Two–Level Models of Hypertext

James Mayfield*
Computer Science Department
University of Maryland, Baltimore County
Baltimore, MD 21228–5398 USA
mayfield@cs.umbc.edu

Abstract

A two–level model of hypertext is one in which a set of texts is augmented with an ancillary structure that captures some aspect of the meaning of those texts. Links between texts are routed through the ancillary structure. This chapter surveys the wide range of two–level hypertext models that have been developed in the past few years. Declarative ancillary structures have included semantic nets, Petri nets, Bayesian nets, and clustering schemes. Procedural ancillary structures have been used to identify new links dynamically, and to create large sets of static links. Advantages of two–level models include greater link expressiveness, increased user control over the appearance and semantics of links, decreased hypertext construction costs, and improved robustness in the face of changing and ill–formed texts.

*This work was sponsored in part by a National Aeronautics and Space Administration Summer Faculty Fellowship.

1 Introduction

In recent years there has been an explosion of interest in models of hypertext that go beyond the explicit hard–wired links found in traditional hypertext models. In these new models, text is augmented with a secondary structure through which all links are routed. This chapter aims to give a broad introduction to these two–level models by examining the general properties of all such models, and by examining particular two–level models in detail.

In this Introduction, I place two–level models in historical perspective by describing the traditional conceptions underlying hypertext and pointing out problems that arise out of reliance on these traditional conceptions. In Section 2, I discuss hypertext models whose secondary structure is declarative in nature. Section 3 explores two–level models in which a process is initiated to traverse a hypertext link. I conclude with some observations on the applicability of two–level models to a variety of new directions in hypertext.

1.1 Traditional Hypertext

The hypertext literature is fraught with conflicting terminology, and even conflict about what does or does not constitute hypertext. While the distinction between hypertext and multimedia systems has blurred in recent years, for the purposes of this chapter **hypertext** will refer to *text* that has been augmented with connections from one piece of text to another. Such connections will be called **hyperlinks**, or **links** for short. The text connected by links is often divided into small pieces, which will be called **nodes** throughout this chapter. The origin of a link is often a small sequence of words in the source node called an **anchor**.

Hypertext usually implies a graphical user interface that allows point–and–click link traversal. This association between the user interface and the underlying hypertext structures is so strong that users often fail to distinguish between the two. For the purposes of this chapter though, I will ignore interface issues; the focus here is on the structures underlying hypertext. In particular, I will focus on the structure and properties of hyperlinks, because it is in links that the major impact of two–level architectures is seen.

The most obvious property of a link that might be influenced by a two–level architecture is its type. Many influential hypertext systems have provided link types. These range from systems that provide very few types (*e.g.* the KMS system [1] and the HDM paradigm [25]) to systems that provide a wide range of link types (*e.g.* the gIBIS [16] system and the TextNet system [55]). Systems with a small number of types often allow the source anchor to provide additional information about a link. For example, KMS has only two link types, but allows an anchor to express the semantics of the link to which it is attached. KMS also allows information about a link to be expressed in the mouse cursor that appears when the mouse is moved over an anchor.

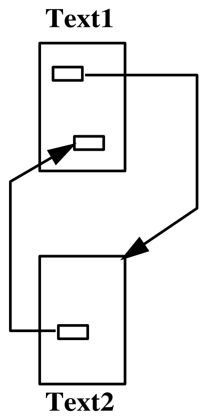

Figure 1: In the traditional hypertext model, links are direct connections from one text or anchor to another.

1.2 Problems with Hypertext Links

A distinguishing feature of traditional hypertext systems is the static nature of their hyperlinks. Static links are *direct* connections from one node or anchor to another. Dynamic links on the other hand are computed each time an attempt is made to traverse them. DeRose calls these two types of links *extensional links* and *intensional links* respectively in his taxonomy of links [19]. Figure 1 depicts static links in this traditional model.

The static links found in most traditional hypertext systems suffer from a variety of problems. These include:

- *Stale links.* When a link points to a node that is subsequently moved or deleted, that link becomes stale; attempts to traverse it will fail. This is not much of a problem for traditional single-author, limited domain hypertexts, because the entire hypertext is typically static. However, as distributed hypertexts lacking clear authorship are developed, stale links are becoming a major problem.
- *Inexpressiveness.* Some conventional system lack link semantics. Even in systems that do provide typed links, if the number of link types is set *a priori*, then the information content that can be attached to a link is restricted to the defined set.
- *Expensive construction.* Although hand–construction of links can provide high–quality connections between nodes (and is in fact for some an essential component of hypertext), it is nonetheless time–consuming and expensive.
- *Inflexibility.* In traditional hypertext, links are created once, and are thereafter fixed; they are unable to rearrange themselves to suit the needs of the moment.
- *Duplication.* The traditional approach leads to a proliferation of links that serve similar purposes. The semantic knowledge implicit in a particular link or link type cannot easily be reused or generalized.

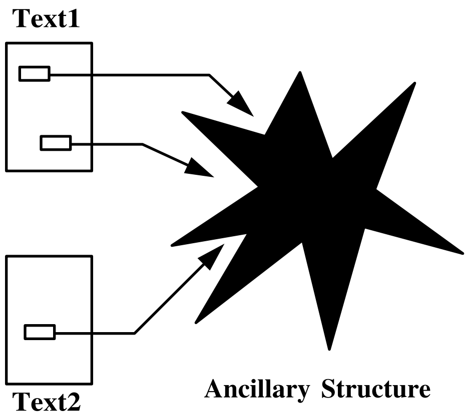

Figure 2: In a two–level model of hypertext, text is augmented with an ancillary structure, through which links are routed.

Many of the two–level hypertext models described in this chapter were developed in response to one or more of these problems.

1.3 Two–Level Models of Hypertext

The main technique underlying two–level models of hypertext is borrowed from software engineering: an abstraction barrier is placed between the user's view of a link and the underlying representation of the link. The abstraction is accomplished by adding a layer of representation between the source of the link and its destination, as shown in Figure 2. This layer can be declarative (*e.g.* a semantic net) or procedural (*e.g.* a WAIS search). I will call this second layer an **ancillary** layer throughout the remainder of this chapter.

