Volltextsuche am Desktop

Suchfunktionen haben am Desktop eine lange Tradition. Sie helfen, Dateien wiederzufinden, die an Orten abgelegt wurden, an denen man sie nicht vermutet oder auffinden kann. Zudem bieten diese häufig ausgefeilte Möglichkeiten, Dateien nicht nur über ihre Bezeichnung, sondern auch aufgrund ihres Inhalts zum Vorschein zu bringen. Damit ist der Weg zur Volltextsuche vorgezeichnet, wie wir sie von Suchmaschinen kennen und mittlerweile gewohnt sind. Harald Jele

»Wer Ordnung hält, ist zu faul zum Suchen«, besagt ein bekanntes Sprichwort. Dass dies nicht immer für jeden und alles zutrifft, erkennt man rasch, wenn man über Jahre und Jahrzehnte die Veränderungen seines eigenen Filesystems beobachtet: Ordnungssysteme sind nicht fest vorgegeben, nicht unbedingt logisch oder selbsterklärend und wandeln sich mit der Zeit. Somit kann es auch für den Ordentlichsten unter den Computeraffinen hilfreich und notwendig sein, sich Suchfunktionen zu bedienen, um einmal Abgelegtes auch in den hintersten Winkeln seines tief verschachtelten Speichersystems wiederzufinden. Erforderlich ist dies auch, will man eine (größere) Menge an Dateien aufgrund vorgegebener Kriterien durchsuchen, deren Inhalte man nicht oder nur oberflächlich kennt. In einem solchen Fall mag es nützlich erscheinen, sich mit den Suchfunktionen am Desktop auseinanderzusetzen und deren Möglichkeiten genauer zu betrachten.

Letzteres war ausschlaggebend für die Entstehung dieses Beitrags. Der Autor erhielt ein Konvolut von 13 GB an komprimierten und zusammengepackten Dateien, deren Inhalt ihm eventuell für andere Recherchen hilfreich sein könnten. Um herauszufinden, ob dies zutrifft, war es notwendig, diese mit Mitteln üblicher Suchfunktionen zu durchstöbern. Ein händisches Durchblättern/ Durchsuchen und schnelles Lesen wäre einerseits zu fehleranfällig und andererseits zu zeitaufwändig gewesen. 554 Dateien, jede für sich im Umfang einer durchschnittlichen Tageszeitung, gründlich mit geübten Adlersaugen zu durchforsten, ist ein Unterfangen, das einiges der Restlebensdauer des Autors gekostet und möglicherweise keine oder zu wenig Ergebnisse zum Vorschein gebracht hätte.

Als sinnvolle Vorgehensweise wurde daher gewählt, die dem Desktop eigenen Mechanismen zu untersuchen, die einer Volltextsuche gerecht werden könnten. Im vorliegenden Fall kam dabei ein Ubuntu System 20.04 LTS mit einer Gnome Oberfläche zur Anwendung, wie sie eine Standardinstallation vorsieht. Das darin integrierte Programm zur Suche ist das/der unscheinbare, aber durchaus leistungsfähige *Tracker* [1]. Parallel dazu ergab eine Internetrecherche, dass zumindest zwei aktuelle Programmpakete vorhanden sind, die sich laut den Kurzbeschreibungen für die hier gestellten Aufgaben eignen sollten. Mit *DocFetcher* [2] und *Recoll* [3] gehen zwei Anwendungen an den Start, die auf die Volltextsuche spezialisiert und für den Einsatz auf einem modernen Desktop vorgesehen sind.

Auf Serversystemen ist die Kombination aus Solr und Lucene [4] Standard, wenn Indexsysteme zur Volltextrecherche realisiert und mittels einer Suchmaschine zugänglich gemacht werden müssen. Sie zeigt, wie leistungsfähig moderne Suchsysteme sein können. Aktuelle PCs aber auch bereits zeitgemäße Notebooks bieten heutzutage ausreichend Leistung, um damit auch eine Indexierung von Dateien mittels dieser Kombination durchzuführen. Allerdings bedeutet dies einen hohen Aufwand, der für durchschnittliche Desktopbenutzer nicht zumutbar ist und für die üblichen Anforderungen an die Arbeit am PC zudem deutlich über die Stränge schlägt. Trotzdem orientieren sich viele der gängigen Anwendungen für den Desktop an den Leistungsmerkmalen dieser beiden.

Mit dem Projekt regain [5] hat es bis zur Version 2.1.0 ein Produkt gegeben, bei dem Lucene als Suchmaschine auch am Desktop eingesetzt wurde. Dieses wurde jedoch nach 2014 nicht weiter fortgeführt.

