).

The emphasized lines indicate a branch in the above tree, i.e. d-a-g-f, (and the tree has two other branches).

Fig. 2.11 A rooted tree**Fig. 2.12** A branch

Trees can be characterized, and drawn, as having directed links (and that will be the preferred form for us). Thus (Fig. 2.13).

The root has in degree 0, no links come into the root. All other nodes have in degree 1 (the incoming link picks a node's 'parent'). The leaves have out degree 0, no links go out from a leaf. All other nodes have an out degree that is not 0. (The outlinks identify a node's children.) All the links point away from the root. Sometimes these rooted trees with directed links are called *rooted directed trees* or *arborescences*. We will tend to abbreviate this and just call them *trees*.

The number of children that a node has is a matter for the tree and its intended use. There are *binary* trees in which every non-leaf node has exactly two children. But there are also many other types of tree. Sometimes the words 'branching

Fig. 2.13 A tree using directed links

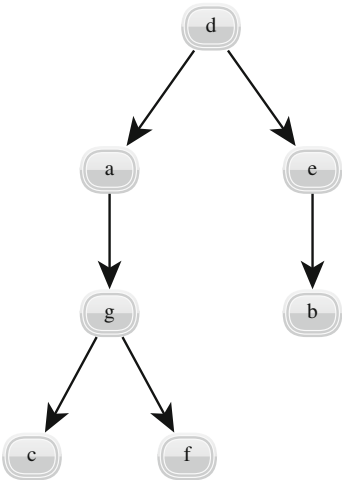
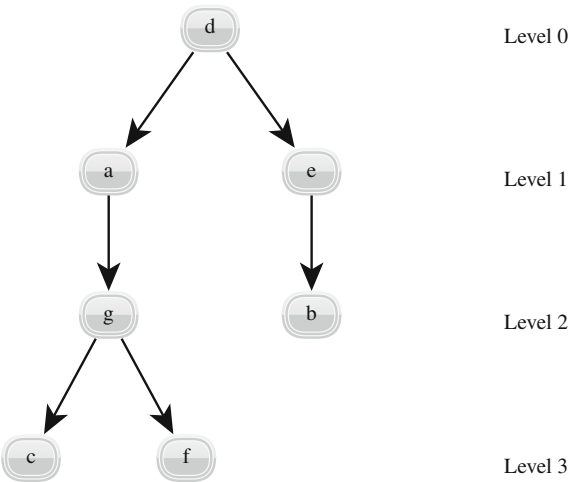


Fig. 2.14 A tree showing levels

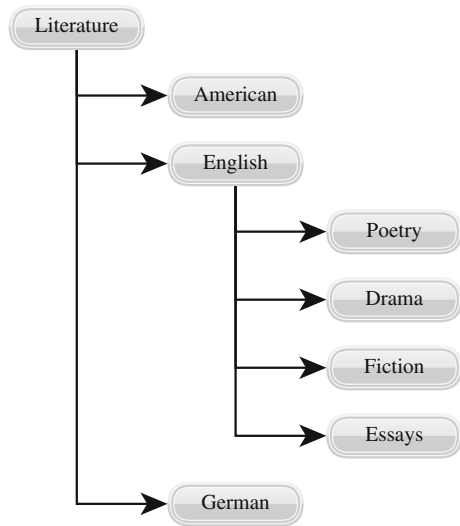


factor’ are used to indicate the relative number of children, so a tree with a high branching factor would tend to be a fat or wide tree.

Rank or *level* can also be given to the nodes in accordance with a node’s distance from the root. So, in the following diagram, the root (i.e. a) itself is level 0, the nodes a and e are of level 1, b and g level 2 and c and f level 3 (Fig. 2.14).

The notion of ‘hierarchy’ often appears in this setting. A hierarchy is a structure of entities where there is a ranking by status or importance or authority. Many hierarchies can be depicted as trees. Then the most important item becomes the root, the second most important items are shown as the children of the root i.e. they are of distance 1 from the root and so on. Hierarchies can depict governments, committee structures, military ranks, etc.

Fig. 2.15 A tree drawn horizontally



Common examples of trees are organizational (‘org’) charts, genealogical trees, and the display of any hierarchy (such as genus-species in biology, classification schedules and subject schedules in librarianship).

The last displayed tree could be a genealogical tree where the links are showing the ‘mother of’ relation (with $a = \text{Alice}$, $b = \text{Betty}$, $c = \text{Chloe}$ etc.). Notice that each of the individual nodes (the root, the leaves, and the internal nodes) depict someone (i.e. the ‘data’ is everywhere), and the relation connecting two nodes (‘mother of’) is the same throughout. Each person is somewhere, at a node, and there are exactly the same number of nodes as people. And the levels can be used to pick out grandmothers, great grandmothers, etc.

[An oddity, that is occasionally remarked on, is that, strictly speaking, family trees are not trees. In a family tree, a child has two parents, in a graph theoretic tree a child has one parent. The work-around we used above was to use just a partial family tree, linking nodes using only the ‘mother of’ relation.]

One familiar use of trees is to depict directory or folder structure on a computer. For that use, a tree will typically be drawn horizontally, thus (Fig. 2.15).

With standard trees, there is no notion of an order to the children (which is the first child, which is the second...)—all that matters is what is connected to what. So, for example, the following two trees are the same tree, or copies of each other (Fig. 2.16).

For certain purposes, for example for representing books, it is convenient to have an order on the children so that, for example, b is the first child of a , in the above graph, and c the second child, etc. These types of trees are *ordered* trees. Order can be depicted just by labeling or tagging the nodes, thus (Fig. 2.17).

Here the numbers are showing which is the first child, which the second, etc. Often, for casual work, we can just let the left to right order on the page represent the order of the children.

Fig. 2.16 Two different depictions of the same tree illustrating indifference to the order of children

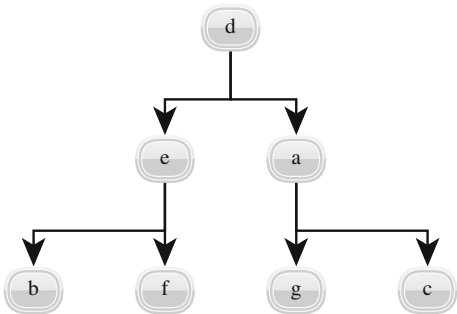
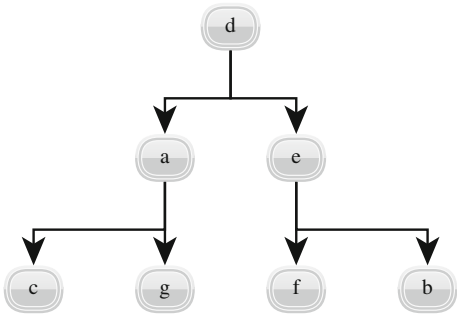
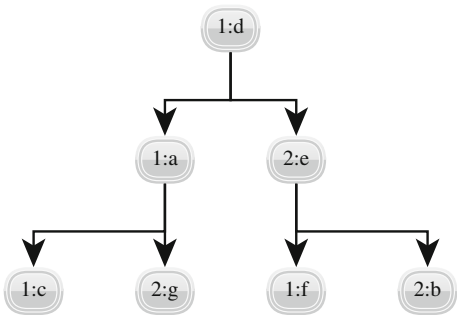


Fig. 2.17 An example tree with ordered children



There is often the need to locate, or provide a reference to a node in a tree. As examples, in a genealogical tree there may be an interest in where Mary Queen of Scots occurs. And in a classification tree, such as the Dewey Decimal Classification (DDC), there may be an interest as to where the topic ‘Physics’ occurs.

One easy way to do this is to provide a *notation* that identifies the individual nodes via parents and children (a combination of levels in the tree and order among the children). So, for example, the numeral 1 could label the root; 11, 12, 13 etc. label the first, second, third, etc. child of the node 1; 111, 112, 113, etc. label the first, second, third, etc. child of the node 11; and so on. There are indefinitely many such notational schemes. One attractive feature that many notations can have is that of being *expressive* and that means that the notation indicates the hierarchy and the nodes’

individual lineages. For example, if we choose the label of a node from the sample scheme, say 1,221, that node is a child of the node 122, it is a grandchild of the node 12, and it is a great-grandchild of the node 1. The node's lineage is available directly from the notation for it; and that is true for all nodes labeled using the sample scheme; the sample scheme is expressive. [If a tree is depicting a classification, the notation can tell of class numbers or classification numbers.]

Alternatively, with a tree, there is a unique path from the root to each individual node which can generate a description. That is the technique used on personal (and other) computers. You might see a 'path' from the document root of the computer down to the file of interest

`/Users/Documents/Synch/Readings/Pliny.pdf`

and that locates the document 'Pliny.pdf'. And the User follows this path by finding the correct child (folder, directory) of the root (Users), then the right child of that child (Documents), and so on. How difficult this would be, depends on how many children there are, at each stage, and whether the children themselves have some arrangement. When the tree is an ordered tree, the children will be in alphabetical (or hopefully some other useful) order. In which case, following the path just involves a sequence of jumped key entries.