While I have made every attempt to present a comprehensive overview of two–level hypertext models in this chapter, I do not claim the coverage presented here to be complete. The reason is that the range of systems that can be construed as 'hypertext systems' is large, and most of those systems can legitimately be placed under the two–level procedural model rubric. By this same reasoning, there is no hard and fast distinction between declarative and procedural two–level models; any so–called 'declarative model' must use some sort of process to traverse its ancillary data structure. The difference between the two types of model then is one of degree; in declarative two–level models the ancillary data structure is the focus of the model, while in procedural two–level models the process that resolves hypertext links is the focus.

2 Declarative Two–Level Models

A declarative two–level model of hypertext is one that augments a set of texts with a knowledge representation structure that captures some aspect of the meaning of the hypertext. A variety of

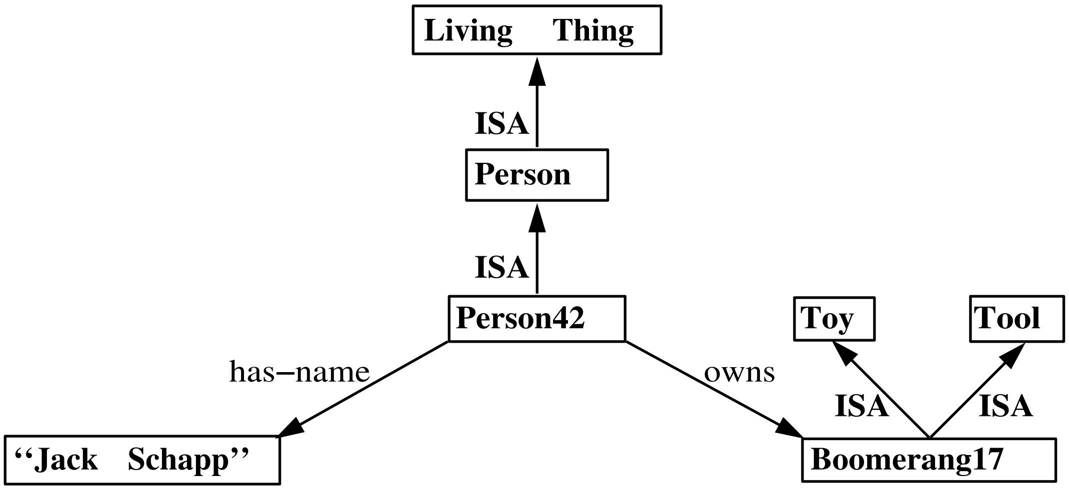

Figure 3: A *semantic net* is a graph in which vertices denote concepts and edges denote relationships between those concepts.

ancillary structures have been used for this purpose; they are described in the following subsections.

2.1 Semantic Net–Based Models

A **semantic net** is a graph whose vertices correspond to concepts (*e.g.* "person" or "boomarang"), and whose edges correspond to relationships between those concepts (*e.g.* "owns"). Most semantic net formalisms support inheritance as well as a variety of other automatic inference processes. Semantic nets are used to represent knowledge about the world and to express the meaning of phrases and sentences of languages such as English. Semantic nets have been studied for over twenty–five years, and a wide variety of semantic net formalisms has consequently been developed [40, 47, 48, 49, 10, 59]. Figure 3 shows a small portion of a generic semantic net.

Many authors have noted the analogy between hypertext and semantic nets, suggesting that the techniques developed for semantic nets are also applicable to hypertext. The reason that this observation is so readily made is that both semantic nets and hypertext are at their core graphs. Rada [41] distinguishes two ways that a semantic net can be used in a hypertext:

> The semantic net of hypertext may be *independent* or *embedded*. In the 'independent' case the nodes and links are tagged with terms. The nodes or links point to text blocks, but the semantic net can be seen without necessarily seeing the text blocks. In the 'embedded' case a text block is at the end of a link. In traversing an embedded semantic net hypertext, the user has to visit a text block. [p. 127]

In Rada's terminology then, the semantic net of a two–level model is typically an *independent* net. Thus, two–level hypertext models based on semantic nets use their semantic net not as a view of the static hypertext graph, but rather as an addition to or replacement for that graph.

Large–Grained Models

One way to use a semantic net as an ancillary structure is to attach entire nodes to vertices or edges in the semantic net. Rada and Wang [41, 58] for example describe how a semantic net can be used to classify portions of text in a document, making it easier to select relevant sub–documents for reuse in other documents. They build a semantic net representing the domain of discourse. Each paragraph of the hyperdocument is then placed on the edge of the semantic net that best expresses the meaning of that paragraph. New documents can be extracted from such a hypertext by performing a traversal (usually a depth–first traversal) of the semantic net. By choosing different starting vertices, and by eliminating undesired vertices and edges from consideration, the user can create documents for a wide range of purposes from a single hypertext.

Another good example of a hypertext system that uses a large–grained independent semantic net as its conceptual model is THOTH–II [15]. THOTH–II models a domain of discourse using a semantic net, then attaches each paragraph of text to a node in the semantic net. Thus, as with Rada and Wang's system, THOTH–II operates at a fairly coarse level of granularity. THOTH–II does allow links from semantic net nodes to particular fixed phrases in the text. These links are syntactic in nature though, because they connect a semantic net node to every location that matches a given string exactly.

Kheirbek [31] uses **Conceptual Graphs** [49] as ancillary structures to alleviate reader disorientation. A Conceptual Graph (CG) is a type of semantic net that has achieved considerable popularity in recent years. Instead of using individual vertices of the semantic net to represent concepts, this approach uses small discrete graphs called *Minimal Canonical Graphs* to represent the concepts of interest. Nodes or (more commonly) aggregates of nodes in the corpus are attached to the Minimal Canonical Graphs that represent their meaning. Under normal operation, the reader uses simple static links to navigate the hypertext. However, the reader also has the option to approach the hypertext through the semantic net if he or she becomes disoriented.