Tracker

Wer sich mit *Tracker* auseinandersetzen will, ist zuerst selbst mit einigem Suchen und Finden ausreichend beschäftigt. Innerhalb der Gnome Seiten wurde ein ebensolches Projekt angelegt und wird dort auch verwaltet. Trotzdem ist das Maß an Dokumentation und hilfreicher Erklärung von den beiden maßgebenden Entwicklern (Sam Thursfield und Carlos Garnacho) an dieser Stelle nicht wirklich ausgeschöpft. Teils finden sich überholte Informationen aus älteren Versionen, teils Ankündigungen für zukünftige, die jedoch nie umgesetzt wurden. Demnach ist der Interessierte sehr auf sich allein gestellt, wenn er herauszufinden versucht, welche Leistungsmerkmale *Tracker* aktuell aufweist. Interessant und lehrreich ist vor allem der Blog, den Sam Thursfield führt. Dort gibt er detailliert Auskunft, über die Entscheidungen, die bei der Entwicklung letztlich getroffen wurden [6].

Tracker besteht im Wesentlichen aus zwei Teilen:

- einer *SPARQL* Datenbank, die „rund um“ SQLite gebaut wurde, sowie
- den sogenannten *Tracker Miners*.

SPARQL ist eine graphenbasierte Abfragesprache, die vom W3C-Konsortium definiert wurde und seit März 2013 in einer stabilen Version vorliegt [7]. Die

Tracker Miners sind als kassische Daemons realisiert. Sie durchstöbern vorgegebene Dateipfade und bereiten die vorgefundenen Daten für die Indexierung auf.

Tracker wurde von Anfang an als Anwendung entwickelt, die möglichst unbemerkt und doch effizient im Hintergrund seine Arbeit verrichten soll. Dabei wurde zudem Augenmerk darauf gelegt, dass dieser die eigentliche Arbeit am Desktop nicht zu sehr ausbremst. Außerdem sollte sein Leistungshunger nicht dazu führen, dass allein das Indexieren sukzessive und unbemerkt die Akkus eines Notebooks leert. *Tracker* ist nicht monolithisch, sondern vielmehr modular aufgebaut. Dieser Umstand macht die Anwendung sehr fexibel, gleichzeitig aber auch ein wenig unübersichtlich. Damit ist leider auch die notwendige Einarbeitungszeit für Benutzer und Entwickler größer.

Welche Pfade und Dateitypen in den Index aufgenommen werden sollen, wählt man am Gnome Desktop eines Ubuntu Systems innerhalb der bekannten ***Einstellungen → Suchen → Orte durchsuchen → Orte/Lesezeichen/Weitere*** aus. Im Terminal wird *Tracker* anschließend angehalten und neugestartet, um die Änderungen in der Konfiguration zu übernehmen.
tracker daemon -t hält den laufenden Daemon und die damit verbundenen Prozesse an,
tracker daemon -s startet einen solchen und die zugehörigen Prozesse (neu).

Tracker führt sein Journal innerhalb des Verzeichnisses des angemeldeten Benuzters unter
~/.local/share/tracker/data/tracker-store.journal
Änderungen im Filesystem wirken sich hier aus. Ein Journal, das sich nicht ändert, bedeutet gleichzeitig, dass die *Tracker* Prozesse mit den gerade anstehenden Arbeiten fertig und alle Daten im Volltextindex vorhanden sind.

Um die aktuelle Arbeit und das Vorankommen des Prozesses besser kontrollieren zu können, bietet der *Tracker* Daemon viele Möglichkeiten. Wichtig und hilfreich können dabei folgende sein:
tracker daemon --watch (zeigt, was gerade bearbeitet wird)
tracker daemon --set-log-verbosity (beinflusst die Geschwätzigkeit des Daemon)
tracker status (zeigt den Zustand des aktuellen Indexierungsvorganges an).
Eine vollständige Übersicht zu den Parametern der *Tracker* Tools wird in [8] aufgelistet.

Das Indexieren ist im Grunde ein aufwändiger Vorgang, der am besten auf viele Prozesse aufgeteilt werden sollte. Bei *Tracker* ist dies leider nicht der Fall. Unabhängig davon, wie viele Dateien es zu bearbeiten gilt und wie viele CPUs der Rechner zur Verfügung stellt, laufen immer bloß zwei arbeitsteilige Prozesse zur Indexierung gleichzeitig.

An dieser Stelle wird auch erkenntlich, dass vielerlei Arbeiten mit dem *Tracker* einerseits über die bekannten Gnome Programme erfolgen, zudem jedoch einiges im Terminal erledigt wird oder werden kann.

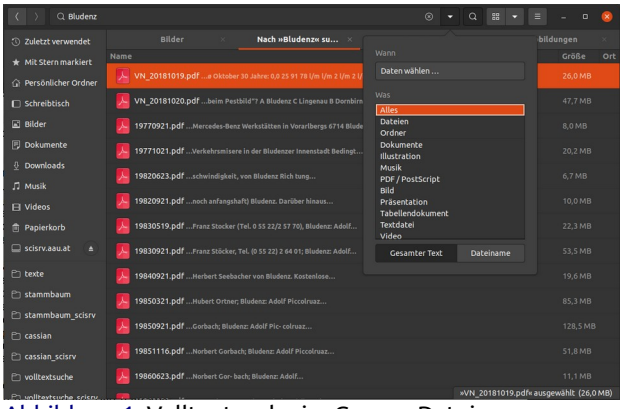

Abbildung 1: Volltextsuche im Gnome Dateimanager.