One can go further by providing an association list, preferably an ordered association list, with the node names (or file names) as keys and the notation for the nodes (paths of the files) as values. That, to take one example, could be an alphabetically sorted association list which amounted to an index from ordered topics ('Chemistry', 'Mathematics', 'Physics') to their positions in a scheme such as DDC; indeed, that is what the Dewey Relative Index is.

Sometimes a graph amounts to several different independent trees. These graphs are *forests*. A tree is acyclic (it has no cycles) and it is connected. Since the trees in a forest are not connected to each other, effectively a forest is a collection of nodes which are acyclic. A forest as a whole does not have to be connected, but the parts of it that are connected form the individual trees. When talking about cycles in connection with trees and forests, no attention is paid to the direction of the links. However, we favor directed links for illustrating trees. And that leads to the notion of a *directed cycle* (having the ability to follow the links in the right direction and get back to where you started).

Graphs that do not have directed cycles are, surprise, *directed acyclic graphs* (or DAGs). Roughly, a DAG is like a forest but requiring the absence of directed cycles instead of the absence of plain cycles. Any tree is a DAG, any forest is a DAG also. But there are DAGs which are neither trees nor forests, this occurs when they contain cycles but not directed cycles. They contain one or more (larger or smaller) 'diamonds'; a simple example is (Fig. 2.18).

What a DAG amounts to for us is this: any node might have several children and thus many descendants down different paths; *any node might also have several parents* and thus many ancestors up different paths; also different descendants might have a child in common, or different ancestors might themselves have a parent in common (i.e. there might be cycles); however, no node can be a descendant or an ancestor of itself. This is a generalization of tree or forest.

Fig. 2.18 A graph fragment with nodes connected in a ‘diamond’

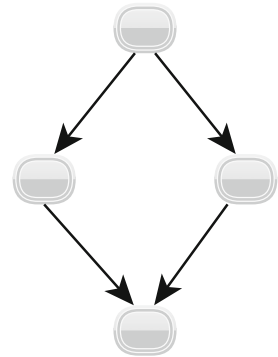
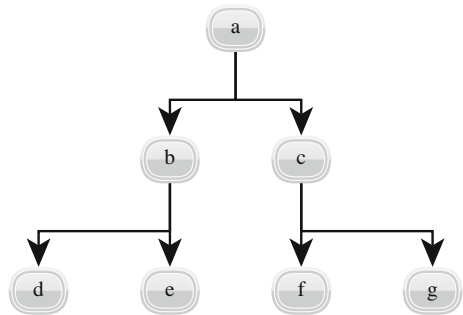


Fig. 2.19 A tree that will be traversed



With DAGs, the notion of branch does not fit too well, but the idea of path can pretty well serve as a replacement. Starting at any node, it may have children, and they may have children, and so on; but following the links downward must eventually come to a stop (because, with the graphs of interest to us, the graph is finite.). This means that effectively each node sits as the source of paths of descendants going down. Similarly that node may have parents and they may have parents, and so on. Now, a link goes from a parent to a child so to travel from a child to a parent is traveling against the direction of the link; so, following the links upwards, against the direction of the links, produces paths of ancestors going up. The paths going through a node are the combinations of these downward paths to the descendants and upward paths to ancestors.

2.8.9 Converting an Ordered Tree to a List

There are various ways of ‘traversing’ an ordered tree. The task of traversing is that of visiting all the nodes in a tree (just once) i.e. the root, the interior nodes, and the leaves. The output from a traversal is a list; it is a list of all the nodes visited, and it needs to contain all nodes in the ordered tree. So, with the tree (Fig. 2.19).

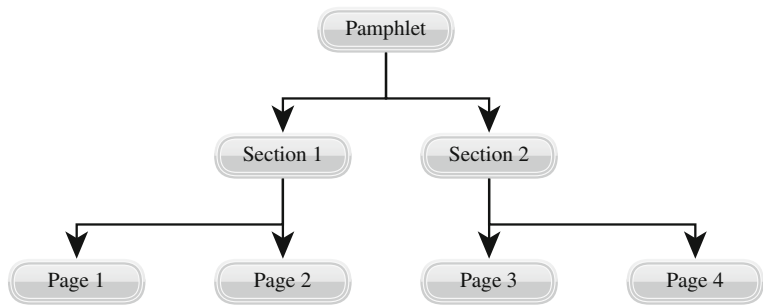


Fig. 2.20 A tree depicting the structure of a pamphlet

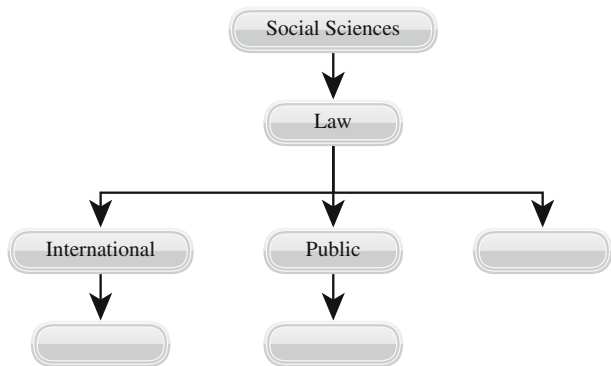


Fig. 2.21 A fragment of universal decimal classification (UDC)

The list a, b, c, d, e, f, g is a traversal, so too is a, b, d, e, c, f, g. The latter style of traversal is of particular interest to us. There are two reasons.

Any book, or magazine, or pamphlet, of Chapters, Sections, Sub-sections, Pages, etc. can be thought of as a structured object, in fact all of them are ordered trees. The book itself it the root, the Chapters come at level 1, the Sections at level 2, and so on. Thus (Fig. 2.20). And the described traversal is how the document would be printed and bound (or presented).

The second reason is that librarians use subject classification to produce classmarks and, in turn, the classmarks are used to place physical IOs in order on shelves. The end result is IOs collocated by subject. The subject classification is often a hierarchical tree, and the shelving order is a list. So the process is that of converting an ordered tree to a list. For example, here are a couple of fragment branches from the Universal Decimal Classification (UDC) (Fig. 2.21).

- 3 Social Sciences
- 34 Law
- 341 International Law
- 342 Public Law

For simplification we will assume that the example books are assigned just one subject. And the cataloging practice is to use both the leaf nodes and the internal nodes from the schedule for actual classification. So, one book might be on the subject of Law and have the classmark 34 and another book might be about International Law and have the classmark 341; and when those books are placed on shelves the 34 will be placed before the 341. So this is the same list producing ordered tree traversal as that used for producing pagination from a structured book.

We will label this traversal, a ‘paginating’ or ‘shelving’ traversal.

2.8.10 Tables, Citation Order, Filing Order

Tables are very familiar; for example, here is a schematic from an Email application showing each individual email as a row. The columns identify the sender of the email, what its subject is, and what its date is

Name	Subject	Date
Ann	Breakfast	2010
Ann	Lunch	2008
Beryl	Breakfast	2009
Beryl	Lunch	2008
Charley	Breakfast	2009
Charley	Dinner	2008
Charley	Lunch	2010

So each row here is an ordered triple, for example the first row is

<Ann, Breakfast, 2010>

where each of the components of the triple are the particular column values for that row e.g. 2010 is the date of that first email. And an individual row represents information about one item or token. So, typically, there are rows and columns, and often row headers and column headers (which identify the rows and columns). Each row contains data pertaining to one item or object (the above table has seven rows, so it has data about seven items, seven emails). Each column contains values for each of the properties that each item might have [so, for example, the Date property has a value of ‘2008’ for three of the emails (i.e. three emails have 2008 as their date)].

With headers in place, in one sense the table and its layout are flexible. The following table is identical to the last one in as much as it contains the same information.

Subject	Name	Date
Breakfast	Ann	2010
Lunch	Ann	2008
Breakfast	Beryl	2009
Lunch	Beryl	2008
Breakfast	Charley	2009
Dinner	Charley	2008
Lunch	Charley	2010

And tables in general can contain many columns, certainly more than three, in which case, for an n -column table the row entry would be not an ordered triple, it would be an ordered n -tuple, and these row values are often called just *tuples*.

In the setting of information resources, it is convenient to call the column order the *citation order*. So the citation order of the first table is Name-Subject-Date, and the citation order of the second table is Subject-Name-Date. The order of the columns, the citations order, is a matter of indifference as far as the total information in the table is concerned. The order of the rows is also a matter of indifference as far as the total information in the table is concerned. However, having the rows sorted is of definite value for the purposes of looking up or finding a row, and of having related rows contiguous with each other. The actual sorting can be done alphabetically, or inverse alphabetically, for names, by currency for dates, or any number of other ways. The result of sorting the first table, by sorting the columns by subject, then ties by name, and finally the ties of those by date (i.e. in citation order) is

Subject	Name	Date
Breakfast	Ann	2010
Breakfast	Beryl	2008
Breakfast	Charley	2008
Dinner	Charley	2008
Lunch	Ann	2008
Lunch	Beryl	2008
Lunch	Charley	2010

To actually produce that result the columns of the table are sorted successively from right to left (somewhat counter-intuitively). Sorting one column, after sorting another, may to a degree overwrite, or unravel, the first one—so the last column sorted becomes, so-to-speak, the master sort. So sorting by Date, then by Name, then, last, by Subject will give the desired citation order sort. And if we change the citation order to Date-Name-Subject, and do a table sort on that, the result is

Date	Name	Subject
2008	Ann	Lunch
2008	Beryl	Lunch
2008	Charley	Dinner
2009	Beryl	Breakfast
2009	Charley	Breakfast
2010	Ann	Breakfast
2010	Charley	Lunch

Notice here that the sorting and the citation order combine in the following way: rows are contiguous with other rows having similar values for the first column of the citation order, but they are scattered with respect to values for other columns in the citation order. So, for example, you can have all of Ann's and Beryl's and Charley's emails together, or you can have all the 2008, 2009, and 2010 emails together, but you cannot have both of these associations at the same time. In classification theory, this is known as *the problem of distributed relatives* (Savage 1946).

