A Comparison of Algorithms for Hypertext Notes Network Linearisation

MIKE SHARPLES[1] ANDREW CLUTTERBUCK[2] JAMES GOODLET[1]

[1]*School of Cognitive and Computing Sciences*
University of Sussex
[2]*School of Humanities and Education*
University of Hertfordshire

New computer-based writing environments are being developed which combine a hypertext 'ideas organiser' with a text editor. We compare two algorithms which could be used in such environments for turning networks of notes representing ideas into linear draft documents. The algorithms treat the notes network as a directed, labelled graph and they are designed to produce a linear ordering of the notes which is acceptable to the writer as a first draft of the document. The experiments test the effectiveness of the algorithms by asking subjects to create notes networks which are then linearised by the two algorithms. The resulting linearisations, plus a random ordering of nodes and a linearisation created by hand, are assessed for textual organisation. The experiments indicate that both algorithms produce linearisations which are acceptable as draft texts, that the 'best first' algorithm is marginally superior, and that providing information about link types to the algorithms has little effect on their effectiveness. The paper concludes by describing an implementation of the best first algorithm as part of the Writer's Assistant writing environment.

1. Introduction

Hypertext was invented (Bush, 1945; Nelson, 1987) as a means to browse or search through a large set of conceptually inter-related texts, by following trails of association. The reader of a hypertext has the power to choose a path through the network, but at a cost of having to select one link from many to follow at each point of choice. There are occasions where a reader will not have the knowledge to make an informed choice of hypertext link, or where the reader is content to let an author determine a good route through the material. In these cases, where the source material is held as a hypertext, but the reader would be best served by a linear presentation, a linearisation algorithm can assist in turning a hypertext into a linear text.

A different use of hypertext is in a computer-based writing environment, where a writer creates a hypertext notes network as an aid to instantiating and exploring ideas, and may then linearise some or all of the notes to form a first draft of the text.

In this paper we compare two algorithms for linearising hypertext networks and describe how hypertext linearisation has been incorporated into a computer-based writing environment.

2. Hypertext linearisation

There are two main occasions where hypertext linearisation can be of use: in supporting a reader browsing through a hypertext database, and in providing a version of a hypertext in linear form.

2.1 Supporting browsing

A reader browsing through a hypertext may not always wish to choose, or be able to make, an informed choice of which path to follow. A hypertext display will typically show the names of the successor nodes, but will not tell the reader whether a link is important, nor whether it leads to a major new part of the network or just to a dead end. A linearisation algorithm could provide a default linear path for the hypertext reader, selected according to criteria such as the reader's interests (for example, a path containing chosen topics), the reader's abilities (for example, a path containing a explanations and digressions matched to the reader's knowledge), or a given point of view (for example, material chosen to represent one side of an argument).

2.2 Hypertext to linear text

There are many reasons to print out a copy of a hypertext database, such as to check it for completeness, to provide a paper copy in a report or dissertation, or to produce a book version of a hypertext reference work. The printed version may still indicate the links between nodes as conventional text references, but the hypertext must be linearised to fit the format of a printed work, with pages collated into sequential order. A simple algorithm, such as listing the nodes in order of time of creation, or in alphabetic order, may be acceptable for some purposes, but others, such as producing a printed reference book with a preferred linear reading, may require a more sophisticated linearisation.

Linear texts have served as the main means of storing and transmitting knowledge over the past five thousand years and there is no sign that printed books, magazines and journals are about to be superseded. For most purposes, there is no need to present the reader with a hypertext; writers want to be in control of the order in which a text is read, and readers are content to be carried along by a narrative flow.

One use of hypertext is, instead of giving choice to a reader, to assist the author of a conventional text, by providing 'mind maps' (Buzan, 1989), or 'notes networks' (Trigg & Irish, 1987; Sharples et al., 1991) which act as 'intermediate representations' between mental associations and draft text. A productive way for a writer to generate new material is to follow mental trails of association, setting down each new idea as a note, and linking the notes together to form a visual map of the topics to be covered. The use of notes networks as an aid to study and writing has been advocated by Buzan (1989), Trigg and Suchman (1989) and others. Creativity comes, in part, from exploring a 'conceptual space' of related ideas (Boden, 1990). An interlinked network of notes representing ideas or topics can make this creativity visible and explicit.

Xerox NoteCards (Trigg & Irish, 1987) is a computer-based implementation of a notes network. A user can 'brainstorm' ideas, write each one on a simulated file card, and link the cards together graphically on the screen. Two or more people can work with the same set of NoteCards, putting their thoughts down on the screen and then linking, organising and sorting them until the ideas merge. Although NoteCards has been used as an 'ideas organiser' by writers there is no easy way to move between the network of notes and a linear text.

Buzan suggests that linearisation by hand is an easy matter:

> Many people, when first shown Mind Maps, question if they can be used for any linear purpose, such as giving a talk or writing and article. … All that is necessary is to decide the final order in which to present the information. A good Mind Map will offer a number of possibilities, When the choice is being

> made, each area of the Mind Map can be encircled with a different colour and numbered in the correct order. Putting this into written or verbal form is simply a matter of outlining the major areas to be covered, and then going through them point by point, following the logic of the branched connections. (Buzan, 1989, p. 107)

For a network of ten or twenty nodes, fitting neatly into one page or screen, then “following the logic of the branched connections” may be quite straightforward, but for large and strongly inter-connected networks (and some NoteCards networks contain over 500 nodes) then computer-assisted linearisation may offer a faster and more tractable alternative to linearisation by hand.

The Writer’s Assistant (Sharples, Goodlet, & Pemberton, 1992) is a prototype writing environment which combines an ideas organiser with a document structure editor and a text editor. A writer can map out ideas and topics in the Notes Network view, can create a hierarchical document structure in the Structure view and work with linear text in the Linear view. An aim of the Writer’s Assistant is to support the easy movement between all the views. To move from Notes Network to Linear view requires creating a linearisation of the hypertext notes and imposing this on a suitable document structure.

A linearisation algorithm can automate the task of constructing a linear text from a hypertext. In this paper we compare two algorithms for linearising hypertext networks which were developed for the Writer’s Assistant. Although we concentrate on the issues of providing hypertext as part of a computer-based writing environment, the algorithms have more general applicability.

2.3 Requirements of a general-purpose linearisation algorithm

This section sets out some requirements of a usable general-purpose linearisation algorithm.

It can be applied to any hypertext network, cyclic or acyclic, with or without labelled arcs.

Different techniques of idea mapping use different presentations: NoteCards allows the writer to construct directed cyclic labelled graphs, while Mind Maps have a presentation in which concepts are attached to lines, so that the networks reduce to a directed cyclic unlabelled graph. The linearisation algorithm must be able to operate on an undirected, unlabelled network, but be able to take advantage of link information if available.

Given one or more start nodes, it is guaranteed to construct a list containing all and only the nodes reachable from the start nodes.

The algorithm must be able to operate from any chosen start node or nodes and to span the entire set of reachable nodes.

It is guaranteed to terminate, having added each node to the linear list once only.

This condition can be relaxed if there is a good reason to repeat a node at more than one place in the linear list (for example if they are widely separated in a large list).

The time taken to traverse the network should be acceptable to the writer, for networks of the order of hundreds of nodes.