Öffnet der Benutzer den Standard Dateimanager, so kann er über dessen Suchfunktion die Suche auf Dateiinhalte ausweiten. Abb. 1 zeigt, dass beim Suchen zwischen „Dateien“ und „Gesamter Text“ gewechselt und dabei allerlei Einschränkungen gesetzt werden können. Ist „Gesamter Text“ ausgewählt, wird der eingegebene Suchbegriff (hier: „Bludenz“) auf die Inhalte gewählter Dateien angewandt und der Begriff im Volltextindex nachgeschlagen. Werden dabei im Index Treffer aufgefunden, so zeigt der Dateimanager die zugehörigen Dateien sowie eine kurze Vorschau der Fundstelle an. Will man gleichzeitig nach zwei oder mehreren Begriffen suchen, wird man mit dem Dateimanager nicht fündig. Dies liegt aber nicht daran, dass *Tracker* das nicht könnte, sondern daran, dass dies schlichtweg im Dateimanager nicht umgesetzt ist. Aber nicht nur der Dateimanager leidet an dieser unvollständigen Umsetzung. Auch die Anwendung „Dokumente“, die eine Art Dokumentenverwaltung unter Gnome sein will, hat dies bislang nicht umgesetzt und zeigt zudem keine Vorschau auf die Fundstellen und kein Highlighting der aufgefundenen Begriffe im Dokument an. Beide Informationen liefert *Tracker* jedoch.

Im Terminal wird eine gleichzeitige Suche nach zwei oder mehreren Begriffen erwartungsgemäß über die entsprechenden Begriffe zur logischen Verknüpfung (AND/OR/NOT) realisiert.
In Abb. 2 sind dazu Beispiele erkennbar.

Darüber hinausgehende, logische Verknüpfungen von Begriffen sind bewusst nicht realisiert. Thursfield schreibt dazu in seinem Blog, dass diese von den durchschnittlichen Benutzern nicht verwendet würden, wären sie vorhanden. Damit hat er wohl recht, denn wenn auch im Information Retrieval z. B. sogenannte Proximity-Operatoren (wie der NEAR Operator) definiert sind, so werden sie wohl nur von wenigen Experten in einer klassischen Volltextsuche eingesetzt. Das gleiche gilt auch für das Word-Stemming, das zwar im Blog diskutiert wird, aber letztlich nicht umgesetzt wurde.

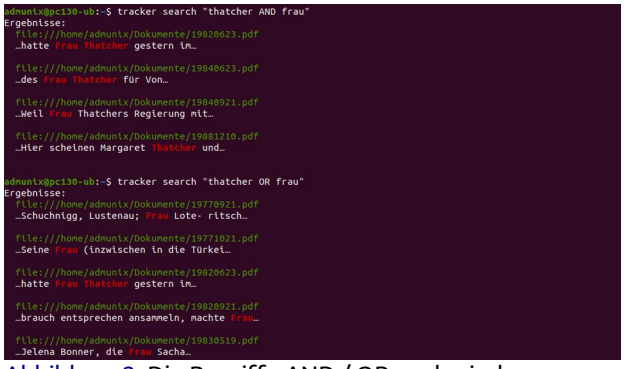

Abbildung 2: Die Begriffe AND / OR zur logischen Verknüpfung bei der Suche in Tracker.

Mit *Tracker* wurden von den Entwicklern viele der Vorgaben erfüllt, die an einen „semantischen Desktop“ gestellt werden. Dabei werden Begriffe im Index so abgebildet, dass auch jene Textelemente mitgespeichert werden, aus denen diese stammen. Damit können Begriffe nicht bloß aufgefunden sondern auch mitbestimmt werden, in welchem Textelement oder Element der Metadaten dieser vorhanden sein muss. Abgebildet wird dies in der Datenbank durch die Definition einer *Ontologie*. Im Fall von *Tracker* wird auf jene zurückgegriffen, die durch das EU-Projekt *Nepomuk* bekannt wurde [9]. Diese ist innerhalb von *Tracker* nicht hart kodiert. D. h., dass diese Ontologie durch eine beliebig andere ersetzt oder selbstständig erweitert und verändert werden kann. Die Elemente von Nepomuk sind zudem sehr übersichtlich in [10] dargestellt.

Einzelne Dateien können auf ihre Prüfung durch die Regeln der hinterlegten Ontologie im Vorfeld einer Indexierung auch analysiert werden.
Mit **tracker info „Dateiname“** gelingt dies. Abb. 3 zeigt dazu einen Teil der Ausgabe im Terminal.

Um die 554 Dateien mit insgesamt 13 GB zu indexieren benötigt *Tracker* 7 Minuten und 5 Sekunden. Damit schlägt er sich im Vergleich zu den anderen beiden Kandidaten ziemlich gut. Zu bedenken bleibt natürlich, dass nicht alle drei über den gleichen Leistungsumfang verfügen.