Fine–Grained Models

In contrast to the large–grained use of semantic nets in the above approaches in which entire nodes are associated with vertices or edges of the semantic net, the SNITCH system [34] uses a fine–grained semantic net. In SNITCH, individual words and phrases in a text are connected to semantic net vertices that capture their meaning. Each hypertext link in the SNITCH model comprises a connection from a text node to the semantic net, a path through the hierarchical and relation edges of the semantic net, and an edge from the semantic net back to another text node. Of course, not every such path represents a desirable hypertext link. Therefore, hypertext links are required to conform to particular path shapes. For example, any path that starts from a text anchor, follows a connection to the semantic net, follows an arbitrary number of `SUBPART` edges within the semantic net, and finally traverses another connection to return to a different anchor in the corpus will be a useful path. Hypertext links of that shape will represent connections to subparts of the objects described by their starting anchors.

Hu and Kirstein [29] also use a fine–grained semantic net as their ancillary structure. They focus on connectionist properties of the net to tailor the hypertext to individual users. Each semantic net vertex is assigned an activation level, while each edge is assigned a weight. Weights are then modified to reflect node popularity patterns as the system is used. Vertex activation levels are adjusted in response to the changing weights. When the user selects a link with several possible destinations, these activation levels are used to rank the alternatives.

2.2 Other Net–Based Models

Semantic nets are not the only useful net structure to be selected as the ancillary level of a two–level hypertext model; Petri nets and Bayesian nets have also been used in this capacity.

Petri Nets

A **Petri net** [39, 42] is a directed graph on which a process model has been defined. Figure 4 shows an example of a Petri net. Circular vertices, called **places**, are locations that may hold one or more **tokens**. Rectangular vertices are called **transitions**. When every place that has an edge to a particular transition contains a token, that transition is said to be **enabled**. When a transition is enabled, it can be **fired**; firing consists of removing one token from each place that has an edge to the transition, and placing a token in each place that has an edge from the transition. Execution of a Petri net is simply the repeated firing of transitions until no transition is enabled.

Stotts and Furuta [50, 22] show how a Petri net can be used for access control over the various paths through a hypertext. In their model, which is called TRELLIS, a Petri net is used to augment a hypertext with browsing semantics. For example, the hypertext author can use the Petri net to prevent the reader from accessing a particular node until all of its prerequisite nodes have been perused.

This basic Petri net mechanism can be used to provide a number of novel hypertext features. For example, TRELLIS' timing mechanism, which controls the duration of display of components of the hypertext, can be used to effect an apparent modification of a hypertext over time and for different readers [51]. This can be accomplished while leaving the underlying hypertext unchanged by manipulating the minimum and maximum duration of TRELLIS events. For example, by setting the maximum duration of a button appearance transition[1] to zero, that button is effectively deleted from the reader's view of the hypertext.

[1] A button in Trellis is simply an anchor that can be selected to traverse a hypertext link. In Trellis, the appearance of such a button is an event that can be controlled by Petri net transitions.

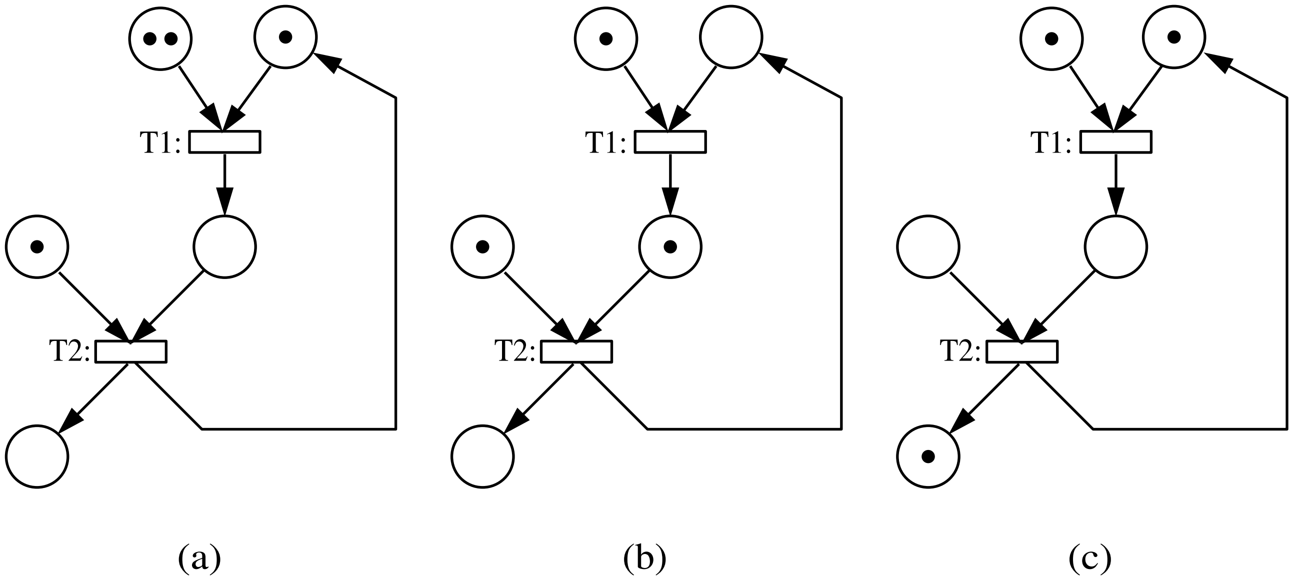

Figure 4: In a *Petri net*, each *place* (represented by a circle) can hold zero or more *tokens* (represented by dots). A *transition* (represented by a rectangle) is enabled when every place that has an edge leading to the transition contains a token. In (a), transition T1 is enabled, while transition T2 is not. Any enabled transition can be *fired* by removing one token from each incoming place, and adding one token to each outgoing place. (b) shows the result after transition T1 has fired. Transition T2 is now enabled; (c) shows the result of its firing.