There is an analogy here with the word-by-word filing order sorting of alphabetical lists. Each column entry needs to be a separate word in the key string, and the citation order is the order of those words in the keys. Then, if the columns can be sorted alphabetically (so, for example, no column consists of numbers sorted by size), a plain alphabetical list and the row order in the table are much the same.

The contents of a table can be linearized, or provided as output in a list form, merely by listing the rows in order. That linearization, its order and what it makes contiguous and what scattered, will depend on the citation order and the ordering or sorting operations that are done on the columns.

Using once again terminology common in the setting of information resources, that (output) list of rows from a table is known as the *filing order* for the rows.

More can be said about the sorting using the columns. Mention has been made of alphabetical sorting, and sorting by date, but, actually any operation will do that produces an order. Consider, for example, our sort on the subjects. There were the subjects 'Breakfast', 'Dinner', and 'Lunch', which were sorted alphabetically. But there are indefinitely many other sorting operations that could have been used. These subjects might gain an order by being just three subjects chosen from a ordered list of 100 possible subjects, or controlled annotations. This larger list, a controlled vocabulary of subjects, could be ordered from general to specific, from most urgent to least urgent, from most interesting to least interesting, from most relevant to financial taxes to least relevant, and in other ways. The sorting on a column groups certain rows and separates them from other rows or groups of rows; for example, an alphabetical sort puts the entries starting with 'C' closer to each other than to entries starting with 'Z'. This means that rows can be grouped or scattered just by virtue of a single column sort.

So, filing order depends both on citation order and on the individual sorting operations on the individual columns. And filing order is an expression of what is collocated and what scattered. There is much flexibility.

2.9 Roget's Thesaurus

Roget's Thesaurus, written by Dr Peter Mark Roget in 1805, is, inadvertently and unintentionally, the single most important work in the theory of information retrieval.

Roget sought to remedy his 'own deficiencies' in finding the appropriate word or words to use when composing and writing ((Roget 1852) Introduction). And to do this he produced a 'system of verbal classification' which was a 'classed catalogue' ((Roget 1852) Introduction). In it, words were partitioned into clusters of synonyms, or near-synonyms, and then these synonym clusters were classified as leaves in a tree. In turn, the non-leaf interior nodes in the tree represented organizing principles, such as

- Abstract Relations
- Space
- Matter
- Intellect
- Volition
- Sentient and Moral Powers

These organizational nodes went from the more general to the more specific as the leaves are approached. Typically sibling leaves, being under the same organizational parent, would be related in some way, though not as synonyms; for example, the 'life' leaf and the 'death' leaf are sibling leaves. Roget then combined this classification of synonym clusters with an alphabetical index. And the result, *Roget's Thesaurus*, has been invaluable to writers and scholars ever since.

In the setting of information retrieval, there is no special interest in using apposite or elegant words when writing. So, what *Roget's Thesaurus* was designed for, and what it is popular for nowadays, is tangential to the present enterprise.

However, Roget's techniques, what he did and the way that he did it, are vital for information retrieval. Roget combined (part of) Reid's classifications, the Triangle of Meaning, the problem of synonyms and homographs, and alphabetical indexing in such a way that all obvious problems are solved. There has been no improvement in Roget's approach in the last 200 years, and it looks doubtful that there could be any major improvement.

So, what did Roget do? He produced a Reid-style classification of some concepts. In fact, this amounted to a tree with about 1,000 leaves. The tree was about of depth 3, so it was shallow and broad (There were Classes, Sections, Subsections, and Leaves.). He labeled each of the nodes, so, schematically, there was

Class #1, Section #1, Subsection #1, Leaf #1,... Leaf #1,000. His choice of concepts for the tree was inspired by the works of Leibniz and Aristotle. What Roget’s concepts actually were is unimportant to us. Roget was choosing concepts to be helpful for writers. In information retrieval or librarianship, we might choose concepts for other reasons (for example, to depict topics or subjects or area of knowledge). Each of the leaves in the tree was to be a near-synonym cluster. Each cluster had a ‘headword’, which, in effect, was a label for its cluster. So, for example, the word

life
is a headword, and it is a label for the cluster

life, vitality, viability; animation; vital spark, vital flame, soul, spirit. [and more].
Roget was not trying to produce true synonyms—he would not have thought for one moment that ‘life’ and ‘vitality’ were synonyms. He was trying to help himself and other writers searching for the right word, the right nuance, and so he clustered near-synonyms. The ‘life’ cluster is a node in the tree and as such it has a node number and, in fact, that number is Leaf 359. So, thus far there is the example data

359: life, vitality, viability; animation; vital spark, vital flame, soul, spirit.
and this data is a leaf in a classification tree. So, if a User has found Leaf 359 in the tree, the User can read off near-synonyms of ‘life’; the User can also go up a link to the Organizational Subsection ‘Vitality in general’ and, if desired, come down to Leaf 360 with headword ‘death’ and read off near-antonyms of ‘life’. What about the problem of finding the ‘life’ Leaf in the first place? If the User fully understands the organizational principles, and where the concept of ‘life’ would be classified, that User would be able to find Leaf 359 and thus access the sought for synonyms. However, there are 1,000 leaves in the tree, and what fits where is always a bit of an open question for a User. What would make the system better is jumped key entry from the headwords to their nodes, from, for example, ‘life’ to Leaf 359. And Roget provided exactly this by including a supplementary alphabetical index from the headwords to their nodes. So, for example, here is a schematic of that alphabetical index for a fragment around ‘life’:

...	
lieu	182
lieutenant	745
life	359
lifeblood	5
lifeboat	273
...	

[The numbers are the numbers of the respective leaf nodes. The leaf nodes are themselves presented in order, in the appropriate section of the entire Thesaurus, so finding one from its number is trivial.]

Roget met the problem of homographs by refusing to use them as headwords (so each actual headword names only one concept). He disambiguated any potential headword homographs, and then used the disambiguations both as headwords and in the alphabetical index. A more complete version of the above index fragment is

lieu	182
lieutenant	
officer	745
deputy	759
life	359
lifeblood	5
lifeboat	273
...	

and this recognizes the plain 'lieutenant' as being a homograph and disambiguates it to 'lieutenant (officer)' and 'lieutenant (deputy)', and those latter two are headwords (or head strings or head phrases).

From our point of view, the only thing that Roget did not do was to use the depth of the classification and make proper use of the interior nodes. To explain:- One of Reid's examples is the class hierarchy

```
Animal
Man
Frenchman
Parisian.
```

All of these are perfectly good concepts (or words for concepts). So, 'Animal', 'Man', 'Frenchman', 'Parisian' could all be headwords and have their associated clusters. However, when these clusters are put in a classification tree, they do not all want to be leaves simply because some are subclasses of others (Parisian is a subclass of Frenchman, which is a subclass of Man, etc.). So, the superclasses all need to be interior nodes, and it is only the terminal subclasses, Parisian in this case, that should be leaves.

We definitely want this modification, because we want well-behaved names (headwords) for all the classificatory concepts including the interior nodes. And, actually, the modification even makes for a better Thesaurus for the purposes of writing. The directed links in the tree would now depict whether one cluster was more specific than another. By this means, a writer could find a hypernym (a word more general than the one in hand) or a hyponym (a word more specific than the initial one). So, if the writer wanted a word more general than 'Parisian', the Thesaurus could suggest 'Frenchman' as a possibility.

There are some minor details. There is the question of which words to index. In the original Thesaurus there were 1,000 headwords and 15,000 words in total. The headwords have to be in the alphabetical index. Should the other words be indexed? If they are, there would be an index of 15,000 entries (or 325,000 entries nowadays, for Roget's Thesaurus has grown with the passage of time), which is large for an

alphabetical index (but not large compared with the New York Telephone Book, which is also an alphabetical index). If the other words are not indexed, then there is a problem with finding synonyms etc. of non-headwords. For example, take the word 'viability' (actually from the life cluster 359): it may be difficult for a User to find synonyms for it, without knowing where it is in the classification. Nowadays, with computers, there would be no reason not to index everything, either by way of an alphabetical index or by keyword entry and search. Additionally, Roget himself enhanced the basic structure by adding many cross references and heuristic guides, both to the clusters in the classification and to the index. These are valuable.

