HishiMochi: 非線形ズーミングを用いた動的検索システム

HishiMochi: A Dynamic Search System with Nonlinear Zooming

豊田正史　　増井俊之　　柴山悦哉*

Summary. We propose a dynamic search system, HishiMochi, with nonlinear zooming. It enables to easily find scattering data from a set of textual data that is constructed hierarchically such as Yahoo! and Java class libraries. HishiMochi shows search results clearly using nonlinearly magnified views of nested rectangles with animation. Transitions of these views are occurred dynamically during keyword inputs. HishiMochi provides simple and powerful user interfaces that can be used for various search strategies. It is also possible to edit layouts of data for making bookmarks or customized databases.

1　はじめに

我々が日常的にコンピュータ上で扱うテキストデータは、ある程度人手で分類されてはいるものの、データベースの様に形式的な検索手段がないことが多い。例として、ホームディレクトリ、Yahoo![1] などの WWW ディレクトリサービス、Java、C++ などのクラスライブラリ、が挙げられる。これらは通常、木構造を用いて分類されているため、構造について熟知していれば容易に枝を辿って目的のデータにたどり着くことができる。しかし、構造についてよく知らない、または忘れてしまった場合、目的のデータを検索するのは難しい。また WWW ディレクトリサービスでは、ある事柄に関係するデータが複数の場所に分散して置かれていることが多く、それら全てを探し出すのは非常に手間のかかる作業である。こういったデータの検索には、以下のような方法がよく用いられるが、それぞれ問題点がある。

1. **木構造の根から枝を辿っていく方法**　各枝に付けられた名前から適切だと思われる枝を選択し、根から順に検索範囲を狭めていく方法である。構造についてよく知らない場合、各節で適切な枝を選ぶのは難しいため、多くのバックトラック操作が必要となる。根に近い節で選択を誤れば誤るほどバックトラックには手間がかかるようになる。さらに、複数箇所に目的のデータが分散している場合、この方法で全てを探すのは非常に手間がかかる。
2. **全文検索を用いる方法**　木構造全体に対して全文検索を行い、結果のリストから目的のデータを探す方法である。しかし、単純なキーワードに対しては多くの結果が出力される場合が多く、その中から目的のデータのみを選り分けるの

* Masashi Toyoda, 東京工業大学大学院 情報理工学研究科, Toshiyuki Masui, ソニーコンピュータサイエンス研究所, Etsuya Shibayama, 東京工業大学大学院 情報理工学研究科