Bayesian Nets

A **Bayesian net** [13] is a directed acyclic graph whose vertices represent random variables and whose edges represent (roughly) relationships between those variables for which known conditional probabilities exist. Informally, the vertices can be thought of as states of the world, and the edges can be thought of as causality. For example, Figure 5 depicts a Bayesian net (sans probabilities) that captures the relationships between the events of acorns falling from an oak tree, the branches of that oak tree moving, squirrels running on the branches, wind blowing, and the arrival of a storm front. A Bayesian net is used to determine the conditional probabilities of all of the variables (or states of the world) in the network, given that some of those variables (or states of the world) have been observed. To use a Bayesian net in this way, one must know the prior probability of each top–level node (squirrels running and storm coming in this example) as well as the conditional probability of each internal node for every combination of that node's predecessors. Given these probabilities, the Bayesian net in Figure 5 allows one to determine the probability that a storm front is nearing given the observation of a falling acorn.

Croft and Turtle [17] suggest the use of a Bayesian net for hypertext. In their model, a Bayesian net is used to associate index terms (*e.g.* words that might appear in hypertext nodes) with the concepts that those nodes might discuss. An edge from a vertex representing a concept to a vertex representing an index term represents the conditional probability that a hypertext node refers to that concept given that the node contains the index term. A query in this model is a conjunction of propositions that a hypertext node contains particular terms; the output is a list of probabilities

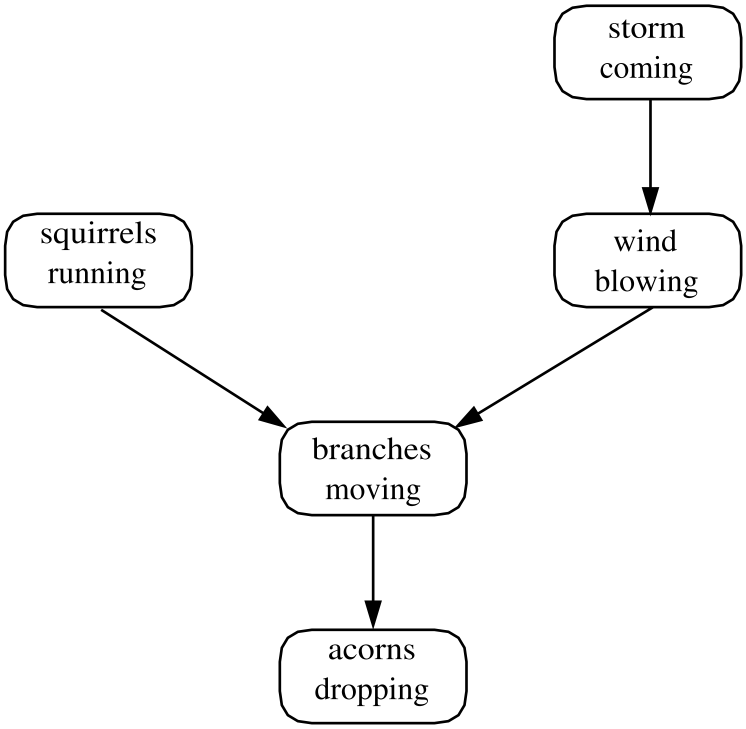

Figure 5: A Bayesian net can be thought of as representing probabilities associated with causal relations. This Bayesian net causally relates five different events; prior and conditional probabilities (which form a key component of every Bayesian net) are not shown.

(one for each document) that the query is satisfied given that the document in question has the desired content.

Frisse and Cousins [20, 21] implement a hypertext system based on a Bayesian net index in their Dynamic Medical Handbook system. In this system, the Bayesian net represents a hierarchical organization of terms that might be found in the nodes of the hypertext. The conditional probabilities of the net, which initially represent even probabilities, can be modified based on reader feedback. The resulting Bayesian net is tailored to the user's interests.

Bayesian nets also form the basis for Savoy and Desbois' two–level model [46]. Their model uses a Bayesian net to represent relationships between the nodes of the hypertext. A user of their system chooses terms of interest in the currently selected node. The system uses these terms as hints about how to update the Bayesian net. The updated net is then used to retrieve relevant documents. Thus, the hypertext reader is able to manipulate the link structure of the hypertext through modifications to the ancillary structure. The adaptive behavior displayed by the Dynamic Medical Handbook and by the Savoy and Desbois systems is especially advantageous in highly–interconnected, multi–author hypertexts.

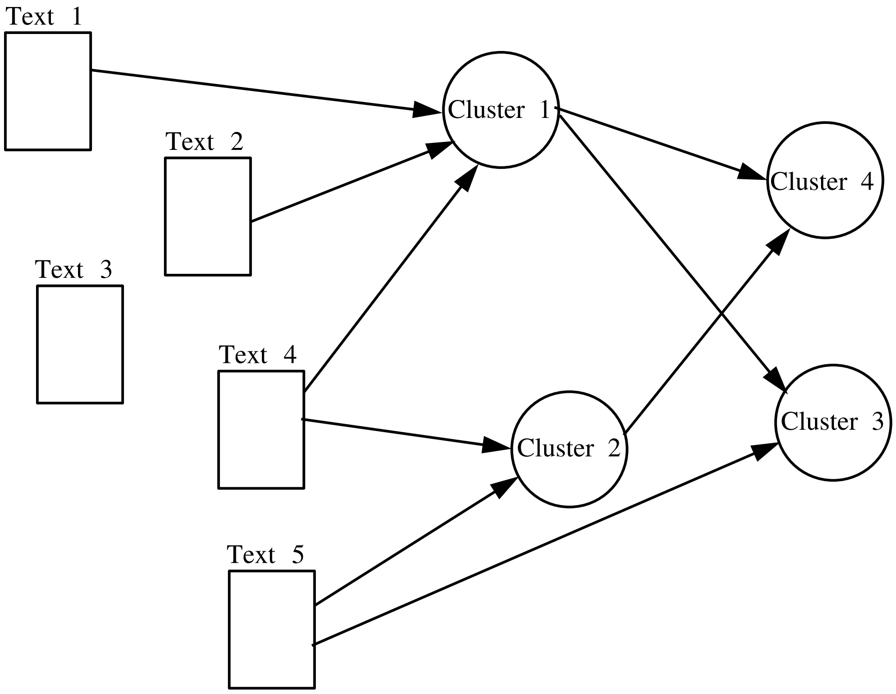

Figure 6: In a *clustering model*, groups of nodes are aggregated to form supernodes, called *clusters*. Clusters may themselves be aggregated to form higher–level clusters.