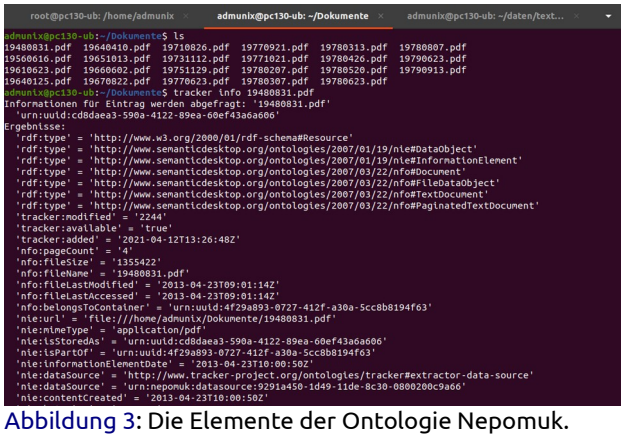

Abbildung 3: Die Elemente der Ontologie Nepomuk.

DocFetcher

Open Source ist auch dieses Programm für die Volltextsuche am Desktop: *DocFetcher* [11]. Und doch tritt es mit völlig anderen Ansprüchen auf. Es möchte quasi auf Knopfdruck möglichst schnell und leistungsstark vorgegebene Dateipfade indizieren und nimmt sich dazu jene Ressourcen, die es braucht, ohne unauffällig im Hintergrund zu werken. Trotzdem sind andere Arbeiten auf einem durchschnittlichen PC nicht völlig blockiert. Mit den Ansprüchen, die *DocFetcher* bei der Installation stellt, hinterlässt dieser außerdem einen deutlich größeren Fußabdruck am Computer als *Tracker*.

Das Programm existiert in zwei Varianten: das nicht kommerzielle *DocFetcher* sowie *DocFetcher Pro*, das seit Jänner 2021 angeboten wird und eine komplette Überarbeitung gegenüber der nicht kommerziellen Version durchgemacht hat. Zum Test kommt hier die zur Zeit aktuelle Version 1.1.22 der ersteren Variante, die als Snap Paket für Ubuntu vorhanden ist und eine Java Installation voraussetzt. *DocFetcher* ist für Linux, Windows und OS X verfügbar.

Mit **sudo apt install default-jre** kommt eine aktuelle Java Runtime Umgebung auf die Festplatte, anschließend installiert man mit **sudo snap install docfetcher** das Programm aus dem Snap Store.

Die kommerzielle Variante unterliegt zwar keinen Limitierungen hinsichtlich der Funktionen, jedoch deutlichen bei der Anzeige. So werden bloß fünf anstelle sämtlicher Ergebnisse einer Suche angezeigt. Die Entwickler meinen, dies würde auch zum umfangreichen Testen reichen.

Der monolithische Aufbau des Programmes verhindert zwar, sich bloß einzelner Module zu bedienen, gestattet aber eine einheitliche Sichtweise auf die Recherchemethoden, die umgesetzt wurden, als auch auf die Möglichkeiten der Bedienung. Beschreibung, Hilfeseiten, „Tips & Tricks“ und ein Benutzerforum liegen an der oben genannten Stelle unter [11] und sind aktuell gehalten.

Die Anzeige von *DocFetcher* ist in Rahmen gegliedert (Abb. 4). Ein klassisches Menü zur Bedienung gibt es nicht. Dies mag zu Beginn vielleicht ein wenig gewöhnungsbedürftig sein; hat man sich jedoch einmal daran gewöhnt, dass mit einem Klick der rechten Maustaste in das entsprechende Fenster die damit verknüpften Befehle aufgerufen werden, so gestaltet sich das weitere Arbeiten jedoch angenehm und auf das Wesentliche konzentriert.

Im linken unteren Frame (dem „Suchbereich“) gibt man jene Dateipfade an, die indexiert werden sollen. Auch jene Daten, die bei der Suche im Weiteren eingebunden werden, sind hier definiert. Im Rahmen darüber gibt man jene Dateitypen an, die indexiert und aufgefunden werden sollen, samt ihrer Dateigröße.

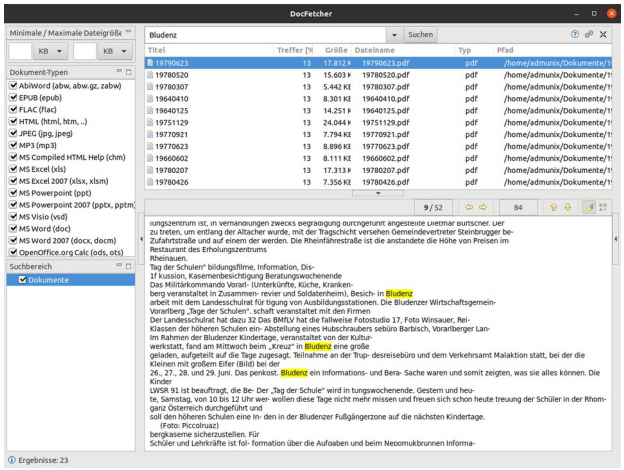

Abbildung 4: Die Anzeige von DocFetcher.