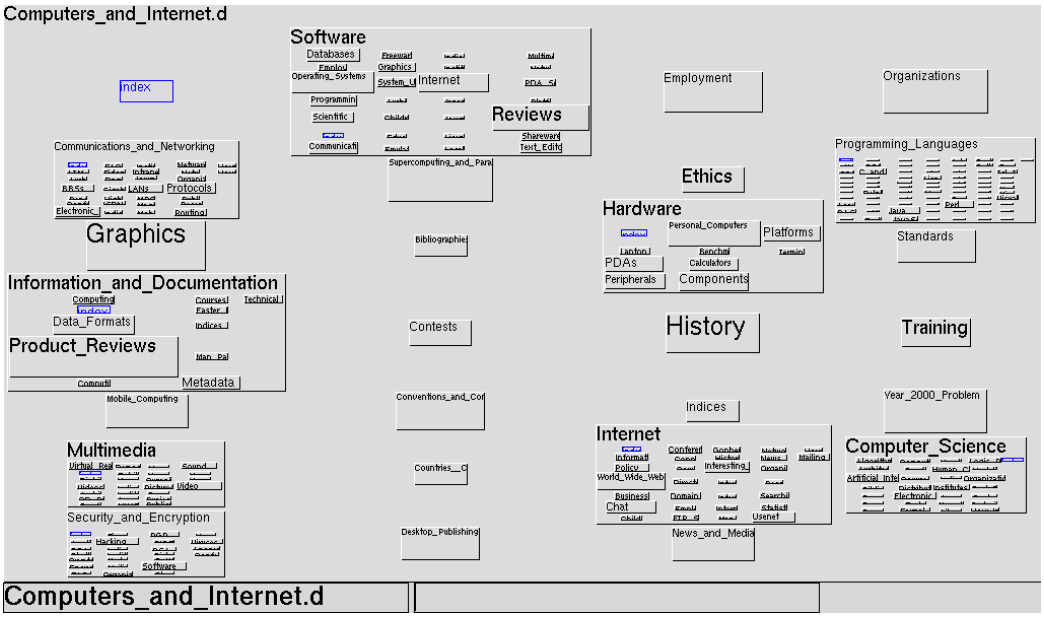


図 1. Yahoo! の Computers & Internet ディレクトリを HishiMochi で検索する際の初期状態

は非常に手間がかかる。また、人間が把握できる範囲まで候補を絞れる適切なキーワードの組み合わせを探すのは難しい。

3. **ある程度枝を辿り、それ以下を全文検索する方法** 根からある程度枝を辿って範囲を絞り、部分木に対して全文検索を行う。この方法でも、適切な枝を選ぶという問題は残されたままである。枝の選択を誤るとデータを探せない場合がある上、2 箇所以上に分散したデータを検索し損なう可能性がある。

これらの問題を解決するためには、複数の検索該当箇所を逃さないように全体に対して全文検索を行いながら、ユーザが適切に検索範囲の絞り込みを行えるようにする必要がある。具体的には、以下の検索方法を単純な操作で容易に行えるユーザインタフェースが必要である。

4. **全文検索の結果を基に複数の部分木に絞り込む方法** まず木構造全体に対して全文検索を行う。検索結果が多すぎて把握できない場合は、検索結果を含み、かつ適切だと思われる部分木を幾つか選択して検索範囲を絞る。選択した複数の部分木に対してさらに続けて全文検索を行い、結果を絞り込んでいく。

2 非線形ズームを用いた動的検索システム

我々は、第 1 節で述べた検索方法 4 を容易に実現するため、矩形の入れ子構造に対する複数フォーカスの非線形ズーム [6, 4, 2, 9] を用いた動的検索を行うシステム、HishiMochi[1] を提案する。図 1 に検索システムの画面ダンプを示す。これは Yahoo!

[1] **Hi**erarchy **s**earc**h** **i**nterface for **Mochi**

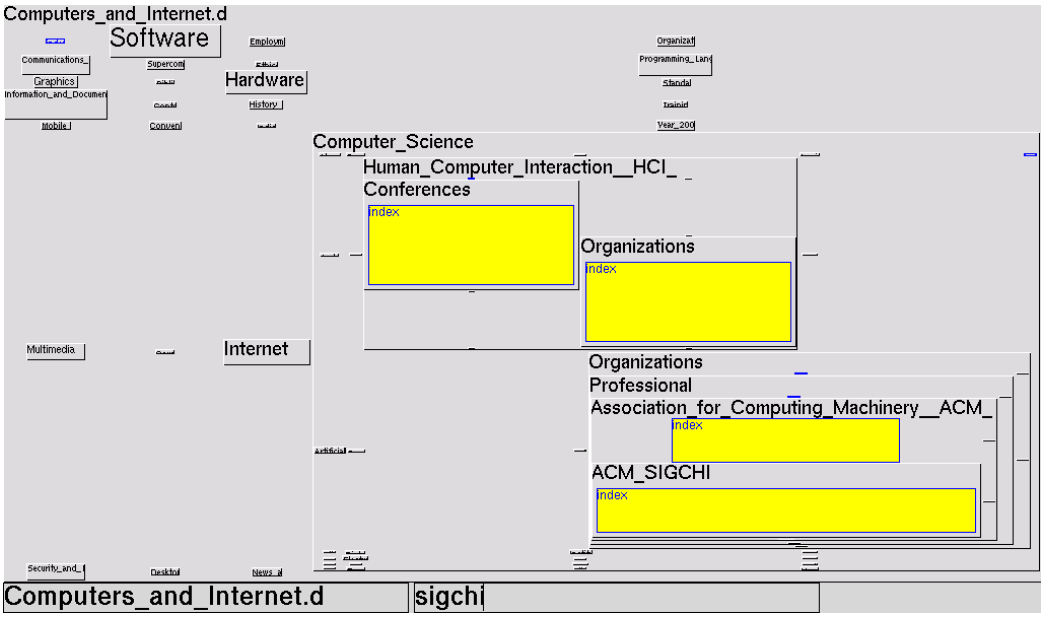


図 2. sigchi **というキーワードで検索を行ったときのレイアウト**

の Computers & Internet ディレクトリの構造を矩形の入れ子で表現したものである。各ディレクトリの大きさは、それ以下に含まれるファイルの数を反映している。検索は主にキーワード入力で行うようになっており、検索に該当する個所に対して拡大およびハイライト表示が行われる。例えば “sigchi”[2] というキーワードで検索すると、図 2 のようにレイアウトが変化する。以下に HishiMochi の特徴を列挙する。

