Retrieval im Desktop

Wolfgang Müller
LS Angewandte Informatik I
Universität Bayreuth
D-95440 Bayreuth
Germany *†

May 14, 2002

Retrieval im Desktop nennen wir die Bereitstellung von Werkzeugen zur Indexierung und Auffindung von Dokumenten im Desktop, wohlgemerkt von allen Dokumenten, d.h. von öffentlich zugänglicher Dokumentation bis zu privaten Mails. Ziel des Vortrags ist es einerseits, Techniken und aktuelle Forschung vorzustellen, andererseits Szenarien zu entwerfen, wo Retrieval jeden Nutzer weiterbringen kann.

Suche nach Texten in großen Beständen hat sich spätestens mit Google und Konsorten demokratisiert. Freie Softwarepakete zur Indexierung von Webseiten existieren. Proprietäre Systeme wie MacOS und Windows machen nun Retrieval-Werkzeuge auch auf dem Desktop zugänglich, jedoch fristet Information Retrieval auf dem Free Desktop noch ein Schattendasein.

Im Moment sind dem Autor zwei Projekte bekannt, die Information Retrieval für Benutzerdokumente dem Free Desktop näher bringen möchten. Erstens, Medusa, das eine Text Retrieval Engine in den File Manager Nautilus integriert, und zweitens Fer de Lance (FdL, [1]).

Ziel von FdL ist die Suche nach jeglicher Art von Multimedia-Dokumenten (also von Texten, aber auch von Bildern etc.) vom Desktop aus zu ermöglichen. Die notwendige Indexierungsarbeit soll im Hintergrund ablaufen, Retrieval Services sollen so einfach zugänglich werden, wie ein Directory Listing.

Im Vortrag wird zunächst auf existierende Softwarepakete eingegangen, die Text-Retrieval-Funktionalität bereitstellen, u.A. Bradley Rhodes' Remembrance Agent. Danach werden Einblicke in inhaltsbasierte Image-Retrieval, wie sie das GNU Image-Finding Tool (GIFT) bereitstellt, gegeben. Hier wird insbesondere auch darauf eingegangen, wie GIFT zur Zeit Multimedia-fähig gemacht wird: durch eine Query Engine, die Ähnlichkeitssuche auf attribuierten Graphen durchführt.

Dann werden existierende Benutzerschnittstellen vorgestellt, von Carsten Pfeiffers kmrml bis zu Forschungs-Ansätzen zur Visualisierung großer Datenmengen (z.B. Chaomei Chen `http://www.pages.drexel.edu/~cc345/`).

Danach werden Nutzungszenarien vorgeschlagen. Hier wird insbesondere darauf eingegangen, inwieweit die benötigten Komponenten schon bereitstehen, und nur noch geeignet verbunden werden müssen, und inwieweit es sich hier um aktuelle Forschung handelt.

Mit einer "tentative Roadmap", wird der Vortrag abgeschlossen.

1 Einführung

Computer sollen uns das Leben erleichtern. Nebenbei erleichtern sie uns auch viele Probleme, die wir ohne Computer nicht hätten, dies ist jedoch ein anderes Thema, eher geeignet für eine Philosophy-Newsgroup.

*Die als Teil dieses Vortrags vorgestellten Arbeiten am GNU Image Finding Tool wurden an Professor Puns Vision & Multimedia Lab bei an der Universität Genf durchgeführt

†Dank geht an Carsten Pfeiffer für's die Arbeit an FdL und für's Durchlesen dieses Artikels, sowie meine Kollegen und Professoren in Genf und Bayreuth für interessante Diskussionen.

In der Büroarbeit helfen uns Computer im Moment dabei, Dokumente zu erzeugen und sie zu verwalten. Für die Verwaltung von Dokumenten muss der Benutzer[1] jedoch meist einen Aufwand leisten: er muss wie im richtigen Leben seine Dateien in Ordnern ablegen, und diese eventuell in weiteren Ordnern ablegen (das wären im richtigen Leben eher Schränke), und ihnen, den Dateien wie den Ordnern, einen Namen mit Wiedererkennungswert geben.

Während bei moderaten Datenmengen hierbei keine Probleme entstehen, kann bei großen Mengen von Dateien (z.B. 5 Jahren E-Mail im Geschäftsbetrieb) schon eine derart tiefe Verschachtelung von Ordnern und Verzeichnissen erreicht werden, dass die viel beschworene Desktop-Metapher nicht nur die Arbeit veranschaulicht, sondern auch zu einer genauen Nachbildung von Ordnungs- und Klassifizierungsproblemen auf dem Schreibtisch führt. "Virtuelle Realität" auf unerwünschte Art, sozusagen.

Ähnlich verhält es sich mit Datenbanken: die Daten können nicht so einfach in einen Topf geworfen werden, den man umrührt und aus dem dann auf magische Weise die gewünschten Daten herauszieht: durch geeignete Datenbank-Schemata muss die Sortierung und Suche vorbereitet werden. Der hier zu leistende Intellektuelle Aufwand ist enorm und ist zumindest ein Teil des Arbeitsinhalts einer großen Zahl von Entwicklern und Administratoren.

Einen für den Benutzer befriedigend einfachen Lösungsansatz bieten Retrieval-Systeme. Sie indizieren Dokumente so, dass man Ähnlichkeitssuchen auf ihnen durchführen kann: Der Benutzer gibt Worte an, die in dem gewünschten Dokument vorkommen sollen, und erhält eine nach Ähnlichkeit[2] zur Anfrage geordnete Liste als Resultat. Das Forschungsgebiet Information Retrieval existiert schon seit ungefähr 40 Jahren. Auch für Normalnutzer ist Retrieval seit längerem bekannt. Vor allem die Web-Suche weist auf die enormen Möglichkeiten hin, aber auch die Suche nach Hilfe-Texten ist schon seit Längerem über Volltext-Indizes, also mit Information-Retrieval-Methoden möglich. Seit zwei Jahren machen sich auch Bestrebungen bemerkbar, Retrieval-Techniken auch dem Desktop zugänglich zu machen, hierbei ist insbesondere Apples Sherlock zu nennen. Das Projekt Medusa, das ein Teil von des Nautilus-Projektes ist, geht in dieselbe Richtung.

Aber was man mit Retrieval im Desktop machen kann, ist hiermit noch in keinster Weise ausgereizt. In diesem Artikel werden einige Möglichkeiten von Retrieval im Desktop aufgezeigt. Hierbei wird einerseits auf die aktuelle Forschung verwiesen, die viel Nützliches schon angedacht, und auch in Prototypen umgesetzt hat. Ferner wird aber auch realistisch dargelegt, wie man durch Kombination von bereits existierenden Programmen mit neuen Programmteilen den Nutzwert von Retrieval für "Otto Normalbenutzer" verbessern kann.