An acceptable time for traversal will depend on the application and context of use, but the time should be of the order O(n) where n is the number of nodes in the network. For large networks it should take appreciably less time than linearisation of the network by hand.

It makes use of information contained in the network to guide linearisation.

The algorithm should not require the hypertext author to add any additional information to the network to guide linearisation, but should make use of whatever information is recorded during the creation of the network. Information potentially available to the algorithm includes node names, node contents, link names, connectivity, time of creation and spatial layout.

It should be deterministic

The order of linearisation should be fully determined by the information contained in the network, not by the processing priorities of the programming language.

It should produce a linearisation which is acceptable to a human writer.

A linearisation algorithm differs from a search algorithm in that the criterion for success is not to reach a specified goal, but to produce an order of nodes which meets the expectations of a reader. A coherent 'easy' narrative leads the reader onwards by following trains of association that resonate with the reader's experience. Thus, it follows that a linearisation algorithm can only be judged by subjective criteria.

3. Descriptions of the algorithms

The experiments described in this paper compare two algorithms for hypertext linearisation. Both algorithms satisfy the requirements above, apart from the 'deterministic' criterion. (The best first algorithm has four heuristics to resolve similarity of links, and other information such as the time of node creation can be employed to ensure the algorithm is deterministic.) They treat a hypertext as a directed, labelled graph. They ignore the content of the nodes (and so can also be used to linearise a network containing non-textual elements), but make use of the 'priority' of the link type. The algorithms have both been implemented in Prolog.

Both algorithms can be implemented to run in time $O(n\ l)$ where n is the number of nodes in the network and l is the mean number of links from each node. Although l will depend on the size of the network, a writer will typically segment a large notes network into topic clusters so that, in general, nodes will be locally inter-connected. Thus, for large networks, $l \ll n$.

3.1 Link types

The nodes in a hypertext carry content information and the links indicate conceptual relations between the contents. The algorithms below do not assume any specified set of link types, but they are informed by ordering the link types in terms of priority of traversal, where high priority links form the 'core' of the information. Core links may vary with the type of text, so that an expository text may have 'causes' or 'explains' as a high priority link, whereas a descriptive text may prioritise the 'describes' link. Low priority links connect incidental material with labels such as 'comments' or 'for example'. One or more link types may share a priority value, and an unlabelled network is just a special case in which all links are given the same value.

3.2 Algorithm 1: Hillclimbing linearisation

This algorithm was first outlined in O'Malley and Sharples (1986). From each node, starting with the entry node, it selects the next node to visit by following the lowest priority link. Any links to nodes already visited are discarded. If it reaches a leaf node, it backs up and chooses the next priority link from the last choice node. The strategy is similar to that of a human reader who makes a systematic tour of a hypertext by reading the incidental material around each hypertext node before moving down the core path to the next main topic.

The algorithm uses only local knowledge to push forward the linearisation, but it can be extended by a heuristic which, when there is more than one link with the same priority value, chooses the link which leads to the smallest subtree (i.e. small sub-topics leading off a main node are included before large ones). This extension was not implemented for the experiments described below.

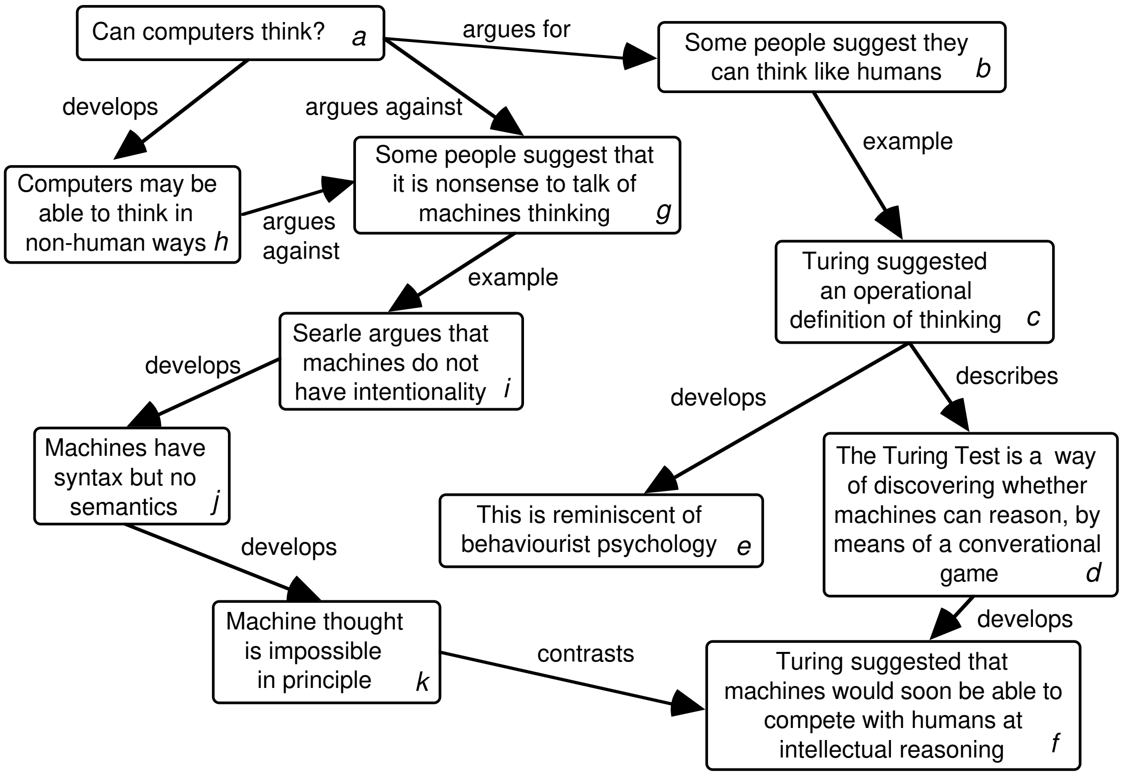

Can computers think? a
argues for
Some people suggest they can think like humans b
develops
argues against
example
Computers may be able to think in non-human ways h
argues against
Some people suggest that it is nonsense to talk of machines thinking g
example
Turing suggested an operational definition of thinking c
Searle argues that machines do not have intentionality i
develops
develops
describes
Machines have syntax but no semantics j
This is reminiscent of behaviourist psychology e
The Turing Test is a way of discovering whether machines can reason, by means of a converational game d
develops
develops
Machine thought is impossible in principle k
contrasts
Turing suggested that machines would soon be able to compete with humans at intellectual reasoning f

Figure 1. A small Notes Network

This algorithm copes well with weakly interconnected notes networks such as the one shown in Figure 1, but because links from only one candidate node are considered at each step, this can cause problems with certain network topologies.[1]

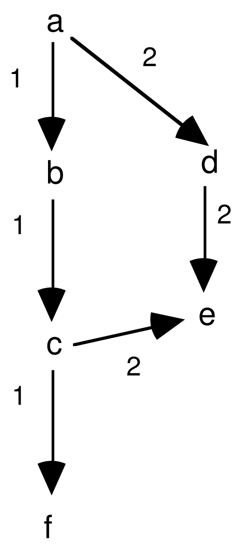


Figure 2. The 'Annotation' problem

[1] In all the networks shown in this paper, a link value of 1 is given to the highest priority link. The larger the link value, the lower the priority.