For our purposes, it will be useful to mention some alternative terminology. We do not have much use for the actual word 'headword', though its role and what it signifies are vital. We will tend to use 'preferred term', or 'name of the concept' or 'canonical name of the concept' or 'canonical name', all as synonyms for 'headword'. [The reason is 'headword' alludes to the first word in a list of words, but our interest is in well-behaved one-to-one names of concepts.]

One major way that we are going to want to adapt Roget's original work is that in the setting of information retrieval it needs to be generalized from atomic or elemental concepts to compound concepts. In Roget, the concepts in the tree are atoms and, for the most part, the headwords and quasi-synonyms are single words. But, with IO retrieval, a greater proportion of the interest is with compound concepts not atomic ones. Searchers are not just interested in 'life', they are interested in 'moral life', 'life in nineteenth century France', 'the good life', 'right to life', 'healthy life', 'extending life', 'the life of Nelson', and indefinitely more compound concepts. The compound concepts will likely need some special treatment to produce 'headwords' which, actually, would likely be 'headphrases'; and, similarly, the phrase near-synonyms, and alphabetical indexing will raise demanding questions.

2.10 The Dewey Relativ Index

It was Dewey who devised or re-invented the use of a combination of classing with an A/Z alphabetical index. He wanted classes to relate books one to another and to shelve books. But that on its own does not help the User either to find which class a book belongs to or to find where the classes are on the shelves. The solution to these problems is first to label each of the classes, with its Dewey Decimal Number, and to put the classes on the shelves in order of their numbers, and second to provide an alphabetical index from class name to class number, the 'Relativ' index. Here is Dewey, writing in the language of full Dui

The plan of this Clasification and Index was developot erly in 1873

Only a fraction of the servis posibl cud be got from [libraries] without clasification, catalogs, indexes and other aids, to tel librarians and readers what they containd on any givn subject....

It was chiefly necessary to find a method that wud clas, arranje and index books and pamflets on shelvs, cards of a catalog, clippings and notes in scrapbooks and index rerums, references to all these items, and indeed any literary material in any form, as redily as an ordinary index gyds to proper paje of a bound book. This difficult problem was solved by uzing no reference marks except the simplest simbols known to the human mind, arabic numerals with their uzual arithmetic values, and by aiding their unequald simplicity by many practical nemonic devices.

Tho the importance of clasification was recognized, the filosofic sistems proposed wer so difficult fully to understand or apply that not 1 person in 1000 cud uze them practicaly. Decimal Clasification simplicity and even more its Relativ Index hav made this work 10-fold eazier. In recent years, use of the sistem has spred rapidly in all civilized cuntries, meeting success in thousands of different applications. In its simpl form a skoolboy can quickly master it and keep for instant reference not only his books but every note, clipping or pamflet. Almost every profession and occupation has lerned its wonderful laborsaving powers. It is in daily use by miriads of business and professional men who wud never even attempt to understand or uze the old sistems.

By mere adition of figures, without chanjing this shorter form, this very simpl sistem is redily made to record the utmost refinements of specialists, and the Relativ Index, as simpl as a, b, c, sends the novis to the exact place where the expert has clasifyd the matter sought (Dewey 1926) Introduction.

And, to an extent, Dewey constructed his keystings for the Relativ Index in such a way that problems of vocabulary control (synonyms and homographs) were addressed. The keys also often showed the subclassing.

Although the setting, context and interpretation, of the two contributions are very different, what is going on here is very similar structurally to the construction of Roget's Thesaurus (with Dewey's decimal numbers substituted for the Roget's leaf numbers).

[The reason that the Dewey 'Relativ' index is called a Relative Index is that it collects together relatives. As we noted, classification typically collocates some items and distributes others (the so-called 'distributed relatives'). So, for example, in the Dewey Decimal System, items of Poetry are distributed (because they appear separately under English Literature, American Literature, etc.); the Relative Index, among other virtues, tells where these are. The Dewey Relative A/Z Index fulfils two functions: it helps to find classes, and it helps to find relatives.]

2.11 Tables of Contents and Back-of-the-Book Indexes

Any book, or series of volumes, might be accompanied by a Table of Contents, and an Index. These are access or retrieval tools. Pliny's *Natural History* of 77CE had Tables of Contents and rudimentary lists of author references (by way of source acknowledgment and citation). A Table of Contents will be similar structurally to the actual contents—it will show the Chapters and Sections in the same order as they are in the book itself. So, really, looking though a Table of Contents is just an abbreviated and quicker version of looking through the book. Tables of Contents usually list Chapters, and Sections, but not pages themselves (except

Table 2.1 Table of contents for Pliny’s Natural History

1. Of land creatures: the commendation of elephants: their understanding.	15. Of the animals of scythia, and of the north countries.
2. When elephants were first yoked.	16. Of lions.
3. The docility of elephants.	17. Of panthers.
4. The clemency of elephants: that they know their own dangers; also of the ferocity of the tiger.	18. The nature of the tiger: of camels, and the camelopard: when it was first seen at Rome.
5. The understanding and memory of elephants.	19. Of the stag-wolf named Chaus: and the Cephus.
6. When elephants were first seen in Italy.	20. Of the rhinoceros.
7. Combats by elephants.	21. Of lynxes, sphinges, crocutes, marmosets, of Indian oxen, of leucrocutes, of eale, of the Ethiopian bulls, of the Man- tichora, the unicorn, of the catoblepa, and the basilisk.
8. The manner of taking elephants.	22. Of wolves.
9. The manner how elephants are tamed.	23. Of serpents.
10. How long an elephant goeth with young, and of their nature.	24. Of the ichneumon.
11. The countries where elephants breed: the discord between elephants and dragons.	25. Of the crocodile and the hippopotamus.
12. The industry and wit of dragons and elephants.	26. Who shewed first at Rome the hippopotamus and crocodiles. Medicines discovered by animals.
13. Of dragons.	27. Of animals which have shewn certain herbs; the red deer, lizards, swallows, tortoises, the weasel, the stork, the boar, the snake, panther, elephant, bears, stock-doves, house-doves, cranes, and ravens.
14. Serpents of prodigious magnitude: of serpents named Boae.	

perhaps as references identifying the locations of the Chapters and Sections). For example, here is a Table of Contents from Pliny (Pliny 78)

IN THE EIGHTH BOOK IS CONTAINED THE NATURE OF LAND ANIMALS THAT GO ON FOOT (Table 2.1).

A Table of Contents supports browsing and searching, but it is better for browsing than it is for searching. To search, you have to find what you interested in within the Table of Contents itself. For example, say you are interested in ‘Stock-Doves’, such a topic, or part heading, may or may not be in the Table of Contents, it may or may not be there under that exact name (it could be there under ‘Doves-Stock’), and there is no key jumping from ‘Stock-Doves’ to its location in the Table of Contents, so, in the worst case you might just have to start at the beginning of the Table of Contents and look through sequentially until you either did or did not find ‘Stock-Doves’ (or some title that you recognized as being related). However, even

with all those difficulties, looking for your topic in the Table of Contents is easier and quicker than looking for it in the book itself. Because of the isomorphism between contents and Tables of Contents, no special decisions have to be made by the author or publisher about making or constructing a Table of Contents. If a book is in the form of Chapters and Sections, the structure of a Table of Contents is a trivial distillation or abstraction of that structure. There is a qualification here. Getting the structure of the Table of Contents from the book itself is trivial. But which actual labels to use is an important and non-trivial consideration. Of course, typically the labels will come from the author (or authors). But all that means is that it is the author that has to do the intelligent thinking. The author will have given the Chapters, Sections, etc. titles or labels. And suitable choices are important. Pliny calls one of his sections ‘When Elephants were first seen in Italy’ and another ‘Of wolves’—there are different levels of guidance and help that are being offered here.

Back-of-the-book Indexes are quite different to Tables of Contents. Indexes were slow to emerge (Witty 1973), and there are reasons why. Many books from around the period of Pliny’s *Natural History* were in the form of scrolls, so there was only one ‘page’ (although, of course, any particular book might consist of several scrolls) and usually there were no line numbers. The text was hand written, so what was on a particular line in one version might not be on that line in another. If a Table of Contents already informed as to sequential order within a scroll, what problem would invite an index as its solution? It is hard to identify one. Presumably any desired access to a scroll, via rolling and unrolling, could be guided and accomplished by a suitable Table of Contents. In the first three centuries AD, *codexes* (i.e. books with pages) start to have a real impact, but these also were hand written. There could be multiple copies of a work, sometimes hundreds, but these copies were often not the same as each other in physical form. It is only with the advent of the printing press, around 1450, that there starts to be large numbers of copies of works with the individual copies having the same pagination. From about 1500, indexes start to appear, but it is probably as late as 1700 before alphabetical indexes are common (Wellisch 1983, 1986, 1991; Witty 1973). As we have already noted, the notion of an alphabetical list itself was slow to be recognized and appreciated. Recognizable modern style alphabetical indexes are relatively recent. As Witty tells us

Index entries were not always alphabetized by considering every letter in a word from beginning to end, as people are wont to do today. Most early indexes were arranged only by the first letter of the first word, the rest being left in no particular order at all. Gradually, alphabetization advanced to an arrangement by the first syllable, that is, the first two or three letters, the rest of an entry still being left unordered. Only very few indexes compiled in the 16th and early 17th centuries had fully alphabetized entries, but by the 18th century full alphabetization became the rule... (Witty 1973, p. 136)

Indexes are usually not ordered in the same way as contents of the book. Contiguous index entries do not have to refer to contiguous passages or sections. An index is an association list of labels (or keys) and references. The labels are the names of the entries, and the references are the Chapters or Sections (or page or

line numbers) that those names individually refer to. An index has to have some order, presumably a principled order. As we have noted, an alphabetical index has jumped key entry, which is a pleasant property. But an index does not have to be alphabetical. There may be reasons to encourage sequential access to the index itself. For example, it might be better to learn or study a particular subject matter in a certain order, and the index could support this. Or, certain tasks might need to be done in a certain order and indexes can support this, so, for instances, an index to Aircraft Flight Manual might amount to a checklist for pilots (for pre-flight checks and similar).