Ziel dieses Artikels ist hierbei auch, auf die Arbeitsteilung zwischen Desktop und Server, zwischen Free Software Hacker und Wissenschaftler einzugehen und ein System vorzuschlagen, von dem beide Seiten in idealer Weise profitieren.

Dieser Artikel ist wie folgt organisiert: Im Folgenden stellen wir zunächst kurz Medusa vor, das bisher einzige Projekt, das zumindest über kurze Zeit auf dem Desktop Fuß fassen konnte. Wir gehen hierbei auch auf die Probleme ein, die dazu geführt haben, dass Medusa in der jetzigen Vorversion von GNOME 2 nicht mehr enthalten ist.

Der Rest des Artikels ist verschiedenen Aspekten des Fer-de-Lance (im Folgenden FdL genannt) Projekts und seinen Vorbildern gewidmet. FdL ist in mehrerer Hinsicht breiter angelegt als Medusa: FdL soll nicht nur die Suche über Text, sondern auch über Multimedia-Beispiele ermöglichen (gib mir ähnliche Bilder zu diesem Bild). Ferner versucht FdL so weit wie möglich nach allen Seiten offen zu sein. Möglichst viel von FdL soll plattformunabhängig sein. Wie sich im Laufe dieses Artikels zeigt, ist FdL ein weites Feld, jedoch *keine* Science Fiction.

Zunächst sei jedoch erklärt, wie Retrieval funktioniert, da es einigen der im Folgenden vorgestellten Programmen zugrunde liegt. Ein Verständnis der Theorie ist instruktiv, wenn auch zum Verständnis dieses Artikels nicht unbedingt notwendig.

[1] die vielen Benutzerinnen mögen dem Autor die ausschließliche Verwendung der männlichen Form verzeihen sie ist innerhalb dieses Artikels als geschlechtsneutral zu verstehen.

[2] Ähnlichkeit ist hierbei ein errechnetes Maß, dass nicht zwingend mit der vom Benutzer empfundenen Ähnlichkeit übereinstimmen muss

2 Das Funktionsprinzip vieler Retrieval-Systeme

In inhaltsbasierter Retrieval versucht das Retrieval-System Dokumente zurückzuliefern, die eine möglichst hohe Übereinstimmung mit einer Query haben. Diese Query ist typischerweise eine Folge von Worten, beispielsweise "Retrieval Desktop".

Die meisten bekannten Methoden werden nun für jedes in der Dokumentsammlung (der Collection) vorhandenen Dokumente einen "Score" berechnen. Hierzu werden für jedes Dokument die Worte bewertet, die Query und Dokument gemeinsam haben. Jedes Wort bringt dem Dokument Punkte. Ein Klassisches Bewertungsverfahren ist das *tf.idf* Verfahren.

Beim *tf.idf*-Verfahren wird von zwei Grundideen ausgegangen:

- Ein Wort (Query *Term*) das in der Query häufig vorkommt (also ein Wort mit hoher `Term Frequency`) ist für die Query wichtig.
- Ein Wort das nur in wenigen Dokumenten vorkommt (ein Wort mit geringer *Document Frequency*) unterscheidet diese Dokumente stark von anderen Dokumenten.

Das Gewicht jedes Query Terms ist also durch den Quotienten $\frac{term\ frequency}{\log(document\ frequency)}$ gegeben. Dies wird abgekürzt als "*tf* mal inverse *df*" oder *tf.idf*. Der Zeitgewinn bei der Verwendung eines Indexes stammt daher, daß nur diejenigen Worte betrachtet werden müssen, die in Query und Dokument vorkommen. Man kann jetzt also für jeden möglichen Query Term eine Liste derjenigen Dokumente vorhalten, die diesen Query Term enthalten. Um korrekte Punktzahlen an die Dokumente zu vergeben, muß also nur für jeden Query Term eine Liste von Dokumenten eingelesen werden.

Dieses Verfahren trägt schon sehr weit. Typischerweise wird es zusammen mit vielen Techniken eingesetzt, die auf Erfahrungswerten basieren. Beispielsweise besagt die Erfahrung, daß sehr häufig vorkommende Worte die Retrievalresultate eher beeinträchtigen. Die meisten Systeme streichen daher sogenannte "Stop Words" aus den Texten, die sie indexieren.

Wichtig in unserem Zusammenhang ist, daß Retrieval-Systeme typischerweise nicht versuchen, den Text zu verstehen, und daß die eigentliche Indexstruktur nicht Worte, sondern nur noch Term-Identifikatoren sieht.

3 Text Retrieval statt `find|grep`: Sherlock und Medusa

Apples Sherlock war einer der ersten Ansätze Retrieval-Methoden auch für ganz normale Dateien einem breiten Publikum zugänglich zu machen. Interessant hierbei: es ist möglich, sowohl im Internet, als auch auf eigenen Dateien nach Texten zu suchen.

Sherlock stand Pate zu Medusa, dem Text-Retrieval-Zusatz zu dem GNOME File Manager Nautilus. Medusa indexiert zu fest gegebenen Zeitpunkten freigegebene Directories und macht sie dann einer Textsuche und einer Suche nach Daten über die Datei (Metadaten), wie z.B. der Dateigröße zugänglich. Zusätzlich ist es in GNOME möglich "Icons" (wie z.B. Smileys) Dateien zuzuordnen. Auch diese Metadaten werden mittels Medusa suchbar.

Medusa ist ein ehrgeiziges Projekt. Leider wurde der Entwicklergemeinde offensichtlich der Nutzwert des Projektes nicht klar. Während Schwächen in der Effizienz wortreich bemängelt wurden, Sicherheitslücken gefunden und viele davon auch geschlossen wurden, bekam Rebecca Schulman offensichtlich wenig Hilfe bei der Implementation. Dies ist zumindest der Eindruck, den man beim Lesen der Medusa-Sourcen gewinnt. Klarer wird dies noch, wenn man in der Nautilus-Entwicklerliste nach dem Stand von Medusa bezüglich Sicherheit und Effizienz fragt. Medusa ist dort offensichtlich seit Langem kein Diskussionsthema mehr, Information ist hier nur schwer erhältlich.

Dies ist nicht allzu erstaunlich: Die Desktop Entwicklergemeinde besteht aus Leuten, die mit den bereits vorhandenen Werkzeugen hervorragend umgehen können. Häufig haben sie im Laufe der Jahre schon ihr

eigenes System zum Ordnung halten gefunden. Der Programmierer hat also andere Wünsche an die Fensteroberfläche als der Normalbenutzer. Schwerer wiegt es jedoch, dass man viel programmieren kann, ohne jemals ernsthaft mit der Funktionsweise von Retrieval konfrontiert worden zu sein. Ernsthafte Programmierer, die nie ein Programm mit Fenstertechnik Programmiert haben werden jedoch eher selten sein. Kurz, die potentielle Entwicklergemeinde einer Retrieval Engine rekrutiert sich aus einem kleineren Personenkreis als ein Desktop-Projekt wie KDE oder GNOME. Ein Desktop-Retrieval Projekt muss sich diesen kleinen Kreis potentieller Entwickler dann noch mit erfolgreichen Web-Suchmaschinen wie `ht://dig` teilen.