The ‘annotation problem’ occurs when a ‘comment’ or other low priority node is indirectly linked to a main node near the entry point (i.e. it is a ‘comment’ on a ‘comment’), and that same comment is directly linked to a node further from the entry (see Figure 2, where node ‘e’ is the ‘comment’ node). Generally, a good linearisation should place a comment beside the main node it annotates, so the network in Figure 2 should produce [a d b c e f], however the hillclimbing algorithm linearises the network to [a d e b c f].

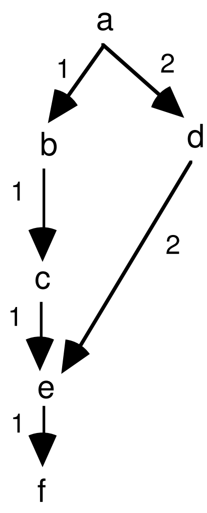


Figure 3. The ‘Short Cut’ problem

The ‘short cut’ problem is when there are alternative paths from a node forming ‘tributaries’ of differing importance which then rejoin (see Figure 3). The hillclimbing algorithm follows the minor tributary to the end, and then backs up to include the remaining nodes on the core path. It produces the linearisation [a d e f b c], whereas a more suitable linear chain would be [a d b c e f].

3.3 Algorithm 2: Best First linearisation

This algorithm maintains an OPEN list of candidate *nodeA–link–nodeB* triples, where *nodeA* is already on the LINEARISED list. The algorithm places the entry node, or nodes, on the LINEARISED list. At each step, it finds all the untraveled links from the nodes that have been newly added to LINEARISED. It removes these links from the graph and merges them with OPEN in order of link priority, with the highest priority (i.e. lowest value) link at the front of OPEN. It then removes the *nodeA-link-nodeB* item from the front of OPEN and, providing *nodeB* is not already on LINEARISED, it inserts *nodeB* into LINEARISED in a position after *nodeA*.

The effect of the algorithm is travel along the most promising paths first, at each step pushing forward the linearisation from the node on the current linearised list which has the highest priority link leading from it. Since the links from nodes placed on LINEARISED are compared with all the links on OPEN at each step, there is a high probability of two links having the same value. Heuristics can be used to resolve the clashes, and Figure 4 gives the full algorithm, as implemented for the experiments.

Create two simple lists, OPEN and LINEARISED, initially empty.

1. Begin at the entry node, or nodes, in the hypertext graph.

2. Place the entry nodes on LINEARISED.

3. Find all untravelled links from each node in LINEARISED. Remove each link from the graph.

4. Merge the links with OPEN, lowest value link to the front.

4a. If there are two or more candidate links with the same value, then put them on OPEN in order of the size of subgraph leading from the link, largest subgraph to the front. (The size of the subgraph is calculated on the pruned graph, with nodes already on LINEARISED removed.)

4b. If there are two or more candidate links with the same value, and same size of subgraph, then put them on OPEN in order of the value of the link leading to the node from which the links depart (lowest value to the front).

4c. If there are two or more candidate links, with the same value, size of subgraph, and value of incoming link, then put them on OPEN in order of the distance of the link from the start node (furthest from the start node to the front).

4d. If there are two or more links at the same distance from the start node, and one or more is already on OPEN, then put any new links in front of the one(s) already on OPEN.

4e. If there are still two or more candidate links, then put them on OPEN in some order determined by information contained in the network (such as the time the node was created).

5. If OPEN is empty and not all nodes have been removed from the graph, then reverse all the remaining links in the graph. Go to 3.

6. If OPEN is empty then stop.

7. Remove the link at the front of OPEN.

8. Call the node from which this link departs the FOCUS NODE and the node to which the link points the SUCCESSOR NODE.

9. If the SUCCESSOR NODE is already on LINEARISED then go to 5.

10. Add the SUCCESSOR NODE to LINEARISED in position immediately after the FOCUS NODE.

11. Find all untravelled links from the SUCCESSOR NODE.

12. Go to 4.

Figure 4. The Best First algorithm

The heuristics are designed to favour the choice of high priority links which lead to or from larger (and therefore more likely to be important) sub-parts of the network. Heuristic 4a requires the size of the subgraph from a node to be computed, but the computation can be bounded without significantly affecting the operation of the algorithm. Line 5 allows for networks where some nodes cannot be reached due to the direction of the links. (It is needed because occasionally a subject connected a cluster of nodes to the main network with a link in the reverse direction.) Adding all reachable nodes to the linear list and then reversing all the remaining links has the effect of including the remaining nodes in the linearisation, but at low priority.

The best first algorithm overcomes the particular problems of the hillclimbing algorithm, producing a linearisation of [a d b c e f] for the network in Figure 2, and [a d b c e f] for the network in Figure 3. It also has the advantage of filling the LINEARISED list in order of link priority, so that, by giving a cut-off value for the link priority, it can filter out parts of the hypertext network, retaining only those nodes on main paths.

3.4 The algorithms in operation

To give an example of the algorithms in operation, Figure 1 shows a small notes network produced by a writer on the topic of 'Can computers think?'. Algorithm 1 builds up the linearised list in the order shown in Figure 5.

```
[a]
[a h]
[a h g]
[a h g i]
[a h g i j]
[a h g i j k]
[a h g i j k f]
[a h g i j k f b]
[a h g i j k f b c]
[a h g i j k f b c d]
[a h g i j k f b c d e]
```

Figure 5. Order of nodes produced by the hillclimbing algorithm for the network in Figure 4

The linearised text corresponding to the final order of nodes is as follows:

> Can computers think?
>
> Computers may be able to think in non-human ways.
>
> Some people suggest that it is nonsense to talk of machines thinking.
>
> Searle argues that machines do not have intentionality.
>
> Machines have syntax but no semantics.
>
> Machine thought is impossible in principle.
>
> Turing suggested that machines would soon be able to compete with humans at intellectual reasoning.
>
> Some people suggest computers can think like humans.
>
> Turing suggested an operational definition of thinking.
>
> The Turing Test is a way of discovering whether machines can reason, by means of a conversational game.
>
> This is reminiscent of behaviourist psychology.

```
[a]
[a g]
[a b g]
[a h b g ]
[a h b c g]
[a h b c e g]
[a h b c e g i]
[a h b c e g i j]
[a h b c e g i j k]
[a h b c e g i j k f]
[a h b c d e g i j k f]
```

Figure 6. Order of nodes produced by the best first algorithm for the network in Figure 4

Algorithm 2 creates the linearised list in the order shown in Figure 6, and the final order of nodes produces the linear text below:

> Can computers think?
>
> Computers may be able to think in non-human ways.
>
> Some people suggest computers can think like humans.
>
> Turing suggested an operational definition of thinking.
>
> The Turing Test is a way of discovering whether machines can reason, by means of a conversational game.
>
> This is reminiscent of behaviourist psychology.
>
> Some people suggest that it is nonsense to talk of machines thinking.
>
> Searle argues that machines do not have intentionality.
>
> Machines have syntax but no semantics.
>
> Machine thought is impossible in principle.
>
> Turing suggested that machines would soon be able to compete with humans at intellectual reasoning.

The ordering of algorithm 2 provides a more plausible framework for a linear text, which the writer could then flesh out with connecting phrases to create a first draft:

Can computers think?