非線形ズーム 非線形ズームは、検索に該当する複数の部分を拡大し、それ以外の部分を縮小した表示を全体の概要を保ちながら行うことができる。検索結果が全体的にどう配置されているか容易に把握できる上、絞り込みの際に複数の部分木の選択を容易に行える、という点で検索方法 4 を実現するのに適している。また、拡大・縮小の際になめらかなアニメーションを行うため、表示の変化が起こってもユーザが混乱することはない。

動的曖昧検索 キーワードを 1 文字入力する度に検索を行い結果の表示を変更する動的検索を行う。検索には、キーワードの打ち間違いをある程度許す曖昧検索を行う。キーワード入力の途中で検索範囲の絞り込みを行うなど、検索操作を自由な順序で行えるため、検索方法 4 のみではなく他の全ての検索方法を支援できる。また検索結果の表示の変化から入力中のキーワードが適切なものかどうかある程度判断することが可能である。

ダイレクトマニピュレーションを用いた絞り込み 画面上のディレクトリをマウスクリックで選択することで、そのディレクトリ以下に検索範囲を絞る事ができる。ディレクトリを複数選択することも可能である。

[2] 右下のテキストボックスに表示される

編集インタフェースへの統合 HishiMochi のインタフェースは Hyper Mochi Sheet[8] で提案した編集のインタフェースに統合されている。このためユーザが、レイアウトを後の検索の際に把握しやすい様に変更し、保存することができる。この機能を用いれば、個人用のデータベースまたはブックマーク構築ツールとして使用することも可能である。検索と編集のインタフェースの統合は何の不都合もなく自然に行うことができる。

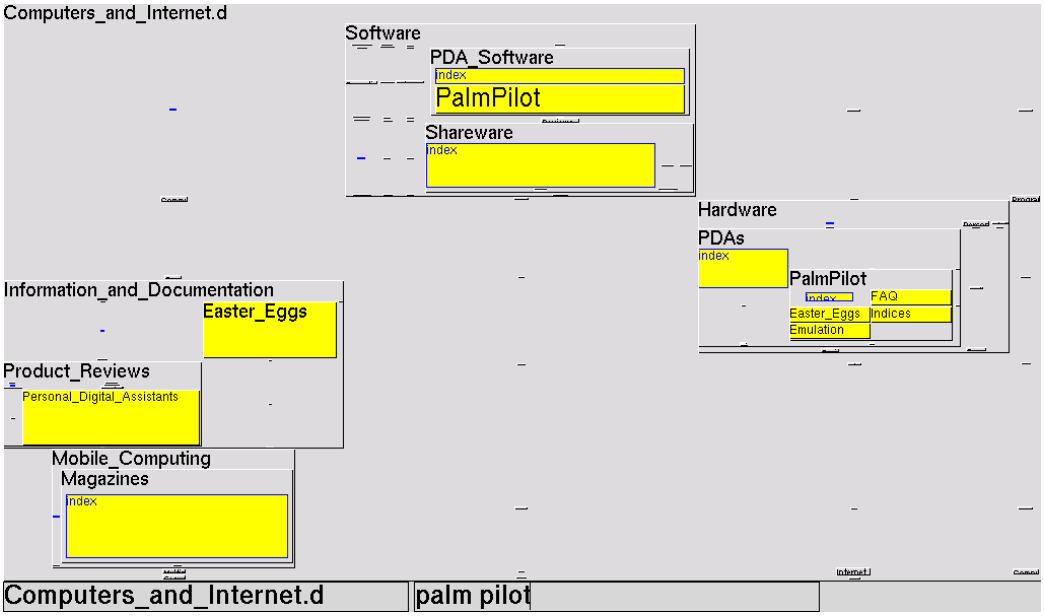
特に検索のインタフェースに関しては、キーワード入力、およびマウスクリックによる矩形の選択操作のみで 1 節で挙げた 1〜4 の検索方法をすべて実現できる点が優れている。矩形の明示的な拡大・縮小操作は検索には必要なく、インタフェースを単純にしたことで検索に多くの手間がかかることもない。