Auf die Sicherheitsprobleme eines Desktop Retrieval Systems wird weiter unten eingegangen.

4 Über Kombination von Softwarepaketen zum Erfolg

Medusas Problem scheint also eigentlich ein typisches Problem der Free Software Welt: Programmierer sind Freiwillige. Was niemand interessant findet, wird niemand programmieren. Was interessant ist, bestimmen häufig externe Faktoren, also die Mode des Augenblicks. wenn sich auch die Mode der Programmierer von der Mode der Normalbenutzer unterscheiden mag.

Dies ist jedoch keineswegs negativ, wenn man sich auf die Stärken von Free Software besinnt, und diese versucht, für sich zu nutzen. Diese Strategie versuchen wir im FdL-Projekt zu verfolgen, wie im Folgenden verdeutlicht wird.

Eine der vielen Stärken der Free Software Bewegung ist, daß sie neben Projekten professioneller und engagierter Freizeit-Entwickler (viele sind beides) auch aus universitären Projekten gespeist wird. Die Interessenlage eines KDE- oder GNOME-Entwicklers wird häufig von der eines Forschers verschieden sein. Während der Forscher, häufig mit einem Proof Of Concept, also dem Beweis der Gültigkeit des Konzepts zufrieden ist, strebt ein Desktop-Entwickler die leichte Bedienbarkeit, die flexible Programmierung und ein fertiges Produkt an.

Das heißt, Forscher und Entwickler befinden sich in einer Situation, von der beide nur profitieren können, wenn sie gut zusammenarbeiten. Der Forscher stellt gültige Konzepte bereit, zusammen mit einem Proof of Concept oder einer Implementation der wichtigsten Bestandteile, der Entwickler ist in der Lage, sie einem breiten, technisch interessierten Publikum zugänglich zu machen. Idealerweise ermöglicht es seine tiefe Kenntnis des Inneren der Benutzeroberfläche und der von ihr bereitgestellten Dienste, schnell viel Nutzwert aus der Benutzeroberfläche zu ziehen.

In diesem Sinne bemühen wir uns im FdL-Projekt, den Kontakt zwischen Forschern und Desktop-Entwicklern herzustellen. Im Moment haben wir einen kleinen Kern (Carsten Pfeiffer und den Autor dieses Textes). Die kritische Masse ist noch nicht erreicht. In diesem Artikel wird jedoch verdeutlicht, wie relativ einfach wichtige Ziele des FdL-Projektes erreicht werden können, und wo interessierte helfen können.

Zunächst wird jedoch in einigen Beispielen dargestellt, welche Forschungssysteme FdL beeinflussen, an welchen Systemen das Projekt sich orientiert.

5 Einige Systeme, die für FdL interessant sind

5.1 Remembrance Agent

Der Remembrance Agent (RA) ist als Doktorarbeit von Bradley Rhodes an der Software Agents Group (Pattie Maes) am Media Lab des MIT entstanden, und jetzt unter GPL erhältlich. Als Besonderheit sei hier vermerkt, daß bei Nichtbenutzung des RA-codes und Bereitstellung einer dem RA ähnlichen Funktionalität wegen der auf dem RA aufbauenden Patente Kontakt mit dem MIT aufzunehmen ist.

Der RA ist ein “JITIR”, ein Just-In-Time Information Retrieval System. Er beobachtet die Eingaben seines Benutzers, und sucht ständig in seiner Dokumentensammlung nach Dokumenten, die zu dem Text passen, den der Benutzer gerade bearbeitet. Eine Kurzzusammenfassung der gefundenen Texte wird ständig angezeigt,

jedoch *ohne* den Eingabefokus an sich zu reißen. Der Benutzer hat dann also die Wahl, ob er auf die Vorschläge des RA eingehen will oder nicht. Dies ist beeindruckend, und führt laut [7] zu Resultaten, die teils die Kreativität fördern, weil sie gut sind, teils aber auch, weil sie gerade *nicht* passen und einen auf andere Gedanken kommen lassen.

Leider ist jedoch der RA in Emacs-Lisp programmiert, d.h. nur auf dem GNU-Emacs lauffähig. Eine Integration des RA in GNOME oder KDE könnte hier relativ direkt die Verbreitung sehr vergrößern. Denkbar wäre, konfigurierbar zu machen, welche Anwendungen derartige JITIR-Funktionalität bereitstellen, der Benutzer würde in den gewünschten Anwendungen immer über ähnliche Dokumente informiert.

5.2 Watson

Auf der Suche nach sogenannten Information Management Agents, die dem Benutzer bei seiner Arbeit helfen, und ohne ihn zu stören mit Zusatzinformationen versorgen, ist das Watson-Projekt [3] der Northwestern University einige Schritte weiter.

In Watsons Architektur sorgen *Applikationsadapter* dafür, daß Watson selbst eine einheitliche Sicht auf die Dokumente verschiedener Applikationen erhält. *Informationsadapter* erzeugen aus dieser Repräsentation Queries für verschiedene Datenquellen, die dann mittels dieser Queries abgefragt werden.

Die entsprechenden Resultate werden dann dem Benutzer zugänglich gemacht. Hier wird wiederum darauf Wert gelegt, daß die Resultate so präsentiert werden, daß der Benutzer sie zwar zur Kenntnis nehmen kann, dies aber nicht unbedingt muß. Watson kann beispielsweise zusammen mit Microsoft Word und MS Internet Explorer verwendet werden. Eine "public version" ist zur Zeit in Arbeit.

5.3 GNU Image Finding Tool

Der Remembrance Agent und Watson verwenden Textanfragen, um die benötigten Dokumente zu finden. Das heißt, auch Bilder werden eher über den sie umgebenden Text oder ihren Titel, als über ihre visuellen Eigenschaften gesucht. Das GNU Image Finding Tool hingegen verwendet rein die visuellen Eigenschaften eines Bildes, um ähnliche Bilder zu finden. Eine Beispielanfrage ist in Fig. 1 zu sehen.

Das GNU Image Finding Tool (GIFT) entstand an dem Vison und Multimedia-Labor der Universität Genf als Teil des *Viper* Projekts[10]. Ziel dieses Projektes ist, die Ähnlichkeitssuche auf Multimedia-Daten, wie z.B. Fotos und Videos zu ermöglichen, und zwar mit Techniken, die zwar nur im Desktop-Maßstab demonstriert werden, jedoch prinzipiell auch auf große Datensammlungen anwendbar sind. Dies ist ein Teil des Arbeitsgebiets der Content Based Image Retrieval (CBIR), das vor ungefähr 20 Jahren begründet wurde. Ein wissenschaftlicher Überblick über das Gebiet wird in [9] gegeben; einem breiteren Publikum zugängliche Ausführungen finden sich beispielsweise in [5].