Rechts oben ist die Leiste zur Eingabe der Suchbegriffe, darunter werden jene Dateien gelistet, die zumindest einen Treffer zum Suchbegriff ergeben haben. Im Fenster rechts unten wird eine Vorschau auf die darüber gewählte Datei präsentiert. Die Fundstellen sind farblich hervorgehoben.

In der Suchleiste werden die eingegebenen Begriffe standardmäßig mit einem logischen ODER verknüpft. Dies überrascht, denn die Verknüpfung mehrerer Begriffe mit einem logischen UND ist üblich. Als Verknüpfungsoperatoren stehen AND, OR und NOT zur Verfügung. Eine Phrasensuche, bei der mehrere Begriffe in der eingegebenen Reihenfolge vorkommen müssen, wird durch Anführungszeichen markiert. Darüber hinaus existiert zudem die Möglichkeit der Nachbarschaftssuche, wobei der sogenannte Proximity-Operator (hier: die Tilde) einer Phrase nachgestellt wird. Die Eingabe von **"Bludenz Bregenz"~10** bewirkt, dass nur jene Texte als Treffer gelten, in denen die beiden Begriffe im Abstand von 10 Wörtern zueinander vorkommen. Wird explizit kein Wert angegeben, so wird Null angenommen und die beiden Begriffe als Phrase (zueinander nebenstehend) gesucht. Die Erhöhung dieses Wertes, der maximal 10 sein kann, ergibt eine größere Trefferanzahl und nähert sich jenen Treffern, die eine logische Verknüpfung mit dem Operator AND erzielt.

Weitere (sehr spezialisierte) Suchmöglichkeiten, die in *DocFetcher* realisiert wurden, sind

- das Boosting: damit können einzelne Suchbegriffe höher gewichtet werden,
- die Feldsuche: dabei können Begriffe in Texten, die Metadaten tragen, in den Feldern „Dateiname“, „Titel“ und „Autor“ aufgefunden werden, sowie
- die Bereichssuche: diese erlaubt die Suche nach Begriffen, die sich lexikographisch in einem bestimmten Bereich befinden.

Außer Textdateien kann *DocFetcher* E-Mails in seinen Index aufnehmen, darüber hinaus jedoch keine Grafik-, oder Multimediadateien. Das Indexieren solcher aber auch von komprimierten Archivdateien (wie z. B. ZIP) ist der kommerziellen Variante vorbehalten. Unter [12] listen die Entwickler die weiteren Unterschiede zwischen der kommerziellen und der nichtkommerziellen Version detailliert auf.

Ist der Index einmal erstellt und haben sich in der Zwischenzeit die zugehörigen Dateien geändert, so werden diese nicht automatisch neu indexiert. Dafür existiert ein Daemon, der Änderungen wahrnimmt und protokolliert. Dessen Aufzeichnungen können bei der Reorganisation des Gesamtindex herangezogen werden, um nur jene Dateien und Verzeichnisse abzuarbeiten, die sich zwischenzeitlich geändert haben. Abb. 5 zeigt dazu einige Möglichkeiten, die über das Menü im Suchbereich gesteuert wird.

Interessant mag für den Benutzer sein, dass *DocFetcher* zudem in einer portablen Version vorhanden ist. Eine solche Installation erlaubt, das Programm sowie die zugehörigen Dateien, von einem Rechner auf den nächsten durch einen mobilen Datenträger wie einen USB-Stick mitzunehmen und verhindert, dass auf jedem neuen Rechner *DocFetcher* installiert, die Daten übertragen und neu indexiert werden müssen.

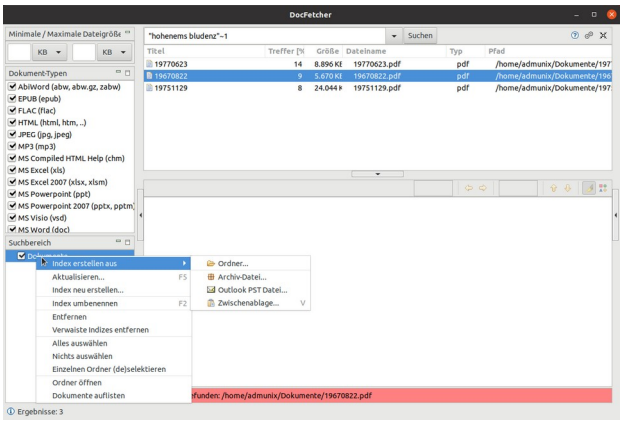

Abbildung 5: Den Index von *DocFetcher* reorganisieren.

Will man zwischen der kommerziellen und der nichtkommerziellen Ausgabe wechseln, bleibt zu beachten, dass die beiden (zur Zeit) nicht mit den selben Indexdateien arbeiten können, sodass Daten beim Wechsel zwischen den Ausgaben stets neu indexiert werden müssen. Beim Aufbau des Index muss besonderes Augenmerk jenen Dateien gewidmet werden, die nicht UTF-8 kodiert sind. Diese werden per se nicht richtig indexiert und sind damit auch nicht in jedem Fall auffindbar.