> Computers may be able to think in non-human ways, *but* some people suggest computers can think like humans. Turing suggested an operational definition of thinking. The Turing Test is a way of discovering whether machines can reason, by means of a conversational game. This is reminiscent of behaviourist psychology. *However,* some people suggest that it is nonsense to talk of machines thinking. Searle argues that machines do not have intentionality. Machines have syntax but no semantics. *He concludes that* machine thought is impossible in principle, *whereas* Turing suggested that machines would soon be able to compete with humans at intellectual reasoning.

The algorithms have been tested with a set of networks, representing prototypical idea structures, including *argument structure*, *inverted pyramid*, and *narrative with descriptive clusters*. For example, Figure 7 shows a prototypical argument structure consisting of background information, definition of issues, statement of thesis to be proven, arguments for and against the thesis, refutation of opposing arguments and summation. With no link information (i.e. all the links set to a priority of 1), both algorithms produce the linearisation [a b c d e f g h i j k l m n o p q r]. When the links indicating transitions between topics (e.g. c $\rightarrow$ g and i $\rightarrow$ o) are given a lower priority than links within topics (e.g. b $\rightarrow$ c and m $\rightarrow$ n) then Algorithm 1 gives the linearisation [a b e h i j k l m q r p n o f g c d] and Algorithm 2 gives the linearisation [a b c d e f g h i j k l m n o p q r].

For the algorithms to be useful in a computer-based writing environment it is not necessary that they should produce perfect linearisations, or even ones as good as by a human, but they should produce an order which is acceptable as a first draft of a linear text. The test cases indicate that both algorithms produce useful linearisations, but that Algorithm 2 gives the most consistently acceptable linear order. The experiments described below were designed to compare the algorithms on more natural data, produced from hypertexts constructed from published texts, and from notes networks produced by experimental subjects in a writing activity.

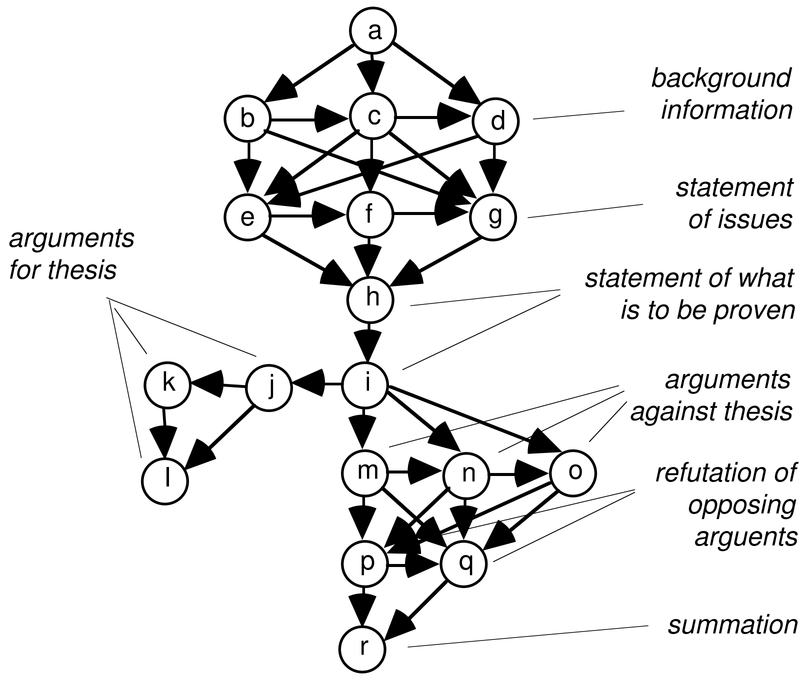


Figure 7. A prototypical notes network for an argument structure

4. Experiments

The experiments test the ability of the algorithms to produce an acceptable linearisation of hypertext notes networks. They compare the two algorithms with each other, with human hand linearisation and with a random linear order. Each algorithm is tested in two conditions: with the algorithm taking account of link information, and with link values ignored.

4.1 Experiment 1

3.1.1 Rationale

We would expect a hypertext which is constructed by hand from chunks of a published linear text to contain links which represent explicitly the conceptual relationships implicit in the original text. A successful linearisation algorithm should be able to traverse the hypertext structure, using the information in the links to reconstruct the order of the original text. The experiment examines this suggestion by asking subjects to form a hypertext from chunks of a published text. Each hypertext is then given to the four linearisation algorithms (hillclimbing and best first, with and without link information) and the resulting linear texts are rated by a human evaluator for textual organisation on a five point scale.
It was also scored by a 'least cost restoration' algorithm for closeness of the reconstructed order to the original text.

Hypothesis 1: There will be a correlation between the scores produced by the 'least cost restoration' algorithm and those of the human evaluator.
Hypothesis 2: The original linear texts will have significantly higher ratings than the best method of automatically linearised texts.
Hypothesis 3: All the automatically linearised texts will have significantly higher ratings than the random orderings.
Hypothesis 4: The best first algorithm with labelled links will produce significantly higher ratings than the hillclimbing algorithm with labelled links.

Hypothesis 5: The best first algorithm with labelled links will produce significantly higher ratings than the best first algorithm with no link information.

3.1.2 Subjects

Twelve subjects participated in the experiment (5 women and 7 men). They were all employed at the University, in jobs including administration, postgraduate research and teaching. Their ages ranged from 22 to 35. None of the subjects had previously used hypertext for idea formation or writing.

3.1.3 Materials

The text types were chosen to represent four of the five prototypical 'writing plans' identified by Meyer, Young and Bartlett (Meyer, Young, & Bartlett, 1989): description (describing events, ideas, or objects); sequence (concepts grouped on the basis of order or time); causation (causal relations between concepts); comparison (relates concepts by similarity or difference); problem/solution (concepts grouped into a problem part and a solution part which responds by trying to eliminate the problem). The four texts were abridged from published articles and consisted of a description (a newspaper story describing a robbery), a sequence (a text describing the sequence of activities of a snake shedding its skin), causation (advice to bird watchers about clothes and equipment) and comparison (a newspaper feature comparing methods of food storage). Each text was chunked into t-units. (A t-unit broadly represents a topic item. More precisely, it is a clause plus its attached modifiers (Hunt, 1965).) The referents were substituted wherever anaphoric reference occurred so that the meaning of each t-unit depended as little as possible on its context. Each chunk was written on a 9cm x 5cm file card. The number of chunks for each text were 20 for the description, 18 for the sequence, 20 for the causation and 15 for the comparison.

Throughout the experiment the subjects were shown a card listing the 13 link types that they could use in constructing the hypertexts and which the algorithm used to set the link priorities (see Table 1). The priorities differ for each type of text: 'problem' and 'solution' are given a high priority for the Comparison text type, 'describes' has a high priority for the Description text types, and so on. Nodes that are connected by high priority links will not necessarily appear close together in the linearised text; rather, these nodes are used to construct a 'backbone' ordering and the other nodes are then inserted between them.

	Sequence	Comparison	Causation	Description
argues against	7	2	7	5
argues for	7	2	7	5
causes	6	2	2	4
comments	8	5	8	8
compares	4	2	3	6
contrasts	4	2	3	6
describes	3	4	4	2
develops	2	3	2	2
example	5	4	6	7
explains	3	3	2	3
problem	6	2	5	4
solution	6	2	5	4
then	1	1	1	1

Table 1. The link priorities for each text type.