Viper, die Image Retrieval Engine des GIFT, ist sehr stark von Text-Retrieval-Techniken inspiriert. Die in [10] propagierte Grundidee ist, jedes Bild in eine Folge von Term-Identifikatoren umzusetzen. Diese Term-Identifikatoren können dann genau so wie ein Text behandelt werden. Unter anderem benutzt GIFT das oben beschriebene *tf.idf*–Gewichtungsschema.

Während GIFT im Moment nur für (unbewegte) Bilder eingesetzt werden kann, ist seine Query Engine *Viper* darauf ausgelegt, beliebige Charakteristika von Dokumenten zur Retrieval heranzuziehen. Schwächen beim Updating führen jedoch dazu, daß zur Zeit eine neue Query engine entwickelt wird, die leichteres Hinzufügen von Dokumenten erlaubt.

Der GIFT Server ist konzipiert als ein Forschungsträger: um experimentieren zu können müssen Veränderungen einfach durchführbar sein. Gleichzeitig soll das Framework nicht zuviel Verwaltungsaufwand verursachen. Heraus kam eine Plugin-Architektur, die sich im Forschungsalltag bewährt hat. Im Abschnitt 6 wird hierauf ausführlicher eingegangen.

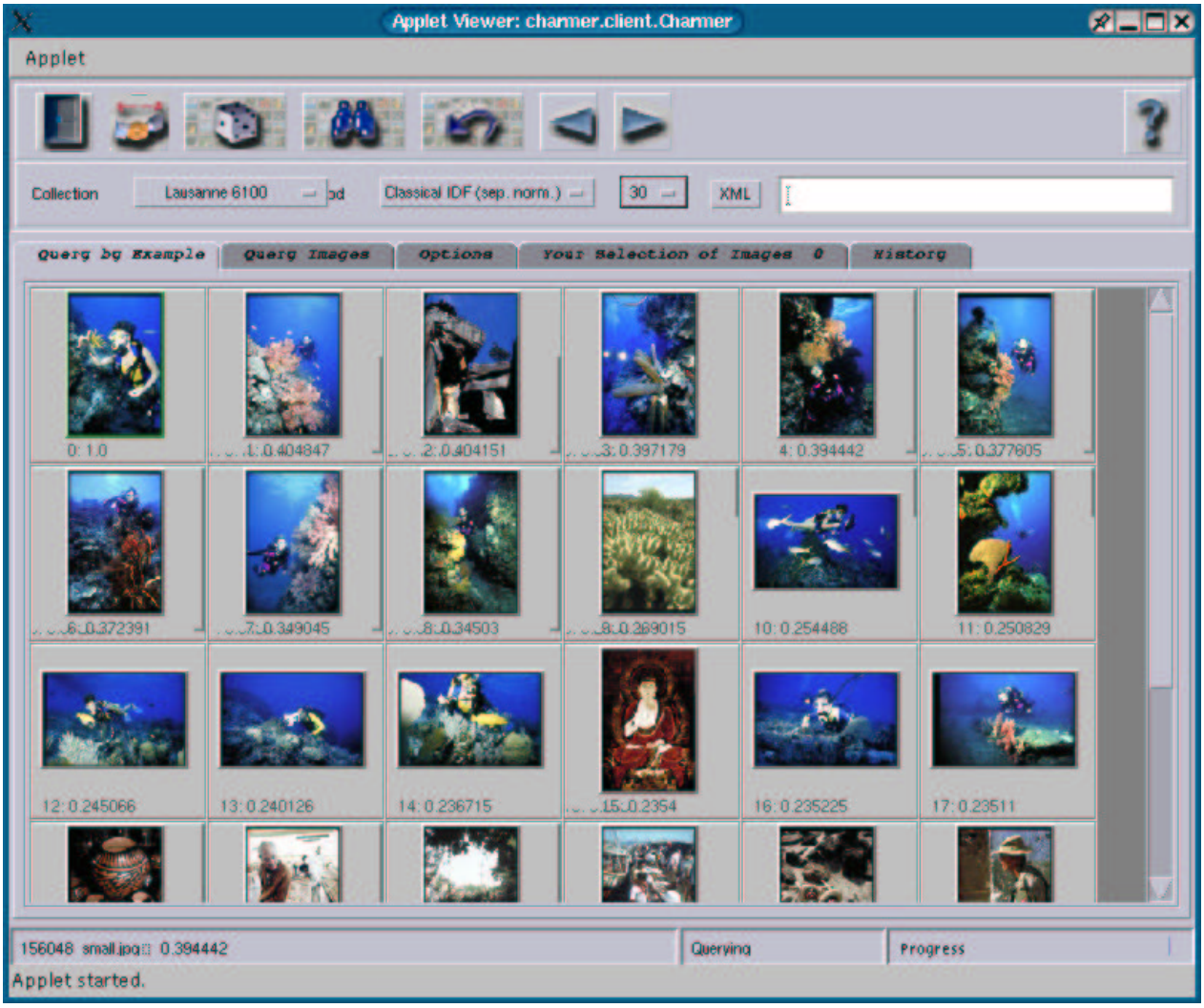

Figure 1: Dieses Bild zeigt SnakeCharmer, ein Interface für das GIFT, das von Zoran Pečenović an der EPF Lausanne entwickelt wurde. Das Bild oben Links ist die Anfrage, die anderen Bilder sind die ersten Resultate der Anfrage. Diese Demonstration ist unter [2] zugänglich.

5.4 Visualisation mit Pathfinder Networks

Den vorgenannten Systemen werden die Daten immer aktiv abgefragt. Entweder erkennt der Benutzer einen Informationsbedarf und führt eine Query per Hand aus, oder das System erzeugt eine Anfrage aus dem aktuellen Verhalten des Benutzers. In allen Fällen wird ein Pseudo-Dokument erzeugt, das die Eigenschaften hat, die von den Anfrageresultaten erwartet werden.

Pathfinder Networks [8] sind ein spektakulärer Vertreter einer Forschungsrichtung, die sich damit beschäftigt, wie man Informationen so aufbereiten und visualisieren kann, daß der Benutzer einen *Überblick* über die Daten erhält und diese Daten "entdecken" kann. Dies ist insbesondere wichtig, wenn man nicht genau weiß, wonach man sucht, jedoch wissen möchte "ob da etwas ist" was interessant ist.