Für die Indexierung der 554 PDF-Dateien im Gesamtumfang von 13 GB benötigt *DocFetcher* nicht ganz 15 Minuten.

Recoll

Mit *Recoll* betritt der Primus der Desktop-Suchprogramme den Parcours [13]. Im Ubuntu Repositorium findet sich dazu die Version 1.26.3. Über das von den Entwicklern geführte Personal Package Archive (PPA) wird die jeweils aktuelle Version installiert (momentan v. 1.30.1). Snap Paket wird hingegen keines zur Verfügung gestellt. Getestet wurde die Version aus dem PPA. Die Installation gelingt durch:

sudo add-apt-repository ppa:recoll-backports/recoll-1.15-on
sudo apt-get update
sudo apt-get install recoll

Recoll ist für Linux, mehrere Unix Varianten, Android, Windows, OS X und sogar OS/2 verfügbar. Seine hohe Leistungsfähigkeit als vielseitiges Werkzeug am Desktop geht auf die Nutzung der Suchmaschine *Xapian* [14] zurück, das im Hintergrund die eigentliche Schwerarbeit erledigt. Die Feature List von *Xapian* ist ellenlang und das allermeiste davon wurde mit *Recoll* auch umgesetzt. Im Wesentlichen wurde die Anbindung an die Suchmachine durch eine Vielzahl an Skripten in der Programmiersprache Python realisiert. *Xapian* und damit auch *Recoll* ist tatsächlich überwiegend für die VollTEXTsuche konzipiert. Das Indexieren von Nicht-Text-Dateien tritt dabei etwas in den Hintergrund, wenngleich auch die Metadaten von Multimedia- und Grafikdateien in den Index mit aufgenommen werden können.

Wie *DocFetcher* geht auch *Recoll* davon aus, dass Texte standardmäßig UTF-8 kodiert sind und stolpert über jene, die davon abweichen. Die dafür nötigen Filter-Dateien von *Recoll* sind jedenfalls für eine große Anzahl an unterschiedlichen Kodierungen vorbereitet.

Beim ersten Start von Recoll wird man gefragt, ob man sogleich jene Verzeichnisse einrichten möchte, deren Inhalte indexiert werden sollen, oder ob dies auf einen späteren Zeitpunkt verschoben werden soll. Wählt man ersteres, wird man, wie Abb. 6 zeigt, sogleich mit mehreren Optionen konfrontiert. Das Augenmerk sollte man hier vor allem auf das Word-Stemming (Reduktion auf das Stammlexem) legen und die heranzuziehenden Sprachen wählen. Zudem werden im Feld „Unac Ausnahmen“ jene Zeichen definiert, die beim Indexieren berücksichtigt werden sollen. Andere, wie z. B. diakritische und Sonderzeichen, sowie Kombinationen aus lateinischen Grundbuchstaben und diakritischen Zeichen werden entsprechend ignoriert.

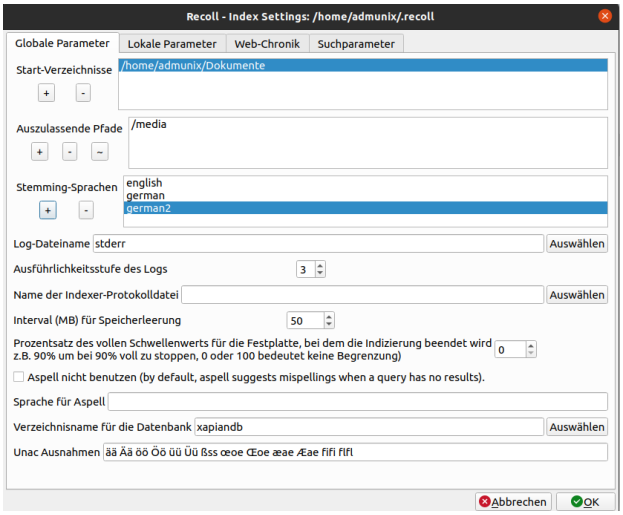

Abbildung 6: Erste Einstellungen zum Indexieren.

Der Indexlauf von Recoll lässt sich über einen Cron-Job regelmäßig durchführen, sodass neue oder geänderte Daten indexiert werden können. Einen Daemon, der die Filesysteme auf Änderungen überwacht, gibt es jedoch nicht. Entgegen der beiden anderen Dienste geht *Recoll* sogleich mit fünf Indexierungsjobs an’s Werk und entsprechend schnell erledigen diese ihre Arbeit. Nach 2,5 Minuten sind die 554 PDF-Dateien indexiert und können entsprechend durchsucht werden. Das ist gegenüber den fast 15 Minuten, die *DocFetcher* benötigt, doch ein deutlicher Unterschied.