2.3 Clustering Models

A variety of hypertext models augment the set of nodes with clustering information—information about which nodes are closely related to one another in a given context. These clustering models differ from the above two–level models in that the secondary structure they use is not a formalized representation model that one might study in its own right. Nonetheless, these models do contribute significant power to the hypertext system. Figure 6 depicts a generic clustering model.

Clusters as Secondary Structures

Casanova *et al.* [11] define a formal model for augmenting a hypertext with clustering information. Their model, which they call the *nested context model*, augments a hypertext with *context nodes*; these nodes are used to group sets of nodes, including other context nodes. Context nodes can be nested to any depth. The nested context model also includes formal models for presentation of and navigation through the augmented hypertext. The model is effectively a reference model; no implementation is discussed.

More than one type of cluster might prove valuable in a hypertext system. For example, Thüring *et al.* [52] augment hypertext with three types of organizational nodes: *structure nodes*, which are basic composites of other nodes without traversal restrictions; *sequencing nodes*, which allow the author to specify a particular node traversal sequence; and *exploration nodes*, which aggregate a set of nodes for arbitrary traversal, but which restrict entry and exit to a single access point.

One important use of clustering information is to provide pictorial representations of a user's location within a hypertext. Such graphical interfaces have been explored by a variety of researchers. Trigg [54] for example allows the hypertext author to create *guided tours*, which are graphs of paths through the hypertext which are displayed for reader perusal. In a more automated approach, Utting and Yankelovich [56] provide automatic generation of both global and local maps in the WEB VIEW system. Gloor [26] introduces the idea of a *CYBERMAP*, which is a small map of the hypertext that is constantly displayed as the user traverses the hypertext. To control the complexity of the map, nodes are partitioned into groups of related nodes called *HYPERDRAWERs*. Maintenance of a user model is cited as being crucial to the selection of a set of HYPERDRAWERs that are useful to the user.

The CANTO hypertext data model [35] augments a hypertext with 'concept nodes,' which contain key phrases for the identification of the content of the hypertext nodes. These concept nodes serve to cluster content nodes along taxonomic lines. This use of clusters is shown to increase the speed with which users can find information of interest, while correspondingly decreasing user disorientation and frustration.

Clustering Algorithms

The systems described above show how clustering information can be used to augment the power of a hypertext system. To be practical though, such clustering information must be derived automatically. Botafogo and Shneiderman [9] present algorithms for clustering the nodes of a hypertext. They are interested in finding aggregates of closely related nodes. A *compactness* metric, a function of the average distance between two nodes in the hypertext, is used as the basis for their algorithms' determinations; the lower the average distance, the higher the compactness. A *semantic cluster* of a hypertext is defined to be a subgraph of the hypertext with a higher compactness than that of the entire hypertext. Two algorithms are presented for organizing the nodes of a hypertext into semantic clusters: a biconnected components algorithm for hypertexts with undirected links, and a strongly connected components algorithm for those with directed links.

Crouch *et al.* [18] use *agglomerative hierarchic clustering* [57] to cluster documents for a hypertext system. In this method, each document starts out in a cluster by itself. The two clusters that are most similar (according to some similarity metric) are then aggregated into a single cluster. This process is repeated until a single cluster remains. This pattern of clusters forms a tree, that can then be browsed by the hypertext reader.

Hara *et al.* [28] suggest an approach to clustering called *Aggregation Clustering with Exceptions* (ACE). In this approach, the graph underlying the hypertext is partitioned into a hypergraph representing the clusters, together with an exception graph which represents the difference between the hypergraph and the original graph. Because calculation of an optimum ACE solution is NP–complete, a heuristic algorithm based on the Kernighan–Lin Algorithm [30] is used.

Automated link construction is the creation of links between related nodes by automated content comparison. Because automated link construction methods often invoke some form of node clustering during execution, they are closely related to the approaches described in this section. Automated link construction is discussed in Section 3.2.

3 Procedural Two–Level Models

A procedural two–level model of hypertext is one that eschews direct links in favor of a process that is invoked when an attempt is made to traverse a link. Procedural models have also been called *query–based browsing* [14], *link resolution* models [27, 53], *generalized link* models [8] and *intensional* models [19]. In a sense, every hypertext model is a two–level procedural model, because some notion of process is always required to traverse a link. However, the focus in this section is on two–level models that have an algorithmic rather than a data structure emphasis.

In a procedural two–level model, two flavors of dynamic link are possible. In the first, the target node exists, but the connection from the source node to the target node does not exist until the link generation process is invoked. In the second, neither the connection to the target node nor the target node itself exists until the attempt to traverse the link is made. Because the details of dynamic node generation rely heavily on the subject matter of the hypertext, I will consider only the generation of links here (irrespective of whether the target node exists).

A reference model is "a structured description of some domain that can be used to compare existing implementations in that domain, design new implementations, and...map out possible areas for standardization and show their relation to one another" [36, p. 9]. A number of hypertext reference models have appeared in the literature [23, 24, 27, 32]. By and large though, they skirt the issue of how links connect nodes. The Dexter model [27] includes provision for a procedural model. Direct links are handled by reference to 'Unique Identifiers':

> Every component has a globally unique identity which is captured by its unique identifier (UID). UIDs are primitive in the model, but they are assumed to be uniquely assigned to components across the entire universe of discourse (not just within the context of a single hypertext). The accessor function of the hypertext is responsible for "accessing" a component given its UID, i.e., for mapping a UID into the component "assigned" to that UID. [27, p. 102]

A UID can thus serve as a direct link. Computed links are handled in the model through reference to a 'resolver function;' such a function takes a component specification as its argument and resolves it to a UID. This resolver function is thus the Dexter model's analogue to an ancillary procedural component.

Many procedural models have at their roots research in Information Retrieval (IR). Savoy [45] provides a general introduction to the use of Information Retrieval techniques in hypertext. He also presents an evaluation methodology for the comparison of search techniques. In the following subsections, I first discuss links that are generated on–the–fly. I then survey current methods for automated link construction. These latter techniques differ from dynamic link techniques in that they are applied to an entire corpus; the resulting links are conventional static links. While they do not fall strictly under the rubric of two–level procedural models, systems that do automated link generation are close enough to computed link systems to be included in this survey.