Pathfinder Networks setzen nur voraus, daß zwischen je zwei Elementen eines Datenbestandes (z.B. zwischen je zwei Text-Dokumenten einer Artikelsammlung oder zwei Bildern einer Bildsammlung) eine "Ähnlichkeit" berechnet werden kann. Würde man einen derartigen Graphen visualieren, so würde dies zu einem unübersichtlichen Wirrwarr führen, der alle Knoten miteinander mehr oder weniger stark verbindet. Pathfinder Networks werden nun aus einem derartigen vollständigen Graphen durch Streichung von Kanten erzeugt. Und zwar wird ein Graph gefunden, der nur diejenigen Kanten enthält, die eine generalisierte Dreiecksungleichung nicht verletzen. Die übliche Dreiecksungleichung besagt, daß für drei Knoten A, B, C gilt

$|A-B|+|B-C| \geq |A-C|$. Die Generalisierung bezieht sich darauf, daß auch Pfade über mehrere Knoten eine entsprechende Ungleichung nicht verletzen dürfen. Was übrig bleibt, sind genau die "kurzen Wege" innerhalb des Graphen. Nach Experimenten von Chen *et al.* in denen Pathfinder Networks mit Text Retrieval Methoden zusammen eingesetzt wurden, können sie, bei geeigneted Distanzmaßen die inhaltliche Struktur der dargestellten Kollektion wiedergeben.

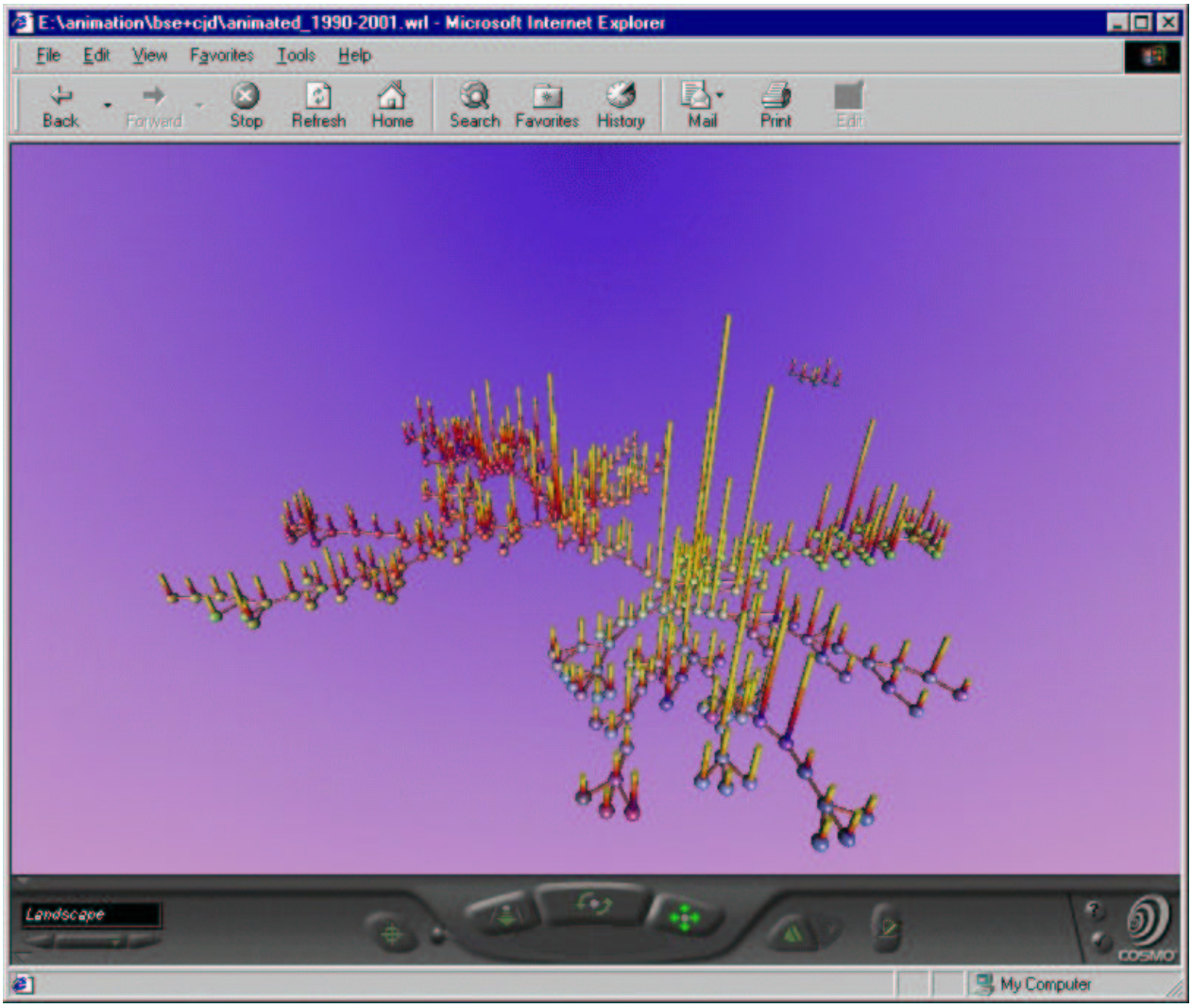

Figure 2: Dieses Bild zeigt ein Pathfinder Graph, der hier unter Microsoft Windows mittels eines VRML-Viewers angezeigt wird. Das Bild wurde freundlicherweise von Prof. Chaomei Chen zur Verfügung gestellt. Details sind der Originalveröffentlichung [4] zu entnehmen.

Pathfinder Networks sind faszinierend. Jedoch würde eine effiziente Implementation, die als Teil eines Desktop-Retrieval-Systems als Hintergrundtask läuft, sehr schwierig.

6 Von der Wunschliste zu Prioritäten

In den vorhergehenden Abschnitten wurden einige spektakuläre Resultate aktueller und auch etwas weiter zurück liegender Forschung vorgestellt, die als Inspiration und teils auch als Portierungsgrundlage dienen können. Natürlich wäre es wünschenswert, all dies auf den freien Desktop zu portieren. Jedoch muß man realistisch sein, und im Auge behalten, daß es auch für Free Software Projekte wichtig ist, sich erreichbare Zwischenziele zu setzen.

6.1 Server

Zunächst einmal ist klar, daß ohne ein Ähnlichkeitsmaß für Dokumente keines der oben vorgestellten Programme funktionsfähig wäre. Ein Index, der ein *Ranking* der Ähnlichkeit von Dokumenten bereitstellt, muß also der Kern von FdL sein. Wenn möglich, sollte dieses Ranking für *beliebige* Dokumenttypen verfügbar sein. Der Index sollte *dynamisch änderbar* sein, und zwar so effizient, daß die daraus bei normalem Betrieb entstehende Systemlast nicht ins Gewicht fällt.

Sicherheitsprobleme sollte man nicht aus dem Auge verlieren: ein Benutzer, der eine Datei nicht lesen kann, sollte sie auch nicht als Retrieval-Result vorgestellt bekommen. Ansonsten kann jeder Benutzer eines Systems die anderen Benutzer durch geschickte Wahl der Queries ausspähen. Dies ist der Grund, weshalb Programme wie `ht://dig` für Desktop Retrieval nicht ausreichen, dies war auch der (inzwischen behobene) Stolperstein von Medusa.