There is no generally-agreed set of basic link types, and hypertext systems which provide pre-specified links range from gIBIS with eight link types intended for developing argumentation (Halasz and Conklin, 1990) to Trigg's TEXTNET (Trigg & Weiser, 1986) with over eighty different link types. The set of links chosen for the experiment was intended to be small enough to be managed by the experimental subjects, but large enough to cover the main types of conceptual relation for the text types used in the experiment. The subjects were instructed to say during the experiment if they required any further link types, and at the end each subject was asked to suggest further link types which they might have found useful. No subject indicated that an additional link type would be necessary to complete the task. Three subjects suggested links which they might have found useful (each of the three people suggested two or more links). These were 'undermines', 'context', 'co-description', 'but', 'new episode', 'facilitates', 'suggests'.

3.1.4 Design

A repeated measures design was used, with each subject producing a hypertext for each the four texts. The order of texts was counter-balanced.

3.1.5 Procedure

Each subject was shown a list of link types and the experimenter explained the meaning of each of the link types. The subject was also shown an example hypertext (Figure 1). The subject was then given the set of cards containing the text chunks for the first example text, as well as the text itself. The text was available for reference during the experiment. The subject was asked to stick the cards onto a whiteboard and to use a board marker to draw in relational links. Each link should have an arrow indicating its direction and a label chosen from the set of available link types. The subject was encouraged to use whichever strategy seemed natural to construct the hypertext. Some subjects placed all the cards on the board and then drew in the links; others added links after placing each card. Subjects were allowed as much time as they wished to carry out the task. The experimenter recorded the layout of the hypertext on paper as the subject created it. When the subject was satisfied that the hypertext was complete the experimenter removed it from the board and gave the subject the next set of cards. The experiment ended when the subject had created four hypertexts.

The hypertexts produced by the subjects were linearised by the two algorithms, firstly with link information available, and then with link information removed. The resulting linear texts were printed in a standard format, as were the chunks in the order of the original published linear texts, and the four texts formed by randomising the order of the chunks. The resulting 200 linear texts (12 subjects * 4 texts * 4 linearisations, plus the four original orderings and four random texts) were given in random order to a marker to score blind on a five point scale for 'organisation': Totally disorganised, Largely disorganised, Somewhat organised/ Somewhat disorganised, Acceptably organised, Perfectly organised. The texts were also scored using an algorithm which computed the cost of reordering the nodes to restore the original published text.

4.2 Results of Experiment 1

All the subjects were able to complete the task. Table 2 shows the evaluator's scores for the algorithms applied to the four types of text: sequence (seq), comparison (cmp), causation (cau), description (des), plus the mean of the four scores (mn). All the original linear texts were rated 5 by the evaluator and all the randomly organised texts were rated 1.

	Hillclimbing Labelled					Best First Labelled					Hillclimbing Unlabelled					Best First Unlabelled				
	seq	cmp	cau	des	*mn*	seq	cmp	cau	des	*mn*	seq	cmp	cau	des	*mn*	seq	cmp	cau	des	*mn*
S1	2	5	2	4	3.25	4	3	3	5	3.75	2	5	2	5	3.5	4	5	4	5	4.5
S2	3	4	3	2	3.0	3	4	2	4	3.25	2	3	2	4	2.75	3	4	3	2	3.0
S3	2	3	3	1	2.25	2	3	2	3	2.5	2	1	2	2	1.75	2	3	3	3	2.75
S4	3	5	2	5	3.75	4	5	2	3	3.5	2	5	2	4	3.25	4	5	2	4	3.75
S5	3	4	3	4	3.5	3	4	4	3	3.5	2	2	3	3	2.5	5	2	1	3	2.75
S6	2	4	2	4	3.0	4	4	3	4	3.75	2	1	3	4	2.5	2	4	3	3	3.0
S7	2	3	2	2	2.25	3	2	2	4	2.75	2	3	3	3	2.75	5	3	3	5	4.0
S8	2	4	2	4	3.0	3	5	2	5	3.75	2	4	2	4	3.0	2	5	2	5	3.5
S9	3	3	3	2	2.75	4	2	3	3	3.0	2	1	3	3	2.25	4	2	2	4	3.0
S10	2	3	2	2	2.25	2	4	1	5	3.0	1	3	2	2	2.0	3	3	2	4	3.0
S11	3	2	2	2	2.25	4	3	2	2	2.75	2	2	3	1	2.0	5	3	3	1	3.0
S12	3	1	1	5	2.5	3	2	2	5	3.0	1	1	1	5	2.0	2	2	3	5	3.0
MN	2.5	3.42	2.25	3.08	2.81	3.25	3.42	2.33	3.83	3.21	1.83	2.58	2.33	3.33	2.52	3.42	3.42	2.58	3.67	3.27

Table 2. The evaluator's scores for the linearised text produced by the algorithms.

Although there is considerable variation in the scores among the text types, this is not surprising given the nature of the texts. The descriptive text (a newspaper report) was on a single topic and part of it consisted of a set of descriptive statements that could be arranged in any order without altering the coherence of the text, whereas the causation text had both a global structure, with distinct sub-topics, and a local ordering of causation and implication. For comparison of the algorithms, the scores have been averaged across the text types.

	Hillclimbing Labelled		Best first labelled		Hillclimbing unlabelled		Best first unlabelled	
	Human Eval.	Least Cost	Human Eval.	Least Cost	Human Eval.	Least Cost	Human Eval.	Least Cost
S1	3.25	3.4	3.75	2.7	3.5	3.1	4.5	3.6
S2	3.0	2.9	3.25	3.3	2.75	2.6	3.0	3.1
S3	2.25	2.6	2.5	3.3	1.75	2.4	2.75	3.3
S4	3.75	3.5	3.5	4.1	3.25	3.4	3.25	4.1
S5	3.5	3.0	3.5	3	2.5	2.7	2.75	3.1
S6	3.0	3.3	3.75	3.6	2.5	2.8	3.0	2.4
S7	2.25	3.1	2.75	3.2	2.75	3.0	4.0	3.7
S8	3.0	3.3	3.75	3.6	3.0	3.2	3.5	3.5
S9	2.75	2.1	3.0	2.6	2.25	2.1	3.0	2.7
S10	2.25	3.1	3	3	2.0	3.3	3.0	3.4
S11	2.25	2.1	2.75	1.8	2.0	2.6	3.0	2.8
S12	2.5	1.9	3	1.8	2.0	2.1	3.0	1.9
Mean	2.82	2.86	3.21	3.00	2.52	2.77	3.23	3.13

Table 3. The mean scores across the text types produced by the human evaluator, and by the least cost restoration algorithm.

Table 3 shows the mean scores for the human evaluator and the scores produced by the least cost algorithm (the algorithm produced scores in the range 6 to 14, these have been normalised to ease comparison in the table). There was a significant correlation between the scores ($r_S = -0.61$, $p < 0.01$).

All the original texts were rated 5 by the evaluator. The highest score for any of the linearisation algorithms is 3.83 (for the best first algorithm applied to the labelled Description hypertext). The randomly ordered texts were all rated 1. The lowest score for any of the algorithms is 1.83 (for the hillclimbing algorithm applied to the unlabelled Sequence hypertext).