3.1 Computed Links

A **computed link** is a hyperlink that does not exist until a user tries to traverse it. When such an attempt is made (perhaps by clicking on an anchor), a process is generated to compute the target of the link. Furuta and Stotts [23] distinguish three types of processes that might be generated during link traversal: finite processes that require no user input, finite processes that are controlled in part by user input, and continuous processes.

Bieber [6, 7] suggests a generic model for two–level procedural hypertext, which he demonstrates in the MAX system. Bieber's goal is to allow existing systems to be smoothly integrated into a hypertext environment. To that end, he constructs rules for translating the output of the underlying system into a hypertext structure; these rules are called *bridge laws*. Each bridge law is a first–order predicate logic sentence relating particular actions the reader might take in the hypertext system interface with operations of the underlying system. When the reader performs some action, MAX searches its database of bridge laws for laws that match the action. Each applicable bridge law will generate one or more links; the reader may then select the link of greatest interest.

Marshall [33] divides the integration of a hypertext system with an external system into three types: data/content–based integration, in which the external application conforms to the hypertext system's anchoring protocol, thereby allowing external information to be treated as if it were part of the system; tool/node–based integration, in which link traversal corresponds to the retrieval of information from the application; and display/window–based integration, in which the hypertext system simply launches the application. In Marshall's terminology then, MAX provides data/content–based integration for systems that have no awareness of their own participation in a hypertext system.

Another general procedural model is suggested by Tompa *et al.* They view a hypertext as "a collection of documents and link–resolving components" [53, p. 119]. They present an architecture that serves as a framework for procedural models. In their model, each procedure that is able to resolve links registers a specification of the keys it accepts with a central link resolution handler. When a reader selects an anchor, the anchor is translated into a key which is then matched against the registered specifications. For each specification that is matched, a corresponding process is invoked to generate the actual link.

Pearl [38] introduces the Sun LINK SERVICE to handle distributed hypertext links under an open systems approach. The LINK SERVICE provides a link registry that binds keys to applications. An application that can display a particular node identifies itself to the LINK SERVICE, and passes a key representing the node; this key can then be used to retrieve the node. Using this architecture, a system can easily provide computed links by initiating a computation to find the desired node or nodes upon the receipt of a key.

This sort of architecture that maps from a stylized key to a node has gained enormous popularity recently in the form of the World Wide Web (WWW) [5]. WWW provides a syntax for keys called a *Uniform Resource Locator* (URL) [4], together with a protocol called *Hypertext Transfer Protocol* (HTTP) [3] for requesting the node corresponding to a key from a remote host. While a URL can specify a static link to a node, it can also be used to submit a query to an arbitrary program. Thus, WWW seamlessly merges static and dynamic links in a single hypertext model.

3.2 Automated Link Construction

A number of researchers have addressed the problem of automated link construction. While the links that result from such an approach are conventional static links, and all possible links are usually created *en masse*, these approaches nonetheless need to be considered in the context of two–level procedural models.

Chang [12] developed the HieNet system to provide user input to the automated link generation process. In the HieNet system, readers are allowed to create links. The system then infers a *Link Profile* that describes the user–created links, and attempts to instantiate new links that match the profile. When a reader adds a link from node A to node B for example, the system automatically adds links from all the nodes that are similar to node A to all the nodes that are similar to node B. The similarity measure is drawn from Salton's vector space model [43]; the similarity of two nodes is defined as a vector inner product, where the vector terms are the term frequency inverse document frequency (TFIDF) of those words in the node whose term frequency is closest to the median term frequency. In this way, HieNet is able to create only links that are likely to be of interest to the reader.

Allan and Salton [2, 44] describe another approach to automated document linking based on the vector space model. They also use TFIDF to cluster documents. They then divide each document into small pieces (typically paragraphs) and perform clustering analysis at this more local level; doing so helps to resolve ambiguities and eliminates connections that are based on superficial similarities. The resulting clustering information can be used to extract several types of links, including links that suggest a shift in topic, links that indicate a superset/subset relationship between two documents, and links that connect collections of related topics.

The main competitor to the vector space model in Information Retrieval is the Boolean query model. The Queries–R–Links (QRL) system [14] provides a hypertext–like interface to a Boolean query system. Users of the system are allowed to highlight words of interest in the displayed text. If a logical AND operation is desired, words may be connected with lines. The results indicate that Boolean queries are easier for users to formulate using this approach than with a conventional Boolean query language, although the complexity of the queries that can be formulated is not yet up to the level of such languages.

Another technique that is enjoying a revival in Information Retrieval and related fields is n–gram analysis. An n–gram is simply a length n subsequence of the characters in a text. Pearce and Nicholas [37] present a model of text linking based on n–grams. In their model, a cosine measure is used to associate nodes that have similar n–gram profiles. One of the most promising features of this technology is its robustness in the face of misspellings and other forms of text degradation.

4 Conclusions

The power of a two–level model of hypertext comes from the separation of the appearance of a link from the link's underlying structure. Because of this added layer of abstraction, two–level models can provide solutions to a variety of classic problems with hypertext. Beyond their applicability to the problems enumerated in Section 1.2, two–level models have been applied to the problems of hypertext linearization, document reuse, user disorientation, access control, document corruption, hypertext complexity, and hypertext reorganization.

Two–level models of hypertext have applicability to a variety of extensions to the basic idea of hypertext. These include:

- *Adaptive hypertext systems.* An adaptive hypertext system is one that attempt to gauge the needs of the reader and dynamically reconfigure some aspect of the hypertext to meet those needs. A two–level model can facilitate user adaptation by allowing modification of its ancillary structure. Inference on many common ancillary structures is both well–studied and well–behaved; adaptive hypertext systems can be simplified by taking advantage of these properties.
- *Distributed, multi–author hypertext systems.* In hypertext systems that are distributed across sites and across authors (as is the World Wide Web, for example), it is difficult to control and manage the links that are created between nodes. Two–level models provide hope for establishing a degree of order and uniformity to the link structure of such systems.
- *Hypermedia systems.* Hypermedia systems tend to have great difficulty in assigning semantics to non–textual objects. When a media object is attached to an ancillary structure in a two–level model, it can then derive semantics from that ancillary structure.