3 使用例

この節では例を用いて、HishiMochi が複数の場所に分散する目的のデータを適切に絞り込みを行いながら検索できることを示す。

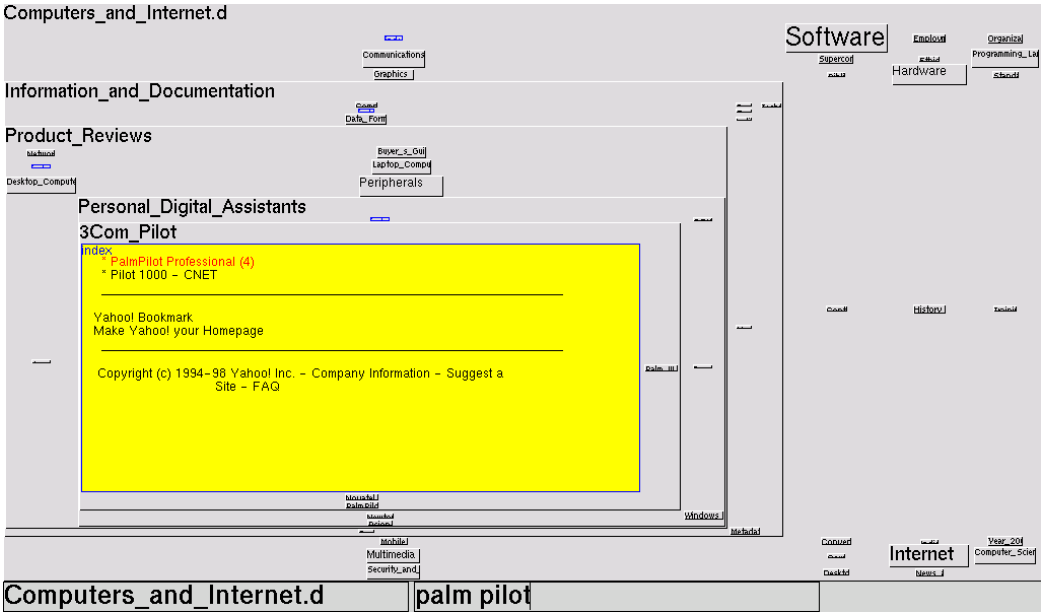
シナリオ PalmPilot[3] を購入したので、使いこなすための情報を集めたい。

キーワードとして "palm pilot" と入力すると、スペースをワイルドカードとして扱い、さらに前後にワイルドカードを付加したパターンで、ドキュメントの各行に対して検索を行う。正規表現で書くと ".*palm.*pilot.*" となる。以下の結果を見ると Software、Hardware ディレクトリなど複数の場所に該当するドキュメントがあることが分かる。結果表示の際にはズームのアニメーションが行われる。