The mean score for the best first algorithm applied to the hypertexts with labelled links is 3.21, and the corresponding mean score for the hillclimbing algorithm is 2.81. A Wilcoxon test ($N = 12$, adjusted for ties) shows the difference between the algorithms to be significant at $p < 0.005$ (one tailed).

The scores for best first algorithm applied to the hypertexts with the labelled links (3.21) and to the hypertexts with the link values removed (3.27) are almost identical.

4.3 Discussion

The good correlation between the ratings of the human evaluator and the least cost restoration scores indicates that computing the restoration distance of a linearised text from the original text is a useful means of measuring the effectiveness of linearisation and could provide a comparative test of new linearisation algorithms. Any new algorithm could be applied to the hypertexts used in this experiment and the linearisations could be reliably compared with those of the two algorithms tested here.

As expected, the best first algorithm was significantly more effective at linearising the hypertexts than the hillclimbing algorithm, and the mean score of 3.21 (above the 'somewhat organised, somewhat disorganised' level) suggests that it produced texts with sufficient organisation to be useful as first drafts. The low score of 2.33 for the Causation hypertext suggests that automatic linearisation may be less useful for hypertexts which contain a number of distinct, but related, topics. One approach may be to group the nodes into topics by hand and then apply a version of the algorithm which keeps together text on the same topic.

The unexpected finding was that removing information about link types did not alter the effectiveness of the linearisation. There a number of possible explanations for this result. The subjects may not have had enough practice in creating hypertexts to be able to put appropriate labels to the links, or the range of link types may not be adequate to indicate the conceptual links in the text, or the algorithm may not make good use of link information due to poor heuristics, or the priorities assigned to the link types may be inappropriate. (It would be possible to investigate the latter explanation by giving the results produced by the computer evaluation of the linearised texts as input to a learning algorithm which determines the optimal priority for each link type).

In this experiment the subjects created hypertexts from published linear texts by applying their skills of reading comprehension to make explicit the referential links embedded in the text. The experiment measured how effective the linearisation algorithms are in selecting links and traversing them in an appropriate order. But creating a hypertext as part of writing is not quite the same activity. A writer, in producing a notes network, is following trails of mental association, with no textual cues for guidance. A hypertext produced during the writing process may have links indicating deep conceptual relations, rather than surface textual ones, and may thus be more difficult for an algorithm to linearise. The second experiment tests this possibility.

4.4 Experiment 2

4.4.1 Rationale

The aim of the experiment is to test the algorithms on hypertext notes networks generated as part of a writing activity. It differs from experiment one in that the subjects are generating their own hypertexts on a given topic. The assumption is that the hypertext acts as a means of 'externalising cognition', allowing the writer to represent a pattern of mental associations between topic items. The two linearisation algorithms were compared against human linearisation, carried out by the authors of the hypertexts, and a random ordering of nodes. The linearisations were scored by two human assessors on a five point scale for textual organisation.

Hypothesis 1: The hand linearised texts will have higher ratings than the best method of automatic linearisation.
Hypothesis 2: All the automatically linearised texts will have higher ratings than the random orderings.
Hypothesis 3: The best first algorithm with labelled links will produce higher ratings than the hillclimbing algorithm with labelled links.
Hypothesis 4: The best first algorithm with labelled links will produce higher ratings than the best first algorithm with no link information.

4.4.2 Subjects

The subjects were the same as for experiment one.

4.4.3 Materials

The subjects were shown the same card of link types as for experiment one. Each subject was given a stock of blank 9cm x 5cm file cards on which to write the text. The materials for creating the hypertexts were as for experiment one.

4.4.4 Design

Each subject produced one hypertext.

4.4.5 Procedure

The subjects were given a list of three topics and asked to choose one topic on which they would create a hypertext. The topics were 'How to choose a summer holiday', 'Should I sell my car and cycle to work' and 'The role of Britain in Europe'. Four of the subjects chose the Holiday topic, seven chose the Bicycle topic, and one chose the Europe topic.

Each subject was given a stock of twenty blank file cards and was asked to generate short sentences on the topic, writing each sentence on a separate card. The subject was asked to produce the sentences as rapidly as possible, without considering how they might be ordered in a finished piece of prose. When the twenty cards had been filled out (some subjects asked for one or two extra cards to complete their train of ideas) the subject was asked to form them into a hypertext by sticking them onto the whiteboard and drawing in arrowed links, with labels chosen from the list of link types. The subjects were encouraged to use their own strategies to produce the hypertexts, as if they were creating notes networks as part of a writing activity. When the subject was satisfied that the hypertext was complete he or she was then asked to remove the cards from the board in an order that would produce a useful first draft of a linear written text.

Each hypertext was assigned by the experimenter to one of the four text types according to the text type and the type of links used. Nine were judged to be 'comparison', two were judged to be 'sequence' and one was judged to be 'causation'. The twelve hypertexts were coded and linearised by the two algorithms, with and without link information. The resulting 48 linear texts, plus twelve texts produced by printing the nodes in random order, and twelve produced by the subjects' hand linearisation, were randomly ordered and given to the two independent evaluators. The evaluators marked the texts using the same five point scale as in the previous experiment. For 15 out of the 72 texts the marks differed by two scale points, and for two texts the markers differed by three scale points. The evaluators were asked to re-mark these 17 texts without reference to the previous scores. The re-marking produced six cases where the evaluators differed by more than one scale point.

4.5 Results of Experiment 2

Table 4 shows the scores (after re-marking) of the two evaluators for each of the 72 texts produced by the linearising algorithms, random ordering, and hand linearisation.

	Hillclimbing Labelled		Best First Labelled		Hillclimbing Unlabelled		Best First Unlabelled		Random		Hand Linearised	
	Eval1	Eval2	Eval1	Eval2	Eval1	Eval2	Eval1	Eval2	Eval1	Eval2	Eval1	Eval2
S1	4	3	4	3	5	4	5	4	4	1	5	4
S2	4	4	4	3	5	3	3	3	2	1	4	4
S3	3	4	4	3	3	4	2	3	1	1	5	5
S4	4	4	3	4	4	4	3	4	2	1	5	4
S5	4	1	2	3	4	1	3	3	1	1	4	3
S6	4	3	5	3	4	3	4	4	2	1	4	4
S7	2	3	2	2	2	3	2	3	1	1	4	4
S8	2	3	4	5	1	3	4	5	1	2	5	5
S9	2	2	3	3	3	3	3	3	1	2	5	4
S10	4	4	3	4	3	4	3	4	1	1	5	5
S11	3	4	5	4	4	4	4	5	1	2	5	5
S12	2	2	3	3	3	4	4	4	2	2	4	5
Mean	3.17	3.08	3.50	3.33	3.42	3.33	3.33	3.75	1.58	1.33	4.58	4.33
Mean	3.13		3.42		3.38		3.54		1.46		4.46	

Table 4. Scores of the two evaluators for the linearised texts.