These are but a few of the benefits that two–level models of hypertext bring to bear on current problems in hypertext system design. Because of their inherent advantages, many more examples of two–level models will emerge in the coming decade.

References

[1] Robert Akscyn, Donald L. McCracken, and Elise Yoder. KMS: A distributed hypermedia

system for managing knowledge in organizations. *Communications of the ACM*, 31(7):820–835, July 1988.

[2] James Allan and Gerard Salton. The identification of text relations using automatic hypertext linking. In James Mayfield and Charles Nicholas, editors, *Proceedings of the Workshop on Intelligent Hypertext*, 1993.

[3] Tim Berners-Lee. HTTP: A protocol for networked information. HTML Document <http://info.cern.ch/hypertext/WWW/Protocols/HTTP/HTTP2.html>, 1994. Work in progress.

[4] Tim Berners-Lee. Uniform resource locators. HTML Document <http://info.cern.ch/hypertext/WWW/Addressing/URL/Overview.html>, 1994. Work in progress.

[5] Tim Berners-Lee. World wide web initiative. HTML Document <http://info.cern.ch/hypertext/WWW/TheProject.html>, 1994.

[6] Michael Bieber. Issues in modeling a "dynamic" hypertext interface for non–hypertext systems. In *Hypertext '91 Proceedings*, pages 203–217. Association for Computing Machinery Press, 1991.

[7] Michael Bieber. Automating hypermedia for decision support. *Hypermedia*, 4(2):83–110, 1992.

[8] Michael P. Bieber and Steven O. Kimbrough. On generalizing the concept of hypertext. *MIS Quarterly*, 16(1):77–93, March 1992.

[9] Rodrigo A. Botafogo and Ben Shneiderman. Identifying aggregates in hypertext structures. In *Hypertext '91 Proceedings*, pages 63–74. Association for Computing Machinery Press, 1991.

[10] Ronald J. Brachman and James G. Schmolze. An overview of the KL–ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[11] Marco A. Casanova, Luiz Tucherman, Maria Julia D. Lima, Jose L. Rangel Netto, Noemi Rodriguez, and Luiz F. G. Soares. The nested context model for hyperdocuments. In *Hypertext '91 Proceedings*, pages 193–201. Association for Computing Machinery Press, 1991.

[12] Daniel T. Chang. HieNet: A user–centered approach for automatic link generation. In *Hypertext '93 Proceedings*, pages 145–158. Association for Computing Machinery Press, 1993.

[13] Eugene Charniak. Bayesian networks without tears. *AI Magazine*, 12(4):50–63, 1991.

[14] Nipon Charoenkitkarn, Jim Tam, Mark H. Chignell, and Gene Golovchinsky. Browsing through querying: Designing for electronic books. In *Hypertext '93 Proceedings*, pages 206–216. Association for Computing Machinery Press, 1993.

[15] George H. Collier. Thoth–II: Hypertext with explicit semantics. In *Hypertext '87 Proceedings*, pages 269–289. Association for Computing Machinery Press, 1987.

[16] Jeff Conklin and Michael L. Begeman. gIBIS: A hypertext tool for exploratory policy discussion. *ACM Transactions on Office Information Systems*, 6(4):303–331, October 1988.

[17] W. Bruce Croft and Howard Turtle. A retrieval model for incorporating hypertext links. In *Hypertext '89 Proceedings*, pages 213–224. Association for Computing Machinery Press, 1989.

[18] Donald B. Crouch, Carolyn J. Crouch, and Glenn Andreas. The use of cluster hierarchies in hypertext information retrieval. In *Hypertext '89 Proceedings*, pages 225–237. Association for Computing Machinery Press, 1989.

[19] Steven J. DeRose. Expanding the notion of links. In *Hypertext '89 Proceedings*, pages 249–257. Association for Computing Machinery Press, 1989.

[20] M. E. Frisse. Searching for information in a hypertext medical handbook. *Communications of the ACM*, 31:880–886, 1988.

[21] Mark E. Frisse and Steve B. Cousins. Information retrieval from hypertext: Update on the dynamic medical handbook project. In *Hypertext '89 Proceedings*, pages 199–212. Association for Computing Machinery Press, 1989.

[22] Richard Furuta and P. David Stotts. Programmable browsing semantics in Trellis. In *Hypertext '89 Proceedings*, pages 27–42. Association for Computing Machinery Press, 1989.

[23] Richard Furuta and P. David Stotts. A functional meta–structure for hypertext models and systems. *Electronic Publishing—Origination, Dissemination and Design*, 3(4):179–205, November 1990.

[24] Richard Furuta and P. David Stotts. The Trellis hypertext reference model. In Judi Moline, Dan Benigni, and Jean Baronas, editors, *Proceedings of the Hypertext Standardization Workshop*, pages 83–93. National Institute of Standards and Technology, 1990. NIST Special Publication 500–178.

[25] Franca Garzotto, Paolo Paolini, and Daniel Schwabe. HDM—a model for the design of hypertext applications. In *Hypertext '91 Proceedings*, pages 313–328. Association for Computing Machinery Press, 1991.

[26] Peter A. Gloor. CYBERMAP: Yet another way of navigating in hyperspace. In *Hypertext '91 Proceedings*, pages 107–121. Association for Computing Machinery Press, 1991.

[27] Frank Halasz and Mayer Schwartz. The Dexter hypertext reference model. In Judi Moline, Dan Benigni, and Jean Baronas, editors, *Proceedings of the Hypertext Standardization Workshop*, pages 95–133. National Institute of Standards and Technology, 1990. NIST Special Publication 500–178.

[28] Yoshinori Hara, Arthur M. Keller, and Gio Wiederhold. Implementing hypertext database relationships through aggregations and exceptions. In *Hypertext '91 Proceedings*, pages 75–90. Association for Computing Machinery Press, 1991.