Nowadays we would reason that there are two complementary accessing tasks of interest and value to the reader: browsing and searching. A Table of Contents supports primarily browsing, and an Index supports primarily searching. The reader needs to be able to do both tasks well; therefore many books need both a Table of Contents and a separate, and differently structured, Index. But it requires insight to realize this. It is easy to be tempted by the following line of thought. Suppose you do feel you would like an Index. Why not just structure the text itself in exactly the same way as the structure of the Index you favor? So, if you would like an alphabetical Index, just make the text itself ordered alphabetically by section entry (as is done in a Dictionary). The moment the text has the same structure as the Index, the Table of Contents and the Index both would have the same structure—they are the same, and so one is redundant. This makes an Index unnecessary. We do not have indexes into Dictionaries, there is no need (Dictionaries also do not have Tables of Contents; that is because there is no need for them to support browsing; the individual entries are essentially independent.).

2.12 More on Back-of-the-Book Indexes

Here, for example, is an index fragment from Colin Howson's *Logic with trees*

absurdity, rule of 137
 accessibility relation 166
 adequate: for truth-functional logic 44;
 sets of connectives vii, 44ff
 algorithm 17
 alpha (α) sentence 49
 alternative denial 44
 ancestral trees vii, 6, 33ff.

and the parts of the OED dictionary definition of 'index' of interest to us read

1. The fore-finger: so-called because used in pointing.
5. **a.** A table of contents prefixed to a book, a brief list or summary of the matters treated in it, an argument; also, a preface, prologue. *Obs.* **b.** An alphabetical list, placed (usually) at

the end of a book, of the names, subjects, etc. occurring in it, with indication of the places in which they occur. One work may have several indexes, e.g. an index of names of persons and places, of subjects, of words, etc. For these the Latin phrases *index nominum*, *locorum*, *rerum*, *verborum* are often employed as headings. **c.** A reference list. *Obs.* **d.** *Computers*. A set of items each of which specifies one of the records of a file and contains information about its address.

So there is here the notion of pointing or direction and an index is a collection or list of those items that do the pointing and those items that are pointed to. The purpose of an index is to help with finding or retrieving.

Often, but not always, the items pointed to can be accessed themselves, one after another, in an enumeration or a sequential scan. In which case, a direct search through the items pointed to could amount to, in the worst case, a possibly laborious, and potentially complete, sequential scan. An indirect search, via a search of the items doing the pointing, is intended to make this more efficient; it is intended to offer access to short cuts. It is certainly possible for a search through the items doing the pointing also to be a somewhat laborious, exhaustive, and entire sequential scan; however, the pointing items themselves are usually smaller in number than the items pointed to and also they have an arrangement, for example, alphabetical order, that allows searches of them to be guided, efficient, and direct. So, for example, if you were interested in the concept of ‘algorithm’, or the word ‘algorithm’, in Howson’s *Logic with trees*, looking directly through the 190 odd pages of the book would take some time; so, you look through the index (partially extracted above) for the term ‘algorithm’; this task also might take you some time if the list of indexed terms themselves had no structure and if the term ‘algorithm’ may or may not be among the list; but, the index terms are presented in alphabetical order, you can avail yourself of jumped key entry and find thereby the important page 17, the single and only page in the book that discusses or mentions algorithms.

Conceptually an index is an association list of key-value pairs. The keys are usually strings; they are headings or index entry names or terms. The values are lists of strings or lists of numbers, they are lists of items or descriptions of items (such as page numbers or web page URLs). So, in the above example, ‘algorithm’ is a key, and {17} is the value associated with it.

Index entries, or the keys, are often structured. So, for example, Howson has

adequate: for truth-functional logic 44;
sets of connectives vii, 44ff

this is showing that there are two subtopics, or sub-keys, for ‘adequate’, one concerning ‘truth-functional logic’ and the other ‘sets of connectives’. [We saw the idea of this earlier in [Sect. 2.8.4](#) on word-by-word filing order.]

The structuring here is a clue to an important underlying feature. With Roget’s Thesaurus we saw the use of structured keys to disambiguate homographs, to separate, for example, ‘lieutenant (officer)’ and ‘lieutenant (deputy)’. But the

Howson index word ‘adequate’ is not a homograph. The word ‘adequate’ labels just one concept and that is an end to the matter. But in the index, it is not ‘adequate’ *itself* that is being indexed, rather it is ‘adequate for truth-functional logic’ and ‘adequate sets of connectives’. These are labels for compound concepts, not elemental or atomic concepts. Roughly speaking, the original historical Roget Thesaurus used mostly atomic concepts and vocabularies for them, but we are going to want to generalize this to compound concepts.

The entry terms themselves, the keys or keywords, have to be found in the index and such devices as alphabetical lists help with this. This is worth mentioning because finding the entry names does not have to be easy. A medical text might have an index with key entries associated together by human body structure or function; in which case, finding the entry for, say, ‘thyroid’, might be a challenge for some Users. An approach here would be to use a second index, possibly an alphabetical index, to show where entries (like ‘thyroid’) appeared in the first index. So the second index gives locations in the first index, and the first index gives the pages in the book (This is somewhat similar to the two-step entry used with indexes in Thesauri.). In short, finding key entry terms, in the general case, is anything but trivial, with considerations of jumped key entry, key syntax, and indirect or stepping stone indexing.

2.13 Themes, Aboutness, Subjects and Polytopicality

There have been Tables of Contents pretty well since there have been written forms. What does a Table of Contents do? It tells you the contents and structure (especially if the IO is something like a linearized ordered tree of chapters, sections etc.). There is a coarseness to a Table of Contents. An IO itself, or a part of an IO, may have many words, sentences, and statements. A Table of Contents for that IO will have a lesser number of words. It is a distillation, an abstracting, a pilot text, as to the true full actual contents. One atom in a Table of Contents may point to an entire chapter in the text. It achieves this distillation by labeling what sections of the content are *about*, its *subjects* or *topics*.

Cutter described the notion of *subject* as being

... the matter on which the author is seeking to give or the reader to obtain information;
(Cutter 1876, p. 12)

It is one of the most important notions in information retrieval. In this book, we use ‘subject’ and ‘topic’ interchangeably, and both of them in a very casual and open fashion. Anything can be a subject. Or, to qualify this slightly, any entity can be a subject; that is, anything that can be denoted grammatically by noun or a noun phrase, can be a subject; any grammatical subject can be an information retrieval subject (or information retrieval topic).

To return to Pliny. Here again is one section heading from Pliny’s *The Natural History*

21. Of Lynxes, Sphinges, Crocutes, Marmosets, of Indian Oxen, of Leucrocutes, of Eale, of the Ethiopian Bulls, of the Mantichora, the Unicorn, of the Catoblepa, and the Basilisk.

This section, Sect. 21, is about Lynxes. But, obviously, it is also about Sphinges (and about Crocutes, Marmosets, etc.). It follows that a section, book, chapter, etc. can be *about more than one subject or topic*. The same is a possibility for any IO. When this occurs, the IO can be described as being ‘polytopical’.

Any section, chapter, book etc. can be about more than one subject. But also, different sections, chapters, books etc. can be about the same subject or overlapping subjects. For example, the first 12 chapters of Pliny’s *The Natural History* Book 8 (Table of Contents above) are all about elephants, perhaps among other things.

So, there can be an association list between IOs and subjects (that is the one between sections etc. and subjects). And this can be ‘inverted’ to produce an association list between subjects and IOs [The process of inversion is that of taking the keys in the first list to be the values in the second and the values in the first list to be the keys in the second. So

```

Chap1 elephants,
Chap2 elephants,
Chap3 elephants,
...
Chap 21 {Lynxes, Sphinges, Crocutes, Marmosets, ...}
Etc.
```

gets inverted to

```

crocutes      Chap 21
elephants     {Chap1,Chap2,Chap3, etc.},
lynxes        Chap 21
Etc.
```

].

And the result is a *subject index*. And, in principle, there could be a subject index that ranges over every IO (or, at least, a very large number of them in a particular collection or union of collections). And, indeed, ordinary indexes do not have to be confined to single books or related IOs—they too can range over many IOs.