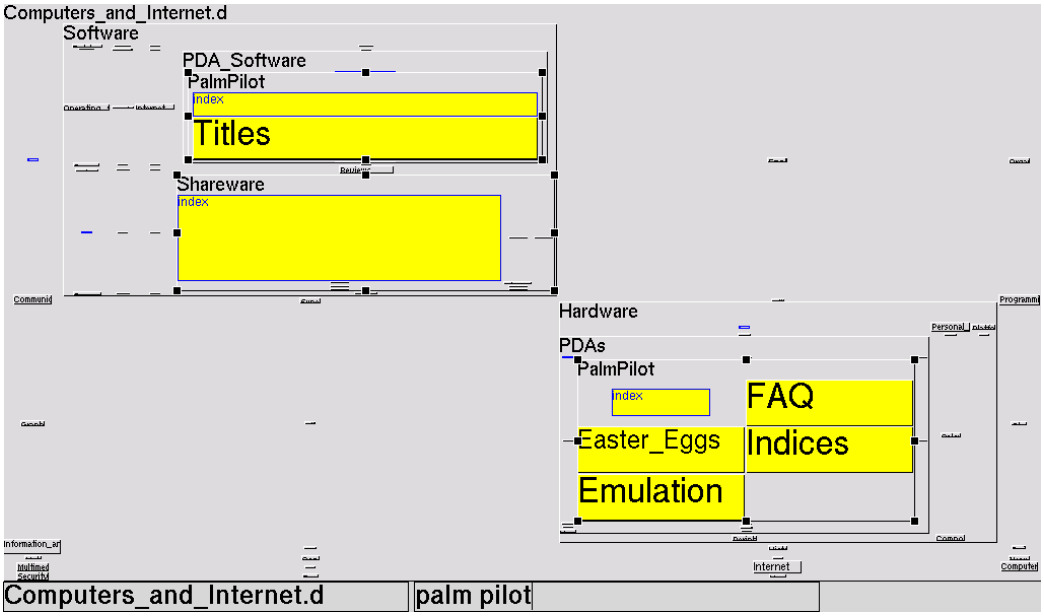


リターンキーを繰り返し押すことで、該当したドキュメントを一つ一つ見ていくことができる。ドキュメントのブラウザも、矩形の階層構造の中に埋め込まれており、検索に該当した行が赤く表示される。

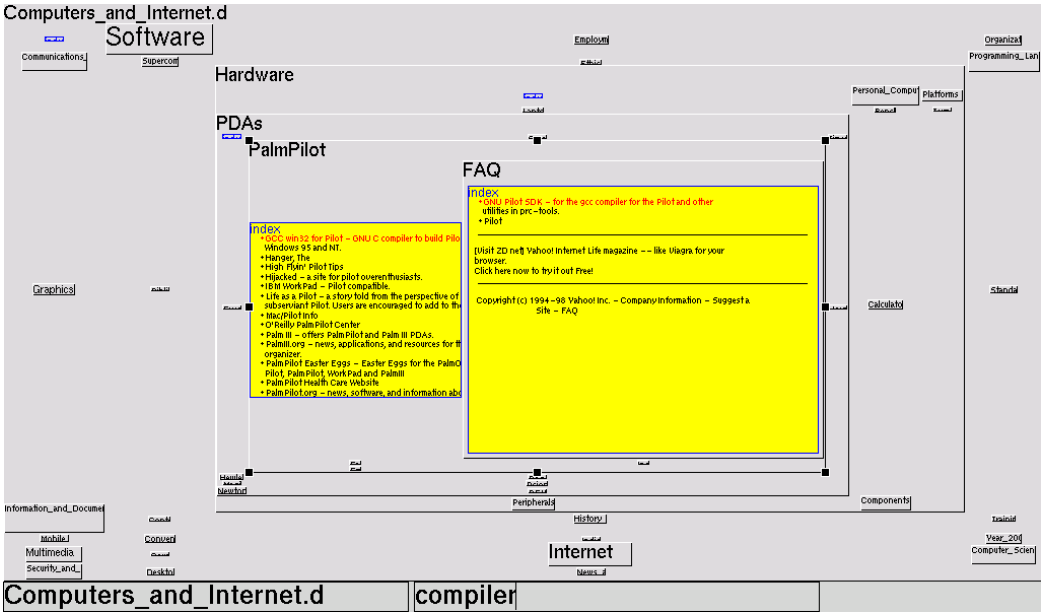
[3] 3Com 社の登録商標



一つ一つ見ていくには候補が多いので、ダイレクトマニピュレーションによる絞り込みを行う。ここでは、product review や magazines には興味がないのでそれ以外の部分を選択する。選択された矩形には 8 方向に黒いハンドルが付けられる。選択した後でスペースを入力すると同じキーワードで再検索を行い、以下のような表示となる。このように異なる階層の異なるレベルにあるディレクトリを同時に選択することが容易である。



選択したディレクトリは常に拡大表示されるので、これらのディレクトリに対して別のキーワードで検索を行うことができる。例えば PalmPilot 用のコンパイラを探すため、“palm pilot” を消して “compiler” と入力すると、2 つの候補が見つかる。



また、選択したディレクトリをドラッグアンドドロップして新たにディレクトリを作ってコピーしておけば、PalmPilot に関する個人用のブックマークが作成できたことになる。こうしておくと次回また検索する際に便利である。

4 検索方法のバリエーション

HishiMochi では、キーワード入力、矩形の選択を自由な順番で行うことができるため、第 1 節で述べた検索方法を全て行うことができる。検索方法 2〜4 に関しては、第 3 節で使用したインタフェースの組み合わせで容易に実現できる。ディレクトリを辿る検索方法 1 に関しては以下のように実現できる。