[29] Michael J. Hu and Peter Kirstein. An intelligent hypertext system. In James Mayfield and Charles Nicholas, editors, *Proceedings of the Workshop on Intelligent Hypertext*, 1993.

[30] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49(2):291–307, February 1970.

[31] Ammar Kheirbek. A two–level hypermedia model based on conceptual graph theory. In James Mayfield and Charles Nicholas, editors, *Proceedings of the Workshop on Intelligent Hypertext*, 1993.

[32] Danny B. Lange. A formal model of hypertext. In Judi Moline, Dan Benigni, and Jean Baronas, editors, *Proceedings of the Hypertext Standardization Workshop*, pages 145–166. National Institute of Standards and Technology, 1990. NIST Special Publication 500–178.

[33] Catherine C. Marshall. A multi–tiered approach to hypertext integration: Negotiating standards for a heterogeneous application environment. In Judi Moline, Dan Benigni, and Jean Baronas, editors, *Proceedings of the Hypertext Standardization Workshop*, pages 167–177. National Institute of Standards and Technology, 1990. NIST Special Publication 500–178.

[34] James Mayfield and Charles Nicholas. SNITCH: Augmenting hypertext documents with a semantic net. *International Journal of Intelligent and Cooperative Information Systems*, 2(3):335–351, 1993.

[35] Charles K. Nicholas and Linda H. Rosenberg. Canto: A hypertext data model. *Electronic Publishing—Origination, Dissemination and Design*, 6(2):93–113, June 1993.

[36] H. Van Dyke Parunak. Reference and data model group (RDMG): Work plan status. In Judi Moline, Dan Benigni, and Jean Baronas, editors, *Proceedings of the Hypertext Standardization Workshop*, pages 9–13. National Institute of Standards and Technology, 1990. NIST Special Publication 500–178.

[37] Claudia Pearce and Charles Nicholas. Generating a dynamic hypertext environment with n–gram analysis. In Bharat Bhargava, Timothy Finin, and Yelena Yesha, editors, *Proceedings of the Second International Conference on Information and Knowledge Management*, pages 148–153. Association for Computing Machinery Press, 1993.

[38] Amy Pearl. Sun's link service: A protocol for open linking. In *Hypertext '89 Proceedings*, pages 137–146. Association for Computing Machinery Press, 1989.

[39] James L. Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice–Hall, 1981.

[40] M. Ross Quillian. Word concepts: A theory and simulation of some basic semantic capabilities. *Behavioral Science*, 12(5):410–430, September 1967. Reprinted in Ronald J. Brachman and Hector J. Levesque, editors, *Readings in Knowledge Representation*, pages 98–118. Morgan Kaufmann, 1985.

[41] Roy Rada. Hypertext writing and document reuse: The role of a semantic net. *Electronic Publishing—Origination, Dissemination and Design*, 3(3):125–140, August 1990.

[42] Wolfgang Reisig. *Petri Nets: An Introduction*. Springer–Verlag, 1985.

[43] Gerard Salton. *Automatic Text Processing*. Addison–Wesley, 1989.

[44] Gerard Salton and James Allan. Selective text utilization and text traversal. In *Hypertext '93 Proceedings*, pages 131–144. Association for Computing Machinery Press, 1993.

[45] Jacques Savoy. Effectiveness of information retrieval systems used in a hypertext. *Hypermedia*, 5(1):23–46, 1993.

[46] Jacques Savoy and Daniel Desbois. Information retrieval in hypertext systems: An approach using Bayesian networks. *Electronic Publishing—Origination, Dissemination and Design*, 4(2):87–108, June 1991.

[47] Roger Schank and Robert Abelson. *Scripts, Plans, Goals and Understanding.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1977.

[48] S. C. Shapiro. The SNePS semantic network processing system. In N. V. Findler, editor, *Associative Networks: Representation and Use of Knowledge by Computers*, pages 179–203. Academic Press, 1979.

[49] J. F. Sowa. *Conceptual Structures: Information Processing in Minds and Machines.* Addison Wesley Publishing Co., 1984.

[50] P. David Stotts and Richard Furuta. Petri–net–based hypertext: Document structure with browsing semantics. *ACM Transactions on Information Systems*, 7(1):3–29, January 1989.

[51] P. David Stotts and Richard Furuta. Dynamic adaptation of hypertext structure. In *Hypertext '91 Proceedings*, pages 219–231. Association for Computing Machinery Press, 1991.

[52] Manfred Thüring, Jörg M. Haake, and Jörg Hannemann. What's Eliza doing in the Chinese Room? Incoherent hyperdocuments—and how to avoid them. In *Hypertext '91 Proceedings*, pages 161–177. Association for Computing Machinery Press, 1991.

[53] Frank Wm. Tompa, G. Elizabeth Blake, and Darrell R. Raymond. Hypertext by link–resolving components. In *Hypertext '93 Proceedings*, pages 118–130. Association for Computing Machinery Press, 1993.

[54] Randall H. Trigg. Guided tours and tabletops: Tools for communicating in a hypertext environment. *ACM Transactions on Office Information Systems*, 6(4):398–414, 1988.

[55] Randall H. Trigg and Mark Weiser. TEXTNET: A network–based approach to text handling. *ACM Transactions on Office Information Systems*, 4(1):1–23, 1986.

[56] Kenneth Utting and Nicole Yankelovich. Context and orientation in hypermedia networks. *ACM Transactions on Information Systems*, 7(1):58–84, January 1989.

[57] E. M. Voorhees. *The Effectiveness and Efficiency of Agglomerative Hierarchic Clustering in Document Retrieval.* PhD thesis, Department of Computer Science, Cornell University, Ithaca, New York, 1985.

[58] Weigang Wang and Roy Rada. A semantic net for reusable hypertext. In James Mayfield and Charles Nicholas, editors, *Proceedings of the Workshop on Intelligent Hypertext*, 1993.

[59] Robert Wilensky. Some problems and proposals for knowledge representation. In Janet L. Kolodner and Christopher K. Riebeck, editors, *Experience, Memory, and Reasoning*, pages 15–28. Lawrence Erlbaum Associates, New Jersey, 1986.