Ist der Indexlauf beendet, meldet *Recoll*, ob sämtliche Daten indexiert werden konnten oder ob z. B. aufgrund fehlender Hilfsprogramme bestimmte Dateien ausgelassen wurden. Sobald die fehlenden Programme nachinstalliert wurden, können jene Dateien, die übersprungen wurden, nachträglich indexiert werden.

Vorbildlich ist bei *Recoll* die Präsentation der Suchergebnisse. In Abb. 7 erkennt man die Trefferliste, die bereits eine minimale Vorschau auf einige Trefferstellen bereithält und durch Anwählen des Links „Schnipsel“ alle Trefferstellen anzeigt, die innerhalb eines Dokumentes aufgefunden wurden. Mit diesen Möglichkeiten erschließt man sehr rasch jene Ergebnisse, die zur Suchanfrage besonders gut passten.

Nicht nur die Such- und Indexparameter lassen sich in *Recoll* sehr tiefgehend anpassen. Über die Benutzereinstellungen gelangt man zu jenen, mit denen sowohl die Anzeige der Oberfläche und der Ergebnislisten gesteuert, sowie Verknüpfungen zu externen Indexdateien hergestellt werden können.

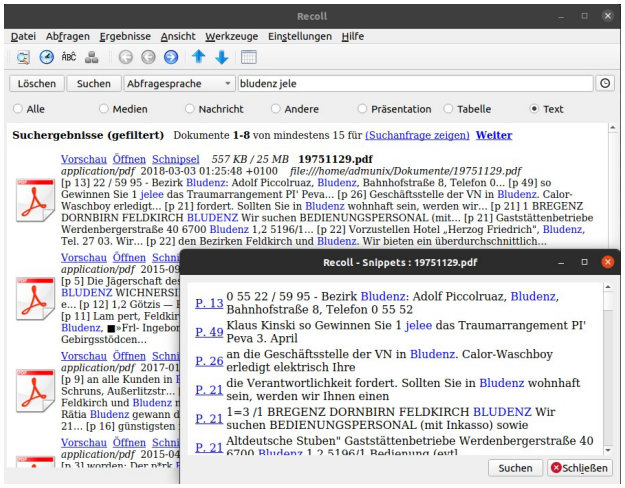

Abbildung 7: Kurzanzeige der Ergebnisse sowie der Trefferstellen im Dokument („Schnipsel“).

Besonders hilfreich ist das Anzeigefenster (Abb. 8) zur Abfragesprache, das immer dann erscheint, wenn man an der Suchanfrage arbeitet und der Cursor dabei in der Eingabezeile zum Stehen kommt. Hier werden einerseits die Verknüfungsoperatoren genannt, die zum Einsatz kommen können, andererseits wird zu jedem gleich beispielhaft gezeigt, wie diese anzuwenden sind. Bei den Booleschen Operatoren mag dies nicht immer notwendig sein, bei den anderen Verknüpfungsoperatoren ist dies hilfreich, speziell immer dann, wenn man gleichzeitig mit mehreren Suchmaschinen arbeitet, die zwar ähnlich, im Detail aber doch ein wenig unterschiedlich arbeiten.

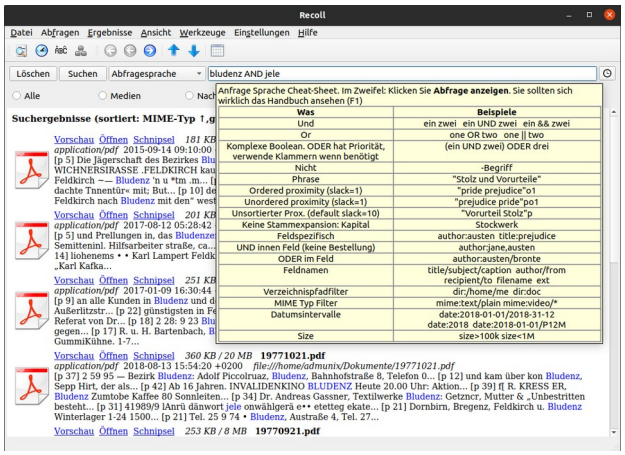

Abbildung 8: Anzeige zur Abfragesprache in Recoll.

Interessant mag für einige Linux Anwender sein, dass sich *Recoll* dafür eignet, als Standardsuchmaschine für den Desktop eingebunden zu werden. Mit dem „Gnome Shell Search Provider“ existiert ein Plugin für Gnome, das bei allen Suchaktionen greift und die Ergebnisse aus dem *Recoll*-Index zurückliefert. Für Anwender, die große Textmengen am Desktop verwalten und darin ständig mit großem Aufwand auf der Suche nach den passenden Texten wühlen, kann dies ein besonderer Vorteil sein.

Zu guter Letzt sei nicht vergessen, dass *Recoll* das Führen von Facetten ermöglicht, mit denen man (sinnvolle) Teilmengen von Ergebnisslisten erzielt, wenn diese besonders groß sind. Am Desktop können solche der Medientyp (Dateiart: Text-, Bild-, Videodatei, E-Mail etc.), Entstehungsdaten oder Datumsangaben der letzten Bearbeitung sein. In dieser Hinsicht zeigt sich Recoll weniger flexibel, denn die Facettierung ist fest vorgegeben.