普通にディレクトリを辿るには、ディレクトリを選択した後、スペースを入力すればよい。また、各ドキュメントにパスを埋め込んでおくと、キーワード入力でディレクトリを辿ることができる。例えば、“/graphics/3D/Software” ディレクトリに到達するにはキーワードとして “/gra /3d /sof” と入力する。何段か先のディレクトリ名が見えている場合には途中のディレクトリを省略して入力しても良いので、ダイレクトマニピュレーションよりすばやくたどり着くことも可能である。

5 その他のアプリケーション

我々は Yahoo! の他にも、小説、ヘルプ、マニュアルなど、様々なデータに HishiMochi を応用した。例えば小説では、シャーロックホームズ全作品の英文テキストを書かれた年代毎にディレクトリに分類したものを扱った (図 3)。これを検索すると、「ある人物はどの年の作品に最初に現れて、以降どんな分布で登場しているか」、「ある事件について、それ以降の作品でどのくらい触れられているか」といった情報を一目で把握することができる。図 4 は、“moriarty” というキーワードで検索を行ったときのレイアウトである。モリアーティー教授が、 1893 年の “The Final Problem” に最初に現れ、以降 6 作品に登場しているのが一目で把握できる。

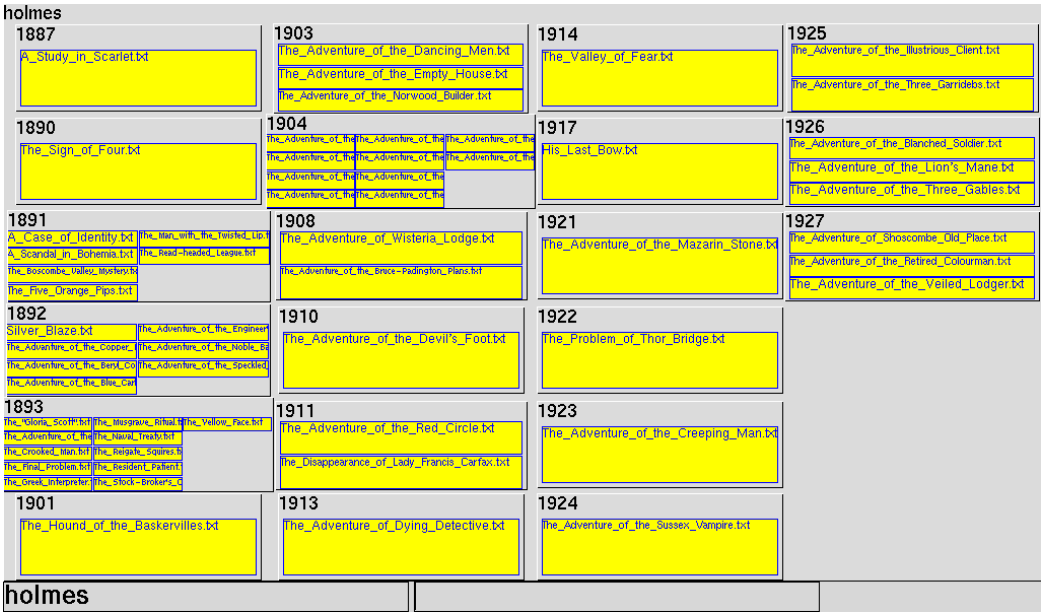


図 3. シャーロックホームズ全作品を書かれた年代毎に分類したもの

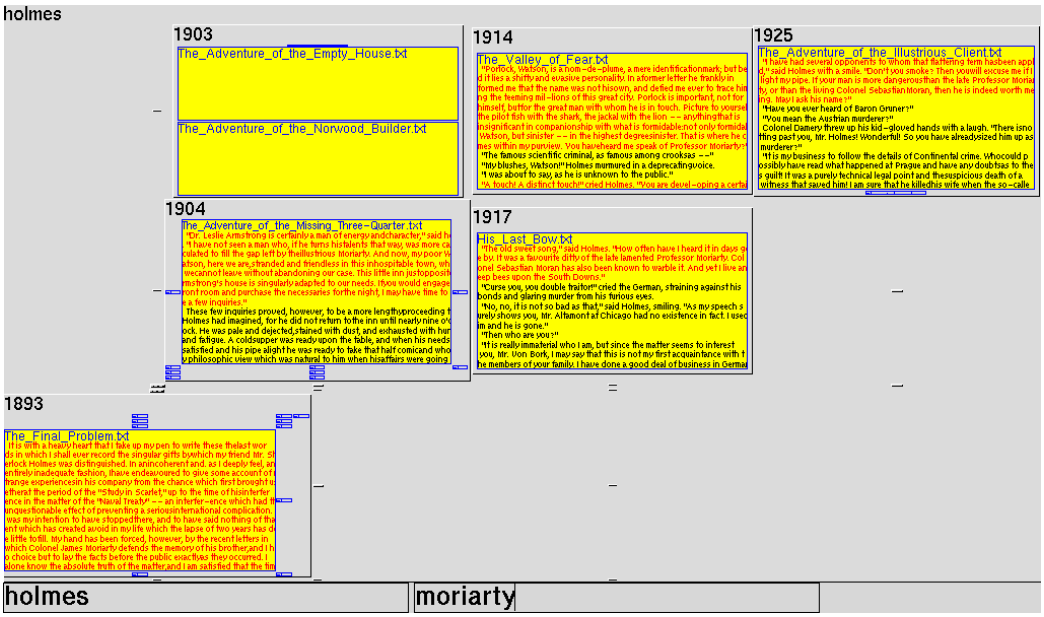


図 4. moriarty というキーワードで検索を行ったときのレイアウト

図 5. 写真ブラウザ

また、HishiMochi は写真のブラウザにも応用できる。デジタルカメラで撮った写真にキーワードを付加することで、写真のキーワード検索が行える。図 5 は、HishiMochi で、写真を含むディレクトリを表示したものである。各写真には日付、および写っている人、物などを表すキーワードを付加してある。アニメーション中にイメージのズームも行うようになっている。

6 実装

6.1 非線形ズーム