An ordinary index and a subject index usually would not be the same, but they are not a lot different either. On various occasions, and for various reasons, there may be the need to index peoples’ names, or names of places, or names of battles, or historical periods, or topics, or themes, or subjects, etc. Of course, the categories of the items being indexed is different or can be different and the means for creating the indexes can be different (for example, with some being more amenable, and some less amenable, to automation). But the end result is much the same. A name index, for example, might be presented separately from a general index; and the main purpose of doing so would be just to make access and jumped key entry more effective.

2.14 Tag, Concept and Topic Annotation Languages

Let us rework the last section in a slightly different direction.

An ordinary back-of-the-book index is an association list between keys and values

absurdity, rule of 137
 accessibility relation 166
 adequate: for truth-functional logic 44;
 sets of connectives vii, 44ff
 algorithm 17
 alpha (α) sentence 49
 alternative denial 44
 ancestral trees vii, 6, 33ff.

To see how this was produced, consider the first entry ‘absurdity, rule of’ and imagine how it was arrived at. An indexer, or some algorithms, looked through the book and when they hit page 137 they decided that it should be labeled, or tagged, or annotated with the key ‘absurdity, rule of’.

Let us for the moment adopt the word ‘tags’ instead of ‘keys’, so back-of-the-book indexing is just tagging the pages—so too for subject indexing: it also is tagging. We know that the index keys can be structured, so, in the general case, the tags also can be structured. And in a standard index, a particular key might have several pages as its values (in the example fragment, ‘ancestral trees’ has the three values vii, 6, 33ff.), and any particular page might be among the values for several different keys (e.g. page 44).

That gives us the following tag structure and use. We can imagine a tag vocabulary or language, which consists of words or terms or phrases. There can be a hierarchy or tree of tags, with parent tags, child tags, leaf tags and a root tag. [In fact, more generally, these tags may be a directed acyclic graph, a DAG, not just a tree, and so some tags might have multiple parents.] Let us call this graph of tags the *schedule*. We can use our tag vocabulary and schedule to label, or annotate, or ‘tag’ anything we wish (book pages, the items in our rumpus room, bottles of pickles and sauces, or IOs). Any tag can be used on more than one item, and any item might have more than one tag.

A *tag language* is just the language of the actual and potential keys. Alternatively, and perhaps more commonly, it can be called an *index language*. It is an *annotation language*.

Now let us start with a particular leaf tag in a schedule and work our way up a branch. A leaf tag produces for us, or can be used to produce, a collection or class of those items which are tagged with it. So, the ‘hockey stick’ tag might pick out three hockey sticks in our rumpus room. The next tag up the tree might be ‘recreational equipment’. We understand this to denote anything it itself is tagged to, together with all those items tagged by its descendant tags. So this is going to identify a collection including the hockey sticks perhaps with a few tennis rackets and some golf clubs, and so on. The root tag ‘rumpus room’ might not itself

directly tag anything however, its descendant tags may tag most things in the room (It depends how assiduous we have been with our labeling.). If we repeated this for a different leaf, say that for ‘vacation items’ we would get a similar outcome. The end result effectively is a structure gerrymander for the whole rumpus room. This structured gerrymander is not the best for physically organizing the room simply because there might be overlap between leaves, a hockey stick might be tagged both ‘hockey stick’ and ‘vacation item’ i.e. two tags which are tags from different branches and which might require those items to be in two places at once—the old problem of physicality. However, the system does help search. We can start with any tag (or combination of tags) and use those tags to retrieve what we wish, and, if we also use the schedule to gain information on parents and children, we can broaden or narrow our search to suit our purposes.

The tagging helps the process of search. In a way, it does not really matter what the tags mean, or indeed if they have meaning in any ordinary sense of meaning. They can be just labels. What matters is that they have some significance for the searcher and there is an appreciation of how the search will work (i.e. with what it retrieves and what broadening and narrowing amount to).

In the setting of information resources and the retrieval of IOs, tags are often used to signify *topics* or *subjects*. So, for example, an IO may be tagged with the string ‘1915 San Francisco World’s Fair’ and that tag would be used to indicate one of the topics or subjects of the IO. In turn, this means that apparently a language of subject headings, used for discussing topics or subjects, can also be seen as being a tag language or an index language or an annotation language.

At this point we can use the Triangle of Meaning to make a large step forward for IO tagging and indexing. Tags, and tag and index languages, if understood in terms of strings, all belong to the Symbol Vertex of the Triangle of Meaning. But we can move to the Concept Vertex of the Triangle of Meaning and tag and index IOs using concepts and a *concept language*.

This can have two great advantages. It steps around problems of synonyms and homographs. And, largely by itself, it can tell of parents and children. With a tag language, there is the need to determine, or decide on, a tree of tags (or a directed acyclic graph of tags), for this informs us which tags are parents of which children. But, with concepts, that may come free, or with little cost. Concepts are related to other concepts; for example, one concept can be a sub-concept (i.e. child) of another concept. Reid has given an example of this with his Parisian-Frenchman-Man-Animal hierarchy.

Concept languages are an attractive destination. And the suggestion of the present text is that symbolic logic that can provide the concept languages. For our purposes, the concepts are going to represent subjects or *topics*. So really the target is topic languages.

[The view that back-of-the-book indexing and subject tagging or annotation are essentially the same was certainly recognized by such pioneers as Rogers, who devised MeSH, the medical subject headings list, and Mortimer Taube, who suggested coordination and postcoordination (Rogers 1960a, b). Rogers and Taube were also both in favor of coordination.

... define coordinate indexing as a system of subject cataloging which capitalizes on the concept of the logical conjunction of ideas (the phrase comes from symbolic logic)... Taube is simply stressing the fact that complex ideas are often best expressed, or even solely expressed, by the intersection of two or more widely separated ideas, and by the intersection, or conjunction, of their separate word symbols when an attempt is made to catalog those ideas. (Rogers 1960b, p. 381)

And they were possibly also both in favor of the use of symbolic logic [‘the logical conjunction of ideas (the phrase comes from symbolic logic)’] and of the use of the Concept Vertex as opposed to the Symbol Vertex.

I cannot believe that the system can ever demonstrate its full potentialities so long as what is being coordinated are just words (Rogers 1960b, p. 383).

]

2.14.1 *Free and Controlled Vocabularies*

There is an issue in information retrieval that arises in many different places (as examples, with keys in indexes, with the insertion of keywords, with subject classification, with metadata, and with tagging) and that concerns *vocabularies*, in particular whether they are *free* or *controlled*.

This issue can be explained in the setting of keywords. Keywords are words that authors and editors attach usually to academic papers to assist with retrieval. In fact, the attaching of keywords is the forming of a kind of index, or the forming of the inverse of an index. If, for example, IO#1 has keywords {Art, Paris}, and IO#2 has keyword {London, Museum, Paris} that generates an association list of IOs and their keywords, with fragment

IO#1	{Art, Paris}
IO#2	{London, Museum, Paris}

And an inversion of that is

Art	{IO#1}
London	{IO#2}
Museum	{IO#2}
Paris	{IO#1, IO#2}

which is a recognizable index.

Words, or collections of words, that get used in these contexts are vocabularies. (Really they are languages, but ‘vocabulary’ is the word most commonly used here.) So, for instance, the collection of words that is used as keys or keywords in a particular journal of academic papers is a vocabulary. There is a choice concerning such vocabularies. Either the authors, editors, and publishers can use whatever words they wish as keywords, in which case the vocabulary is a *free vocabulary*, or

there is a fixed list of proscribed and permitted words that can be used as keywords and the authors etc. have to choose from among them, in which case the vocabulary is a *controlled vocabulary*.

The advantages and disadvantages of controlled vocabularies are easy to summarize. No one knows better than the author of an article what that article is about, so the author is the best person to attach keywords and further the author is best placed to judge exactly what word or words should be the keywords. Score one for free vocabularies. However, this leads to two problems, the patron problem and the several authors problem. The reason there are keywords, indexes, subject classifications etc. is to allow patrons to browse, find, and retrieve IOs. So the keywords, for example, need to be words that patrons or users are familiar with and would use for retrieval. Users are important in the choice of keywords. Seemingly, no one knows better what a searcher are looking for than the actual searcher. Second, if there are two authors each writing a paper on the same topic, and one of them uses one keyword or set of keywords, and the other uses different synonym keywords, then the patron will not be able to retrieve both papers using either authors keywords individually. Here is an example of both these problems. Take the words 'butterflies' and 'rhopalocera'. (As noted, these mean more-or-less the same thing, one is a common everyday word, the other a mildly technical term from biology.) If author one uses the keyword 'rhopalocera', and the patron searches for 'butterflies', the patron will not find the paper (by string match). Recall will be lower than it might have been had the author and the patron been in tune with each other. If author one uses the keyword 'rhopalocera' and author two uses the keyword 'butterflies', then a patron will not find both papers by searching either for 'butterflies' or 'rhopalocera'. Again, recall will be lower than it might have been (and precision might also be adversely affected). The solution to these problems is to use a controlled vocabulary. Such a vocabulary might have the word 'butterflies' in it as a preferred term, but *not* 'rhopalocera'; then all authors writing on this topic *must* use the keyword 'butterflies', and all patrons searching for this topic *must* also use the search term 'butterflies'. Score two for controlled vocabularies. Controlled vocabularies will not usually contain synonyms. For each 'synset' (set or class of synonyms), a choice will be made as to a single *preferred* term or canonical representative for it; and there may be links or references from the other *lead in* terms to the preferred term.