Die Nützlichkeit eines Retrieval-Systems ist natürlich direkt abhängig von der Zahl der Dateien, die mit ihm bearbeitet werden können. Also sind entsprechende Importfilter zu schreiben, oder zumindest ein Framework zu entwerfen, das solche Filter bereitstellt.

6.2 Desktop Environment: Nutzungs-Szenarien

Auf Seiten des Desktop Environments ist nun die Herausforderung, die vom Server bereitgestellte Funktionalität auf möglichst einheitliche Weise an möglichst vielen Stellen, zugänglich zu machen. Zum Beispiel könnte man sich die folgenden Nutzungszenarios vorstellen:

6.2.1 Retrieval überall

Diese Situation ist wohl jedem bekannt: Man schreibt einen Text, will aus einer anderen Datei einen Text einfügen und findet sie nicht sofort. Für Textdateien kann man jetzt schon `grep` oder eben `kfind` zu Rate ziehen, FdL wird zusätzlich Bildsuche bieten. Wichtig ist hierbei, den Nutzer mit möglichst wenig Mausklicks zum Erfolg kommen zu lassen.

6.2.2 Benutzer-konfigurierbare automatische Queries

Ein Schritt zu einer dem Remembrance Agent ähnlichen Funktionalität wäre es, KDE- oder GNOME-Widgets bereitzustellen, die neben ihrer eigentlichen Eingabe-Funktionalität eine Query an den FdL-Server absetzen. So könnten z.B. die Texteingabe in Text-Widgets, oder auch einfach Text- oder Bild-Markierungen automatisch in Queries umgesetzt werden. Diese Funktionalität könnte dann entweder global (d.h. für alle KDE-Applikationen oder lokal (für jede Applikation einzeln) vom Benutzer konfiguriert werden.

In beiden Fällen sollte diese Funktionalität Programmierern wie Benutzern leicht zugänglich sein. Ist sie Programmierern nicht leicht zugänglich, so wird sie nur in ausgewählten Applikationen verfügbar sein. Ist sie den Benutzern nur in ausgewählten Applikationen zugänglich, so wird sie nur von "Power-Usern" benutzt werden.

Der Vorteil der Arbeit an der Desktop-Seite von FdL ist, daß man sich schrittweise vorarbeiten kann, und sich bei jedem Schritt einer benutzbaren neuen Funktionalität erfreuen kann.

7 Die FdL Grundarchitektur

Ausschlaggebend bei der Arbeit an FdL ist die Sicht, daß wir die "kritische Masse" am ehesten erreichen, wenn wir erstens schnell Software produzieren, die den Nutzen und auch die "Coolness" von FdL sichtbar macht, und zweitens durch eine Plugin-Architektur es Freiwilligen schnell ermöglichen, sich einzubringen.

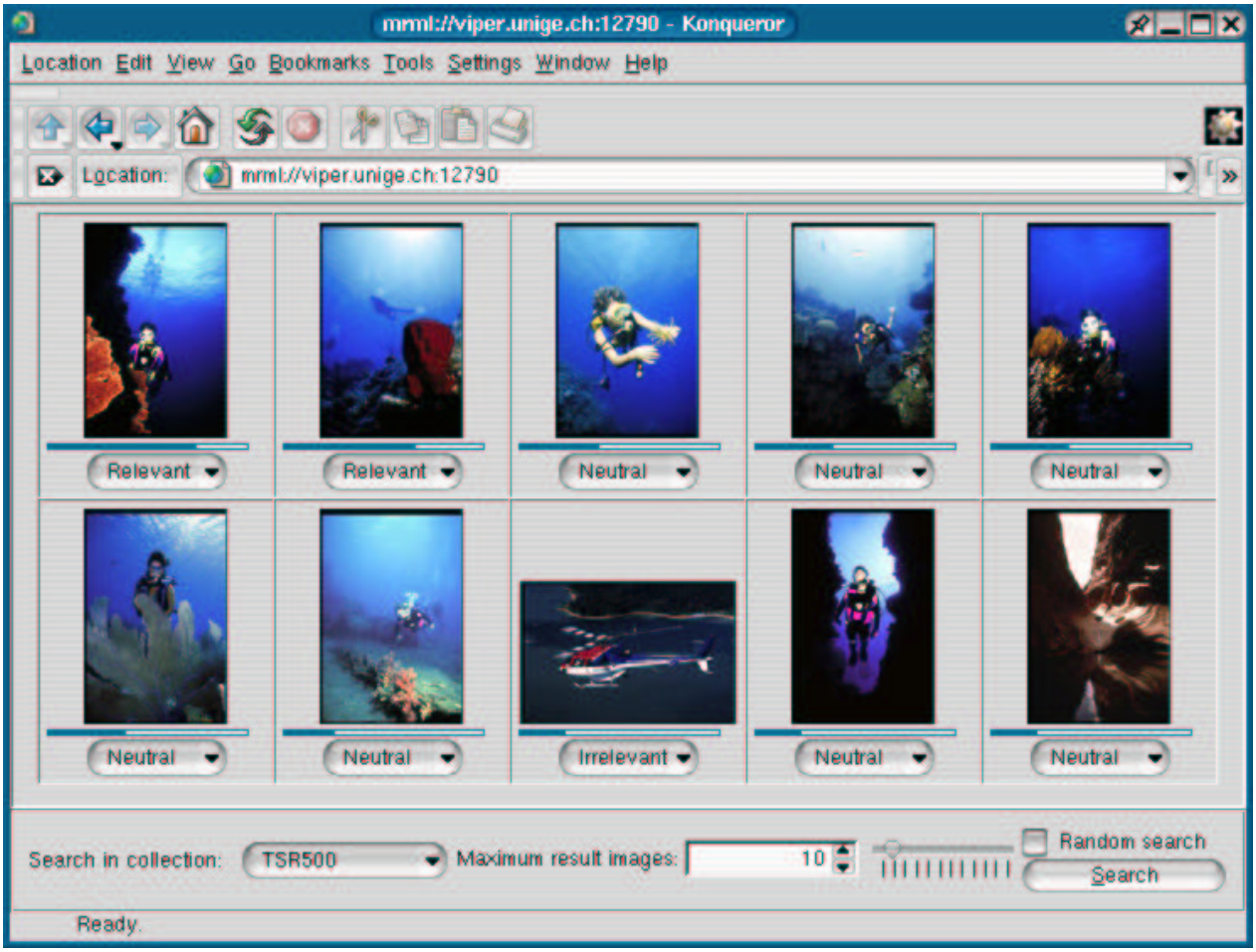

Figure 3: kmrml, ein Konqueror KPart der MRML spricht, das Protokol, das GIFT zur Kommunikation mit seinen Clients verwendet.