The correlation between the scores of the two evaluators is r_S=0.63, significant at p<0.01. The mean scores of the two markers for each of the text types (see Figure 8) is hillclimbing labelled: 3.13; best first labelled: 3.45; hillclimbing unlabelled: 3.38; best first unlabelled: 3.54; random: 1.46; hand linearised: 4.46. All the automatically linearised texts have lower ratings than the texts linearised by hand, and all are higher than the random orderings. The differences between the scores are significant in both cases at $p < 0.005$ (Wilcoxon, N=12, one tailed). The two scores produced by the best first algorithm are higher than those for the hillclimbing one but the differences are not significant. As for experiment one, there is no significant difference between the scores for the best first algorithm with label information and without it.

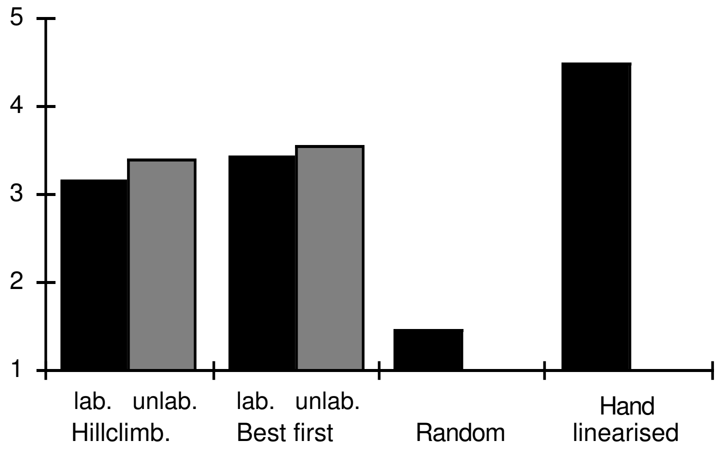
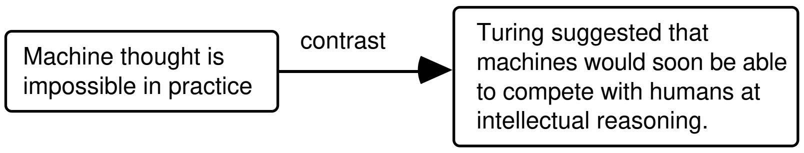


Figure 8. Mean scores for the linearised texts

4.6 Discussion

Both algorithms produce linearisations in the range between 'somewhat organised/somewhat disorganised' and 'acceptably organised'. It indicates that automatic linearisation could provide a useful bridge between an ideas organiser and a text editor as part of a writing environment. The linear text may well need to be edited, but the algorithms produce enough partial ordering to provide an initial draft text. There is little difference between the results of the algorithms but, as for experiment 1, the best first algorithms produce marginally higher scores. Given that the best first algorithm is as easy to apply as the hillclimbing one, and that it overcomes some specific problems in test cases (see section 2.3), then it is the better for most purposes.

As for experiment 1, there is little difference between the best first algorithm applied to networks with labels and without them. Although this may be due to there being an inadequate set of link types, the subjects expressed no difficulty in showing conceptual relations using the available links. Another possibility is that the subjects may not have been skilled enough to apply the links in a way that added meaning to the network structure. They had all produced four previous hypertexts as subjects in Experiment 1, but generating appropriate and consistent links may take more practice. The most important factors appear to be that the algorithm was poorly able to make use of the semantics of the links to guide linearisation and that the semantic relations of the links are not apparent in the linearised text. Some links, such as 'example, and 'contrasts' show relations which cannot be expressed by simply concatenating the chunks of text. So, for example, the relation


Machine thought is impossible in practice
contrast
Turing suggested that machines would soon be able to compete with humans at intellectual reasoning.

is not adequately expressed by "Machine thought is impossible in practice. Turing suggested that machines would soon be able to compete with humans at intellectual reasoning." The algorithm may be more successful if connecting phrases like 'by contrast' and 'for example', were inserted into the linearised text to substitute for those links which indicate textual connectives.

There are other ways in which the algorithm could be made more effective. The most obvious one is to make use of information contained within the nodes. In a more formal domain, such the design of instructional guides, text within a note (such as ‘first’ and ‘secondly’) might indicate a partial ordering. More generally, similarity of keywords in the notes might be used to form clusters which should appear together in the linear list. Alternatively, the writer could explicitly group together nodes into clusters, or could indicate partial orderings of nodes (this can be done in the Writer’s Assistant by the writer linking notes into the linear document structure, see below). Lastly, the spatial layout of the nodes could be used to guide linearisation. This method is used in the StorySpace (Bolter et al., 1992) program, which provides facilities for creating hypertext notes networks and linearises the notes by simply traversing the screen from top left to bottom right. The problem with relying on spatial layout is that writers generally do not plan notes networks to fit a space, but grow them by association and when the space becomes cluttered, adding new notes wherever there is some free space. Buzan’s Mind Maps are generally shown growing outwards from a central node, while other notes networks (such as the one in Figure 1) grow downwards and sideways. If the writer could specify the ‘direction of growth’ of a network then this might be of some assistance in linearisation.

5. An application of the algorithm

The best first algorithm has been implemented as part of the Writer’s Assistant writing environment (Sharples, et al., 1992). It links the Notes Network view, where a writer can set down and organise ideas as written notes, with the Structure view, for creating hierarchical document structures, and the Linear view, for linear text editing. A writer who wishes to move from ideas organising to structuring or writing simply selects another view and the system converts the material to structure or linear form. Practical application in a writing environment has required some minor changes and extensions to the best first algorithm as described above.