Controlled vocabularies are typically used in conjunction with thesauri (which inform of synonyms, preferred terms etc.). So, for example, if a controlled vocabulary had the term 'butterflies', its associated thesaurus might have an entry

rhopalocera, use butterflies.

Ordinary indexes sometimes mesh together a controlled vocabulary and a thesaurus. For example, a key entry, or fake key, might be 'rhopalocera, see butterflies' (such a key is 'fake' because it is not really part of the association list, rather it is a helpful directive on how to use the association list).

The fact is: controlled vocabularies increase precision and recall. It would take a book-length monograph to argue that point, to cover the different cases and logical and experimental results. But, at the end of the day, almost always controlled vocabularies are better. As Elaine Svenonius writes

Perhaps as near as one can come to generalization about the value of a CV [Controlled Vocabulary] is simply to say where precision and recall are important retrieval objectives, then a CV of some kind is mandated. (Svenonius 2003, p. 837)

We will take that as a given.

2.14.2 *Information Retrieval Thesauri*

Modern information retrieval thesauri date from about the 1960s (Dahlberg 1993; Gilchrist 2003; Evens 2002). In their simplest examples, such thesauri are just lists of preferred terms and their synonyms for use with a controlled vocabulary. However, especially with hierarchical vocabularies, it is possible to pack more information into a thesaurus. There can be information not merely about synonyms, but there can be information about antonyms (love/hate), broader terms (BT), narrower terms (NT), and related terms (RT). So, an entry for ‘butterfly’ might note ‘used for (UF) ‘rhopalocera’ and ‘lepidoptera’ as a broader term (BT) (for lepidoptera include moths as well as butterflies); an entry for ‘yellow’ might note ‘colored’ as a broader term, ‘mustard yellow’ as a narrower term, and ‘jaundiced’ as a related term. There can also be Scope Notes, which are explanations of the usages [See, for example, Lexical FreeNet (Datamuse 2003)]. As Jean Aitchison, Alan Gilchrist, and David Bawden write

[a thesaurus is a] vocabulary of a controlled indexing language, formally organized so that *a priori* relationships between concepts are made explicit (Aitchison et al. 2000, p. 1)

Aitchison, Gilchrist, and Bawden are very much experts in this area, and the phrasing they use here in this quotation is interesting. The important relationships are those between concepts (and concepts occupy the Concept Vertex of the Triangle of Meaning). But the vocabulary (words, phrases, and the like) belong to the Symbol Vertex. So what a thesaurus is doing is giving explicit Symbol Vertex representation to important abstract Concept Vertex relationships.

The plainest thesauri deal for the most part with *single* words or terms, not multi-word phrases or expressions (International Standard ISO 2788). In this, they are like Dictionaries. Occasionally Dictionaries have entries consisting of a couple words or a phrase (e.g. ‘birth control’), and so too do plain Thesauri. But phrases are not the standard form for entry key strings for either Dictionaries or plain Thesauri. Single words are the standard form. The reason for this is that, in their basic core, thesauri are to address the problem of synonyms, and synonyms are largely single words.

In the modern era, thesauri are often just seen as indexing languages. And indexing languages often do need compound terms and sometimes even compound terms of some length. Anthony Foskett mentions a real example, from the *British Technology Index*,

the manufacture of multi-wall kraft paper sacks for the packaging of cement (Foskett 1996, p. 97)

A string like ‘the manufacture of multi-wall kraft paper sacks for the packaging of cement’ can be a subject heading, and so it can be part of a subject indexing language; it thus can also be part of an index-thesaurus; however, it will have no synonyms and no obvious broader or narrower terms so it does not really have any of the special qualities of typical thesaurus entries.

Our sympathies are very much with Aitchison, Gilchrist, and Bawden, but we would give pride of place to conceptual structure. There are concepts, and, for any particular concept in a conceptual scheme, there might be one or more Broader Concepts (BCs), and there might be one or more Narrower Concepts (NCs). Broader concepts and narrower concepts are just inverses of each other. Quite what establishes this broader to narrower link will be discussed in detail later in the book (There might also be Equivalent Concepts (ECs), but they are not so much of interest.). Each concept is an entity unto itself, however, it may be atomic or elemental or it may be complex or compound. Thus the conceptual scheme consists of a Directed Acyclic Graph (DAG) of concepts.

This DAG of concepts can be talked about using such node labels as terms, words, phrases, or symbols. Any concept, atomic or compound, may have a single word term that names it, or short phrases, or long phrases (and there may also be clusters of synonym terms or ‘synonym phrases’ for the node). If the nodes do have terms or short phrases as labels we are in the territory of the classical information retrieval thesaurus. However, even if the labels are long there is still the idea of Broader Phrase (BP) and Narrower Phrase (NP) and these just mean phrases for a broader (narrower) concept.

In sum, the DAG of concepts lurks in the background, and the classical indexing thesaurus has cashed out only some of the value in it.

2.14.3 Tidying up the Nomenclature of Tags and Topics

The word ‘tag’ is in widespread use nowadays, denoting a core element or feature of many popular websites (such as Flickr, for photographs or image, Digg, for endorsement of web-based information items, for Del.icio.us, for ranking items, etc.) (Heymann and Garcia-Molina, 2008, 2009; Heymann et al. 2010). And the word ‘tag’ means a certain sort of thing: essentially a free vocabulary string. The Users are at liberty to make up and use as a tag any word, phrase, or even nonsense string they wish. That the vocabulary or vocabularies are free tends to mean that they are independent of any hierarchical structure or hierarchical thesauri. The tags in common use are annotations using strings from a free vocabulary that does not have a hierarchy.

Topics are also going to be used for annotation. Topics, for us, are concepts. These are going to be described by symbolic logic, and they will have a controlled

form and inter-related structure. Topics can also be described by more ordinary strings. If so, the strings would often be called topic or subject *headings*; and those subject headings would amount to a controlled vocabulary, usually with structure and thesauri-like graph-theoretical hierarchical or other relations among them.

In large part, then, we will use ‘tagging’ to mean free string annotation, and ‘assigning a topic’ (‘topic-ing’?) to mean annotation with controlled concepts, topics, or, perhaps, subject headings.

2.15 The Lesson of the Museum and its Written Guides

There is a central device or distinction within information resources that it is easy to get confused over.

It can be brought to life with the example of museums and literature written about the contents of those museums. Museums classify, catalog and inventory what they have. So, for example, a museum might have sarcophagi—including Egyptian sarcophagi—and coins, including Roman coins. Typically a museum would use a fairly elaborate classification scheme which would include concepts related to other concepts in hierarchies or other graph-theoretical structures. The curators of the museum have an interest in concepts but their principal interest is: which artifacts fall under the denotation of which concepts, for example, in determining which of their items are Egyptian sarcophagi. In imagination, we could certainly suppose that they used their scheme to produce very small labels that they attached or associated with their artifacts; so, for example, an Egyptian sarcophagus might have an ‘Egyptian sarcophagus’ label on it.

In addition to its actual contents, a museum will likely have an Information Desk, Bookstore, or Gift Shop containing written guides about what the museum has. Some of these guides could be more comprehensive than others. Some could be about all the holdings, some could be about coins, some could be about just Roman coins. The visitors have an interest in what the topics or subject matters of those guides are. The museum could certainly indicate topics by sticking exactly the same labels on the pamphlets as they do on the items that are actually in the display halls. So, a guide may have an ‘Egyptian sarcophagi’ label on it. Of course, this does not mean that the guide is an Egyptian sarcophagus, it means that the guide addresses the topic of ‘Egyptian sarcophagi’. The authors of the guides also have an interest in much the same concepts as the museum curators; but they are not interested in those concepts to classify actual relics, they are interested in them as labels of topics. So, now the classificatory concepts, and their labels, are playing double duty. They are identifying things and they are identifying topics. Not only that, the relations between topics is parasitic upon, or symbiotic with, the classification scheme. A reader would generally expect that the topic ‘Roman coins’ would be a sub-topic of the topic ‘Coins’ and this is because Roman Coins are a subclass of Coins.

We are going to have an interest in the various styles of classification of ‘things’ simply because those classifications and classification styles are the inspirations for the topic hierarchies.

But there is a further point of potential confusion. Classical libraries certainly have some commonalities with museums. And librarians have also classified what they have. Most actual existing IO classification schemes use what an IO is *about*, its subject or topic, to classify what the item *is*. In the Dewey Decimal Classification, for example, a book on physics is classified differently to a book on chemistry, and the basis of this difference is that the books are about different subjects.

Books can have many properties. A book might be an encyclopaedia, a dictionary, a monograph, a research study, a conference proceedings, a report, and so on. But in addition to what a book *presents as* there seems to be the question of what a book *is about*. There can be research studies on physics, research studies on juvenile anti-social behavior, and so on. All these IOs are research studies but they are about different subjects. Often the subjects lead. Certainly, there is a lot going on.