非線形ズームには Mochi Sheet のアルゴリズム [9] を用いた。このアルゴリズムは、ある階層内で幾つかの矩形が拡大された際に、全ての矩形がその階層に入りきらなくなった場合、全矩形を縦横の位置関係を保ちながら均等に縮小する。また、矩形の位置関係として、中央の座標を合わせて縦横に並べる制約が導入されている。

各矩形には、通常のサイズ (小サイズ) と検索に該当したときのサイズ (大サイズ) が定義されており、検索が行われるとその結果に従ってサイズの切り替えが行われる。検索に該当する矩形の祖先にあたる矩形はすべて大サイズに切り替えられる。すべてのテキストブラウザの矩形には、同じサイズが定義されているが、ディレクトリの矩形には、そのディレクトリに含まれるファイルの個数に応じたサイズが定義されている。従って、ファイルの多いディレクトリほど内容を詳細に見ることができる。

Yahoo! の例では、約 3000 のディレクトリ、および約 3000 のファイルが画面に配置されている。テキストデータの大きさは約 10M バイトである。現在の実装では Pentium II(233MHz) を搭載した PC 互換機において、全文検索には 1 秒程度かかり、アニメーションは秒間 5 コマ程度で行える。また、応答速度を高めるため、キー

ワードをすばやく入力すると中間の検索を中止するようになっている。

6.2 動的曖昧検索

HishiMochi は入力されたキーワードに対して動的曖昧検索を行う。曖昧検索のアルゴリズムは [7] で用いられているものを利用した。最初は正規表現に対して正確なマッチングを行い、それに該当するデータがない場合、1 文字の誤字・脱字を許すマッチングを行う。例えば、 “virtual” を “virtial” と打ち間違えてもマッチする。それでも見つからない場合、2 文字の誤字・脱字を許すというように、曖昧度を上げながら検索を行う。ある曖昧度で該当するデータが見つかればそのデータを結果として表示する。曖昧度の上限は 3 としている。

HishiMochi では、検索中に曖昧度が上がった場合、画面の背景が暗くなっていくフィードバックを提供している。1 文字入力する度に動的検索を行っているため、キーワードが適切なものかどうかの判断が入力途中でもできるようになっている。