7.1 FdL Server

GIFT wurde als Basis für den FdL-Server gewählt. Einerseits lag dies aus persönlichen Gründen nahe (der Autor dieses Artikels ist neben seiner Beteiligung an FdL auch Maintainer des GIFT), andererseits wegen der Plugin-Architektur des GIFT. Ein weiterer Faktor war, daß MRML, das XML-basierte Protokoll, das GIFT zur Client-Server-Kommunikation verwendet gleichermaßen einfach KDE-lern und GNOMEn zugänglich sein sollte, eine gewisse Plattformunabhängigkeit also gewahrt werden würde.

Um Experimente mit Text/Bild Queries zu erleichtern wurde ein Plugin geschrieben, das GIFT mit `ht://dig` verbindet. Somit können mit GIFT jetzt auch sowohl Text- als auch Bild-Anfragen durchgeführt werden.

Für FdL wird zur Zeit die Plugin-Architektur des GIFT erweitert. Bisher ist GIFT nur flexibel in der Anfragebearbeitung. Hier kann während der Laufzeit zwischen verschiedenen Query Engines gewählt werden. Die Indexierung hingegen muß wie z.B. bei `ht://dig` per Hand gestartet werden. Die neue erweiterte Plugin-Architektur wird es ermöglichen, Indexierer für verschiedene Query Engines und Indexstrukturen auf einheitliche Weise anzusprechen. Dies wird Neulingen den Einstieg in das Projekt sehr erleichtern.

Die größte Baustelle zur Zeit ist jedoch bothrops, eine neue Query Engine für GIFT, die auf die Bedürfnisse von FdL ausgerichtet ist. Sie soll es ermöglichen, die Retrieval-Effizienz von GIFT/*Viper* mit einer leichten Aktualisierbarkeit des Indexes zu kombinieren. Ferner ermöglicht sie Queries, die auch die Struktur von Dokumenten berücksichtigen. Die in Abschnitt 2 vorgestellte Retrievalmethode ignoriert die Struktur des behandelten Textes und die Struktur der Query vollständig. Eine Anfrage nach “Mann beißt Hund” wird hier dieselben Resultate ergeben wie eine Anfrage nach “Hund beißt Mann”. Bothrops hingegen ermöglicht

es, die Struktur des Dokumentes zu berücksichtigen.

Die Relationen zwischen Dokumentteilen werden hier als Kanten (Verbindungen) eines Graphen implementiert. Sowohl die Kanten als auch die Knoten sind attributiert, d.h. mit Etikett versehen. Bothrops verwendet nun sowohl die Kanten als auch die Attribute, um die Ähnlichkeit von Graphen zu messen. Hierbei werden immer zunächst die Knoten der Graphen und dann

Während die Grundidee und erste Experimente zur Retrieval-Performance von bothrops schon seit längerem publiziert sind [6], wird im Moment daran gearbeitet, die Bedienbarkeit von bothrops, insbesondere die Indexierung während der Laufzeit, zu implementieren und zu verbessern.

7.2 FdL Client

Auf der Client-Seite existiert im Moment ein Prototyp, *kmrml*, der zeigt, wie eine Verbindung zwischen Desktop und Retrieval Engine aussehen könnte. `kmrml` ermöglicht es mittels Konqueror, dem KDE File Manager und Webbrowser, Bilder zu suchen. Dies kann dadurch geschehen, daß durch Eingabe einer `mrml://` URI Kontakt mit einem MRML Server aufgenommen wird (siehe Fig. 3). Es ist aber auch möglich, direkt aus einer Liste von Dateien ähnliche Bilder zu suchen (Fig. 4). Die Resultierenden Bilder können per Drag & Drop verschoben werden, und beispielsweise mit einem Drag als Bildschirmhintergrund verwendet werden.

Im Moment arbeitet Carsten Pfeiffer, Autor von `kmrml` daran, `kfind`, das KDE-Programm zur Dateisuche so zu erweitern, daß inhaltsbasierte Bildsuche möglich ist.

Es sei bemerkt, daß das Fehlen von GNOME-Entwicklern im FdL-Projekt nicht beabsichtigt ist. GNOMEn sind jederzeit willkommen: `fer-de-lance-developer@lists.sourceforge.net`.

Dieser Abschnitt hat gezeigt, daß es im FdL-Projekt vielversprechende Ansätze gibt, Retrieval und den Free Software Desktop einander anzunähern. Im nächsten Abschnitt sei zusammengefaßt, was noch zu tun ist, bis FdL Retrieval von Nutzerdaten in den Desktop integriert hat.

8 Programmierer gesucht

8.1 Server

Vieles muß noch auf Server-Seite geschehen. Ohne eine Sicherheits-Architektur ist Desktop-Retrieval nicht sinnvoll. Vorher wird jeder einzelne Benutzer seinen eigenen Index per Hand oder skriptgesteuert verwalten müssen, was die Effizienz nicht gerade steigert. Auch nimmt der Netto-Nutzwert von Desktop-Retrieval ab, wenn der Administrationsaufwand hoch ist.

Ohne die oben beschriebene Plugin-Architektur für Dokument-Filter sowie schnelles Einfügen, Löschen und Bewegen von Dokumenten wird das Desktop Retrieval-System ebenfalls an Nutzwert verlieren. Dieser Aufgabenkomplex hat daher im Augenblick hohe Priorität. Interessierte, die helfen wollen, melden sich bitte bei `help-gift@gnu.org`. An der selben Stelle sind auch potentielle Plugin-Schreiber sehr willkommen.

8.2 Desktop

Carsten Pfeiffer geht im KDE-Projekt den Weg, Schritt für Schritt Applikationen, die mit dem GIFT in Kontakt treten können zu KDE hinzuzufügen. Hierbei kristallisiert sich ein API für Suche in KDE heraus. Wünschenswert wäre aus Sicht des Autors, hierbei noch weiter zu gehen. Im Moment ist es beispielsweise unmöglich, aus einem File Dialog heraus nach Bildern zu suchen, die ähnlich einem gegebenen Bild sind. Wie dies am Besten zu bewerkstelligen ist, steht im Moment noch zur Diskussion.

Viel gibt es also auch im Desktop-Bereich zu tun, interessierte wenden sich bitte an `fer-de-lance-developer@lists.sourceforge.net` oder direkt an `pfeiffer@kde.org`.