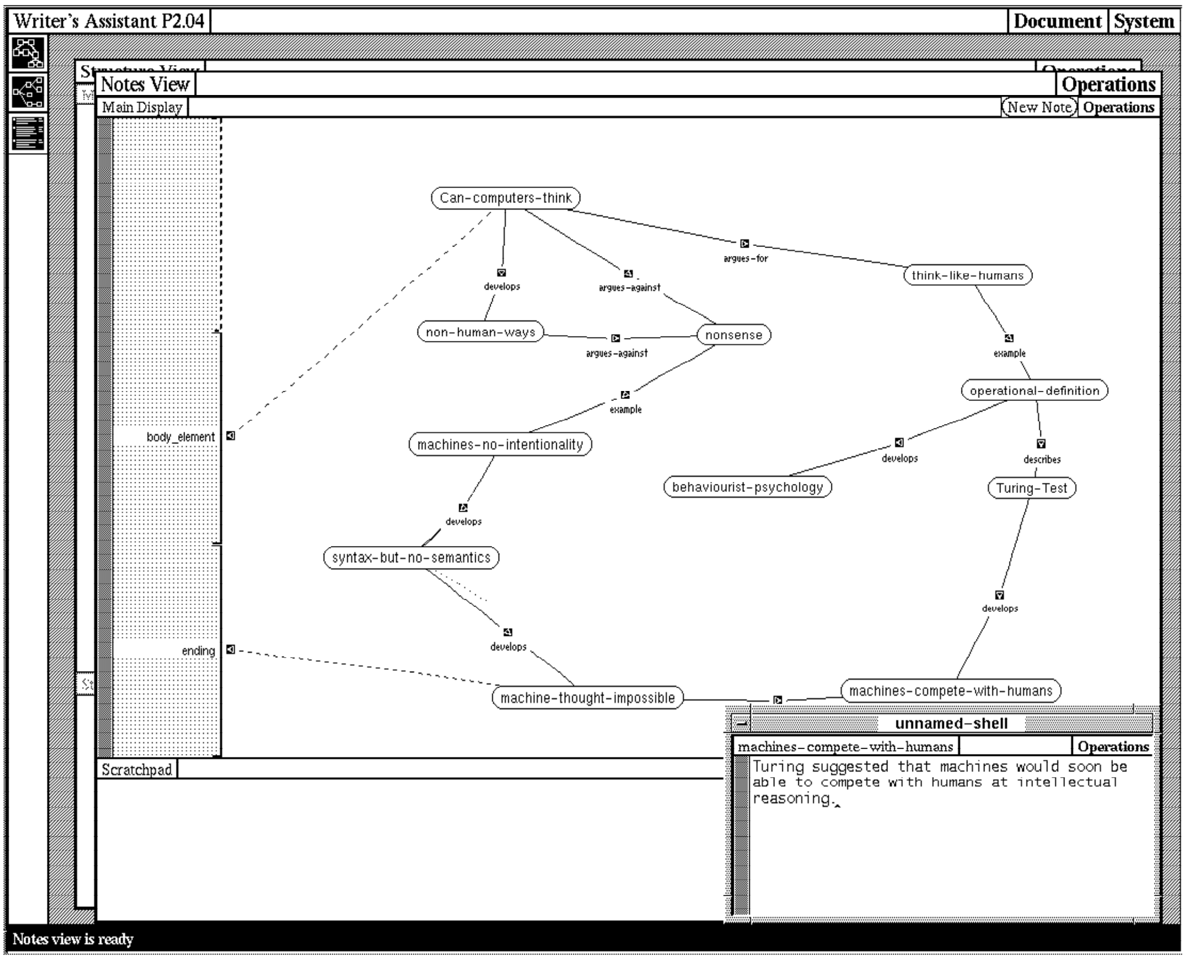
5.1 Multiple start nodes

Although the experimental subjects indicated a single starting point for each of their networks, the algorithm outlined in Figure 4 can linearise a network with more than one start node, and this is a valuable feature of the Writer’s Assistant.

Idea networks for practical documents often contain clusters of notes indicating topics and a writer may wish to indicate in which order the clusters should appear in the document, or to associate a cluster with a specific part of the document, for example to ensure that a particular set of notes forms the conclusion of a report. The Writer’s Assistant allows a writer to ‘pin’ notes to particular points in the document’s hierarchical structure, using a special ‘structure link’. Figure 9, a Notes Network view, shows a structure link pinning a cluster of notes to the ‘body’ and ‘ending’ of a document structure, on the left of the screen.

5.2 Hierarchical document structure

The Structure View of the Writer’s Assistant presents the writer with a view of the hierarchical document structure and assists the writer in creating documents which conform to specified ‘text types’. When a notes network is linearised, the algorithm must determine positions for the linearised text elements in one of the many possible document structures which conform to the given text type. To assist this, elements of a text type are associated with tags which indicate the relative suitability of parts of the document structure for attaching linearised notes.



Turing suggested that machines would soon be able to compete with humans at intellectual reasoning.

Figure 9. The Notes Network view of the Writer's Assistant

5.3 Incremental linearisation

The Writer's Assistant is designed to support a writer in moving rapidly between engagement with a text and reflection on explicit ideas and intentions (Sharples, 1993). This means that the linearisation algorithm must be repeatedly applied to a notes network as it develops. The principle of least surprise suggests that the algorithm should exhibit 'inertia', retaining the structure of the notes network when it is re-visited, and minimising changes to the existing linear order when the network is re-linearised.

In the current version of the Writer's Assistant, this inertia is achieved through two mechanisms. Firstly, structure links created in previous sessions with the Notes Network view are retained, but with a lower priority than ones newly added by the writer. This has the effect of maintaining associations between clusters of notes and sections of the linear document *in the absence of* strongly contradictory structure links. The second method is to honour previously generated associations between individual notes and elements of document structure in the final stages of the linearisation. Whenever the algorithm has some latitude in determining which structural element is to represent a note in the linearisation of the document, it will choose to use structural elements already so related.

5.4 Bidirectional links

The Best First algorithm in Figure 4 considers links in the reverse direction after all the forward links have been followed. This is the method implemented in the Writer's Assistant, since it increases the robustness of the algorithm without significantly affecting the linearisation behaviour. Another approach is to assign a priority to each link type for its reverse direction (for example, the 'causes' link would be given a priority for its reverse direction corresponding to the 'is caused by' relation). This was the favoured method, but it was not implemented due to a lack of evidence to guide the choice of link weights.

6. Conclusions

We have described a robust general algorithm for notes network linearisation which has been implemented as part of a writing environment which combines an ideas organiser with a document editor. The experiments suggest that the best first linearisation algorithm is acceptable for creating a first draft of a linear text from a notes network but that further work is needed to make good use of the link information to guide linearisation and to evaluate the use of the algorithm as part of a writing environment.

References

Boden, M. (1990) *The Creative Mind: Myths and Mechanisms*. London: Weidenfeld and Nicolson.

Bolter, J. D., Joyce, M. and Smith, J. B. (1992) *Storyspace: Hypertext Writing Environment.* Eastgate Systems Inc., PO Box 1307, Cambridge, Ma, 02238.

Bush, V. (1945) 'As we may think.' *Atlantic Monthly*, 176(1), 163–165.

Buzan, T. (1989) *Use Your Head* (Revised Edition). London: BBC Books.

Hunt, K. W. (1965) *Grammatical Structures Written at three Grade Levels.* Research Report 3, NCTE, Champaign, Illinois.

Meyer, B. J. F., Young, C. J., and Bartlett, B. J. (1989) *Memory Improved: Reading and Momory Enhancement Across the Lifespan Through Strategic Text Structures*. Hillsdale, New Jersey: Lawrence Erlbaum.

Nelson, T. (1987) *Computer Lib / Dream Machines*. Redmond, Washington: Tempus Books of Microsoft Press.

O'Malley, C. E., and Sharples, M. (1986) 'Tools for management and support of multiple constraints in a Writer's Assistant.' In M.D. Harrison and A.F. Monk (eds.), *People and Computers: Designing for Usability,* 115-131. Cambridge University Press.

Sharples, M., Goodlet, J., Beck, E., Wood, C., Easterbrook, S., Plowman, L., and Evans, W. (1991) 'A Framework for the Study of Computer Supported Collaborative Writing', *CSRP 190*, School of Cognitive and Computing Sciences, University of Sussex.

Sharples, M., Goodlet, J., and Pemberton, L. (1992) 'Developing a Writer's Assistant.' In J. Hartley (ed.), *Technology and Writing: Readings in the Psychology of Written Communication,* 209–220. London: Jessica Kingsley.

Sharples, M. (1993) 'Computer support for the rhythms of writing', *CSRP 285.* School of Cognitive and Computing Sciences, University of Sussex.

Trigg, R., and Weiser, M. (1986) 'TextNet: a network based approach to text handling.' *ACM Transactions on Office Information Systems*, 4(1), 1–23.

Trigg, R. H., and Irish, P. M. (1987) 'Hypertext habitats: experiences of writers in NoteCards.' In *Proceedings of Hypertext '87*, 89–108, Chapel Hill, N.C.

Trigg, R. H., and Suchman, L. A. (1989) 'Collaborative writing in Notecards.' In R. McAleese (ed.), *Hypertext: theory into practice*, 45-61. Ablex.