6.3 編集インタフェースの統合

レイアウトの編集に必要な、矩形の移動およびリサイズ操作は容易に統合可能である。矩形の移動は、ドラッグアンドドロップで行える他、リサイズは矩形の選択に用いるハンドルで行うことができる。また、Hyper Mochi Sheet[8] では、矩形に大小 2 種類のサイズを定義し、これをすばやく切り替えるインタフェースを提案した。これは検索結果表示の際のサイズ切り替えに対応しており、そのまま統合可能である。

7 関連研究

HishiMochi は、Shneiderman らの提唱する dynamic query[5] の一種である。階層的データに対しては、Johnson が Treemap[3] を用いて dynamic query を行っている。

彼らが検索の対象としているのは、各データが定まった属性を持つデータベースである。このため、各属性に対して、値 (の範囲) を指定するための選択ボタン、メニュー、スライダーなどのウィジェットを用意する必要があり、インタフェースが複雑になる傾向にある。また、扱うデータベースが変われば、それに応じてインタフェースも変更しなくてはならない。HishiMochi では、主に、決められた属性を持たないテキストデータを対象としている。検索のインタフェースは Dynamic Query よりも単純であり、このインタフェースのみで、様々な種類のデータに対応できる。また、編集インタフェースを統合している点で、より利用範囲が広くなっている。

Treemap は、最初の階層では横に、次の階層では縦に、と画面を繰り返し分割していくことで階層構造を表示する手法である。Treemap を用いた dynamic query では、検索結果の表示にはハイライトのみが用いられている。ダイレクトマニピュレーションによって検索範囲を絞ることができるが、階層の深くにあるディレクトリは非常に小さくなっているので選択するのは難しい。また、この表示方式では、絞り込みによってレイアウトが大きく変わるため変化を把握するのは難しくなっている。HishiMochi では、非線形ズームを用いることで深いディレクトリも容易に選択することができる。さらにズーム中に矩形の位置関係を保つことからレイアウトが変わっても混乱せずに変化を把握することができる。

8 まとめ

木構造を持つテキストデータに対して、非線形ズームを用いた動的検索システム HishiMochi を提案した。このシステムでは、普通の検索方法で必要な、バックトラックや巨大なリストのチェックなどを行う必要はない。ユーザは、検索範囲を適切に絞りながら、複数箇所に分散するデータを容易に探し出すことができる。さらに、単純なインタフェースのみで、木構造に対する検索を様々な方法で行えることを示した。また、レイアウトの編集機能を用いることで、ユーザが個人用のデータベースを構築することも可能である。

参考文献

[1] Yahoo!. http://www.yahoo.com/.

[2] Lyn Bartram, Albert Ho, John Dill, and Frank Henigman. The Continuous Zoom: A Constrained Fisheye Technique for Viewing and Navigating Large Information Space. In *Proceedings of UIST '95*, pp. 207–215, November 1995.

[3] Brian Scott Johnson. *Treemaps: Visualizing Hierarchical and Categorical Data*. PhD thesis, University of Maryland, 1993.

[4] Emanuel G. Noik. Exploring Large Hyperdocuments: Fisheye Views of Nested Networks. In *ACM Conference on Hypertext and Hypermedia*, pp. 14–18, 1993.

[5] Ben Shneiderman. Dynamic Queries for Visual Information Seeking. *IEEE Software*, pp. 70–77, November 1994.

[6] 三末和男, 杉山公造. 図的思考支援を目的とした図の多視点遠近画法について. 情報処理学会論文誌, Vol. 32, No. 8, pp. 997–1005, 1991.

[7] 増井俊之, 水口充, George Borden, 柏木宏一. なめらかなユーザインタフェース. 第 37 回冬のプログラミングシンポジウム予稿集, pp. 13–23, 1996.

[8] 豊田正史, 高橋伸, 柴山悦哉. Hyper Mochi Sheet: 複数フォーカスズーミングエディタにおけるナビゲーションのためのフォーカス予測手法. インタラクティブシステムとソフトウェア V WISS'97, レクチャーノート/ソフトウェア学 18, pp. 87–94. 近代科学社, December 1997.

[9] 豊田正史, 高橋伸, 柴山悦哉. Mochi Sheet: 大規模なビジュアルプログラムの効率的編集を支援するズーミングインタフェース. 情報処理学会論文誌, Vol. 39, No. 5, pp. 1395–1402, 5 1998.