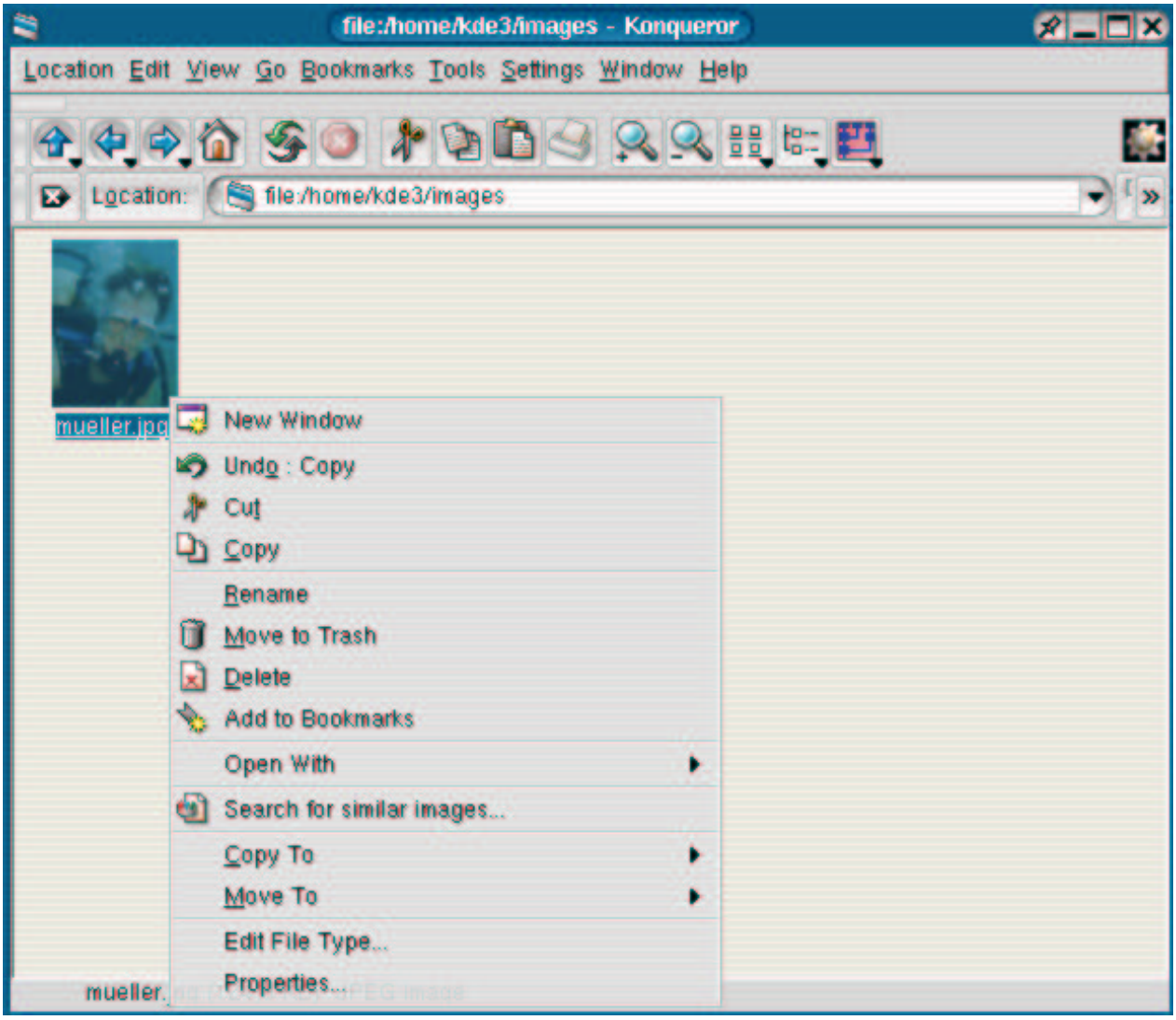

Figure 4: kmrml erlaubt aus dem Dateilisting von Konqueror heraus die Suche nach Bildern, die einem gegebenen Bild ähnlich sehen.

9 Tentative Roadmap

Wie im Laufe dieses Artikels schon klar wurde, ist FdL im Moment die Kombination zweier Projekte, die sich zwar in Absprache, jedoch organisatorisch recht unabhängig voneinander entwickeln.

Bis KDE 3.1 werden wohl kmrml und ein MRML-Fähiges kfind wohl fertig sein.

Zu demselben Zeitpunkt wird die neue, erweiterte Plugin-Architektur des GIFT ein brauchbares Stadium erreicht haben, und für Bilder lauffähig sein. Wir nehmen an, daß dies die Sichtbarkeit des FdL-Projektes genügend erhöhen wird, um Plugin-Schreiber für verschiedene Medientypen für das Projekt zu begeistern.

10 Zusammenfassung

In diesem Artikel wurde vorgestellt, wie aktuelle Forschung Retrieval Methoden benutzt, um dem Benutzer bei der Bewältigung der Informationsflut behilflich zu sein. Diese Forschungsergebnisse sind die Motivation zum Fer-de-Lance (FdL) Projekt. Als ersten Schritt strebt FdL die vollständige Integration von Retrieval-Funktionalität in das Desktop Environment an. Dies kann dem Benutzer schon bald helfen, benötigte Dateien schneller zu finden und insbesondere Suchresultate schneller zu verwenden.

Diese Funktionalität bereitzustellen ist ein ehrgeiziges Projekt. Eine Kombination exsistierender Freier Software hilft dabei, den Neuentwicklungs-Aufwand gering zu halten, und so die Erfolgschancen zu verbessern. Die Chancen stehen gut, bis zum Ende des Jahres 2002 ein gutes Zusammenspiel zwischen dem KDE-Desktop und GIFT (und anderen MRML verwendenden Retrievalsystemen) zu erreichen.

References

[1] `http://www.fer-de-lance.org`.

[2] `http://viper.unige.ch`.

[3] J. Budzik and K. J. Hammond. Watson: Anticipating and Contextualizing Information Needs. In *Proceedings of the Sixty-second Annual Meeting of the American Society for Information Science*, Medford, NJ, 1999.

[4] C. Chen, J. Kujis, and R.J. Paul. Visualizing latend domain knowledge. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 4(31):518–529, 2001.

[5] Henning Müller. Jäger des verlorenen Fotos, Das GNU Image Finding Tool für Linux in der Praxis. *c't*, pages 252–, 6 2002.

[6] Wolfgang Müller, Stéphane Marchand-Maillet, Henning Müller, David McG. Squire, and Thierry Pun. Evaluating Image Browsers Using Structured Annotation. *Journal of the American Society for Information Science and Technology*, 52(11):961–968, 2001.

[7] Bradley J. Rhodes. *Just-In-Time Information Retrieval*. PhD thesis, MIT Media Lab, May 2000.

[8] R. W. Schvaneveldt, F. T. Durso, and D. W. Dearholt. *The Psychology of Learning and Motivation*, chapter Network structures in proximity data, pages 249–284. Number 24. Academic, New York, 1989.

[9] Arnold W.M. Smeulders, Marcel Worring, Simone Santini, A. Gupta, and Ramesh Jain. Content based retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, 2000.

[10] David McG. Squire, Wolfgang Müller, Henning Müller, and Jilali Raki. Content-based query of image databases, inspirations from text retrieval: inverted files, frequency-based weights and relevance feedback. In *The 11th Scandinavian Conference on Image Analysis (SCIA'99)*, pages 143–149, Kangerlussuaq, Greenland, June 7–11 1999.