The way to be clear here is to rely on Reid. All classification is the possession of attributes (or properties). Attributes of interest within information retrieval include

...is an encyclopaedia
 ...is a research study
 ...addresses the subject ‘physics’
 ...addresses the subject ‘chemistry’
 ...

and so on. And individual IOs possess a selection of these.

References

- Aitchison J, Gilchrist A, Bawden D (2000) Thesaurus construction and use: a practical manual, 4th edn. Fitzroy Dearborn, Chicago
- Anderson JD (1997) Organization of knowledge. In: Feather J, Sturges P (eds) International dictionary of library and information science. Routledge, London, pp 336–353
- Bealer G (1982) Quality and concept. Oxford University Press, Oxford
- Blair A (2003) Coping with information overload in early modern Europe
- Cataloging PaSO (2007) Library of congress subject headings: pre- vs. post-coordination and related issues. http://www.loc.gov/catdir/cpsa/pre_vs_post.pdf.
- Craven TC (1986) String indexing. Academic, Orlando
- Cutter CA (1876) Rules for a printed dictionary catalogue. Government Printing Office, Washington
- Dahlberg I (1993) Current trends in knowledge organization. Paper presented at the first conference on knowledge organization and documentary systems, Madrid
- Daly LW (1967) Contributions to a history of alphabetization in antiquity and the middle ages. Latomus, Brussels
- Datamuse (2003) Lexical FreeNet. <http://www.lexfn.com/>. Accessed 12 Oct 2010
- Dewey M (1926) Dewey decimal classification and relativ index for libraries, 12 edn. Library Bureau, Boston

- Ellis B (2008) Essentialism and natural kinds. In: Psillos S, Curd M (eds) *The routledge companion to philosophy of science*. Routledge, London, pp 139–148
- Evens M (2002) Thesaural relations in information retrieval. In: Green R, Bean CA, Myaeng SH (eds) *The semantics of relationships: an interdisciplinary perspective*. Kluwer Academic Publishers, Dordrecht, pp 143–160
- Foskett AC (1996) *Subject approach to information*, 5th edn. Facet Publishing, London
- Foucault M (1970) *The order of things*. Random House, New York
- Fraser AC (1898) Thomas Reid. Oliphant, Anderson & Ferrier, Edinburgh
- Gilchrist A (2003) Thesauri, taxonomies and ontologies—an etymological note. *J. Documentation* 59(1):7–18
- Haack S (1999) A fallibilist among the cynics. *Skeptical Inquirer* 23(1):47–50
- Heim I, Kratzer A (1998) *Semantics in generative grammar*. Blackwell, Oxford
- Hempel CG (1965) Fundamentals of taxonomy. In: Hempel CG (ed) *Aspects of scientific explanation and other essays in the philosophy of science*. The Free Press, New York, pp 137–154
- Heymann P, Garcia-Molina H (2008) Can tagging organize human knowledge? Technical report. Stanford University, Palo Alto
- Heymann P, Garcia-Molina H (2009) Contrasting controlled vocabulary and tagging: Do experts choose the right names to label the wrong things? WSDM '09, Barcelona, Spain
- Heymann P, Paepcke A, Garcia-Molina H (2010) Tagging human knowledge. In: Third ACM international conference on web search and data mining. New York City, New York
- Jacob EK (2004) Classification and categorization: a difference that makes a difference. *Library Trends* 52(3):515–540
- Koslicki K (2008) Natural kinds and natural kind terms. *Philos Compass* 3(4):789–802
- Kuhn SM (1972) Review of contributions to a history of alphabetization in antiquity and the middle ages by Lloyd W. Daly. *Speculum* 47(2):300–303
- Mann T (1993) *Library research models*. Oxford University Press, New York
- Miksa F (1983) *The subject in the dictionary catalog from cutter to the present*. American Library Association, Chicago
- Mill JS (1843) *A system of logic: ratiocinative and inductive; being a connected view of the principles and evidence and the methods of scientific investigation*. Reprinted in collected works. University of Toronto Press, Toronto, p 1963
- Milstead JL (1984) *Subject access systems: alternatives in design*. Academic, Orlando
- Ogden CK, Richards IA (1972) *The meaning of meaning: a study of the influence of language upon thought and of the science of symbolism*. Harcourt & Brace, New York
- Partee BH, Ter Meulen A, Wall RE (1990) *Mathematical methods in linguistics*. Kluwer Academic Publishers, Dordrecht
- Pliny the Elder (78) *The natural history*. <http://www.perseus.tufts.edu/hopper/text?doc=Perseus:text:1999.02.0137>. Accessed Dec 21 2011
- Popper KR (1972) *Objective knowledge; an evolutionary approach*. Clarendon Press, Oxford
- Ranganathan SR (1937) *Prolegomena to library classification*, 3rd edn 1967; 1st edn 1937. The Madras library association, Madras
- Reid T (1785) Abstraction. In: Bennett JF (ed) *Essays on the intellectual powers of man* (redacted text)
- Rogers FB (1960a) *Medical subject headings*. Preface and introduction. U.S. Department of Health, Education and Welfare, Washington, pp i–xix
- Rogers FB (1960b) Review of taube, mortimer. *Studies in coordinate indexing*. Bull Med Lib Assoc 42 (July 1954): 380–384
- Roget PM (1852) *Roget's thesaurus*. Longman Group Limited, Burnt Mill, Harlow, Essex
- Rosch E (1973) Natural categories. *Cogn Psychol* 4(3):328–350
- Rosch E (1975) Cognitive representations of semantic categories. *J Exp Psychol Gen* 104: 192–233

- Rosch E (1978) Principles of categorization. In: Rosch E, Lloyd B (eds) *Cognition and categorization*. Lawrence Erlbaum, Hillsdale, pp 27–48
- Rosch E, Mervis CB (1975) Family resemblances: studies in the internal structure of categories. *Cogn Psychol* 7(4):573–605
- Rosch E, Mervis CB, Gray W, Johnson D, Boyes-Braem P (1976) Basic objects in natural categories. *Cogn Psychol* 8(4):382–439
- Salton G (1962) Manipulation of trees in information retrieval. *Commun ACM* 5(2):103–114
- Savage E (1946) *Manual of book classification and display*. Allen & Unwin, London
- Sedgewick R (1988) *Algorithms*, 2nd edn. Addison-Wesley, Reading
- Siracusa J (2001) Metadata, the mac, and you. <http://arstechnica.com/apple/reviews/2001/08/metadata.ars>. Accessed 10 Oct 2010
- Siracusa J (2005) Mac OS X 10.4 tiger. <http://arstechnica.com/apple/reviews/2005/04/macosx-10-4.ars>. Accessed 10 Oct 2010
- Smith B (2004) Beyond concepts: ontology as reality representation. In: Varzi A, Vieu L (eds) *Proceedings of FOIS 2004*. International conference on formal ontology and information systems, IOS Press, Turin, pp 73–84
- Soergel D (1974) *Indexing languages and thesauri: construction and maintenance*. Melville, Los angeles
- Svenonius E (ed) (1978) *String indexing*. School of Library and Information Science, The University of Western Ontario, London
- Svenonius E (2003) Design of controlled vocabularies. In: *Encyclopedia of library and information science*. Marcel Dekker, New York, pp 822–838
- Taube M (1951) Functional approach to bibliographic organization: a critique and a proposal. In: Shera JH, Egan M (eds) *Bibliographic organization: fifteenth annual conference of the graduate library school*. University of Chicago Press, Chicago, 24–29 July 1950, pp 57–71
- Taube M (1953) *Studies in coordinate indexing*. Documentation Incorporated, Washington
- Taube M (1961) On the use of roles and links in co-ordinate indexing. *Am Documentation* 12:98–100
- Taube M, Thompson AF (1951) *The Coordinate indexing of scientific fields*. Unpublished paper read before the symposium on mechanical aids to chemical documentation of the division of chemical literature of the American chemical society, 4 Sept 1951
- Taylor AG (2004) *The organization of information*, 2nd edn. Libraries Unlimited, Westport
- Tichy P (1988) *The foundations of Frege's logic*. de Gruyter, Berlin
- Unicode I (1991–2010) The unicode consortium. <http://unicode.org/>. Accessed 10 Oct 2010
- W3C (2010) Large triple stores. <http://esw.w3.org/LargeTripleStores>. Accessed 10 Oct 2010
- Wellisch HH (1981) How to make an index—16th century style: Conrad Gessner on indexes and catalogs. *Int Classif* 8(1):10–15
- Wellisch HH (1983) 'Index' - the word, its history, meanings and usages. *The Indexer* 13(3):147–151
- Wellisch HH (1986) The oldest printed indexes. *The Indexer* 15(2):73–82
- Wellisch HH (1991) *Indexing from A to Z*. H.W.Wilson Co, New York
- Witty FJ (1973) The beginnings of indexing and abstracting: some notes towards a history of indexing and abstracting in antiquity and the middle ages. *The Indexer* 8(4):193–198



<http://www.springer.com/978-1-4614-3087-2>

Logic and the Organization of Information

Frické, M.

2012, XVI, 312 p., Hardcover

ISBN: 978-1-4614-3087-2