	Tracker	DocFetcher	Recoll
Erkennung der Zeichenkodierung	+	-	-
Highlighting des Suchbegriffs	+	+	+
Boolesche Verknüfungsoperatoren	+	+	+
Proximity-Operatoren	-	+	+
Mehrsprachiges Word-Stemming	-	-	+
Facettierung der Suchergebnisse	-	-	+
Indexierung von Multimediadateien	+	-	+
Bindings/offene APIs	+	-	+
Gewichtung von Suchbegriffen	-	+	+
Phrasensuche	-	+	+
Mitsuchen von Synonymen	-	-	+
Mobile Version	-	+	-
Indexierung von SQL Datenbanken	-	-	+
Integration in den Desktop	+	-	+
Unterstützt mehrere Betriebssysteme	-	+	+
Wildcards (Platzhalter) zur Suche	-	+	+
Autocomplete der Suchanfrage	-	-	+
Dauer der Indexierung von 554 PDF-Dateien, 13 GB (min:sek)	07:05	14:40	02:30
Anzahl der Indexierungsprozesse	2	1	5

Fazit

Eine alternative Volltextsuche auf Linux Desktop Systemen kann sich auszahlen, wenn dort mit sehr umfangreichen Verzeichnissen und einer großen Anzahl an Dateien gearbeitet wird. Die Möglichkeiten, die die Systeme selbst bereitstellen, sind in der Regel ausreichend aber möglicherweise nicht immer einfach zugänglich oder in jedem Fall benutzerfreundlich realisiert. Das gilt in jedem Fall für das leistungsfähige Programm *Tracker*, das für die Suche am Gnome Desktop als Standard eingesetzt wird. Dabei sind die Möglichkeiten, die *Tracker* bietet in keiner der Anwendungen und bei weitem nicht genutzt.

Hier können Alternativen durchaus helfen und liefern schnell und einfach passende Ergebnisse. Der Test zeigt, dass sowohl *DocFetcher* als auch *Recoll* am Gnome Desktop eine gute Figur machen und den allergrößten Teil der Anforderungen gerecht werden. Beide weisen zudem Alleinstellungsmerkmale auf, die den einen oder anderen zu einem Einsatz verleiten können. So mag z. B. die mobile Version von *DocFetcher* für einen Anwender sinnvoll und hilfreich erscheinen, die komplexen Indexierungsmöglichkeiten oder jene zur Facettierung, die mit *Recoll* möglich sind, die Anforderungen eines anderen erfüllen.

Waren Online-Suchmaschinen in der weiteren Vergangenheit noch mit komplexen Suchmasken und -sprachen ausgestattet, so sind sie heute zumeist mit einer einfachen Eingabezeile und einer sehr zurückhaltenden Abfragesprache ausgestattet. Dieser Trend setzt sich auch am Desktop fort. Anstelle der Ansätze aus der Vergangenheit treten heutzutage eine sinnvolle Sortierung (großer) Ergebnismengen und die Möglichkeit, diese durch eine geschickt gewählte Facettierung nachträglich in immer kleinere Mengen so zu teilen, um letztlich rasch zu passenden Treffern zu gelangen.

Quellen

[1] The Tracker Project.
[https://gnome.pages.gitlab.gnome.org/tracker/]
sowie
[https://wiki.gnome.org/Projects/Tracker]

[2] DocFetcher: Open Source Desktop-Suchprogramm.
[https://docfetcher.sourceforge.net]

[3] Recoll: A desktop full-text search tool.
[https://www.lesbonscomptes.com/recoll/]

[4] Solr: An open source enterprise search platform built on Apache Lucene.
[https://solr.apache.org/]

[5] regain: Suchmaschine für den Desktop.
[http://regain.sourceforge.net/]

[6] Sam Thursfield: Tracker 3.0: Where do we go from here? Blog des Entwicklers zu Tracker.
[https://samthursfield.wordpress.com/tag/tracker/]

[7] SPARQL 1.1 Overview.
W3C Recommendation 21 March 2013.
[https://www.w3.org/TR/sparql11-overview/]

[8] Tracker: Commandline reference.
[https://gnome.pages.gitlab.gnome.org/tracker/docs/commandline/]

[9] NEPOMUK – The Social Semantic Desktop – FP6-027705 (= Networked Environment for Personalized, Ontology-based Management of Unified Knowledge).
[https://nepomuk.semanticdesktop.org/]

[10] Gnome Developer: Tracker Ontology Reference Manual.
[https://developer.gnome.org/ontology/stable/]

[11] Web Seite zu DocFetcher.
[http://docfetcher.sourceforge.net/de/index.html]

[12] DocFetcher Pro: Features.
[https://docfetcherpro.com/features/]

[13] Recoll: A desktop full-text search tool.
[https://www.lesbonscomptes.com/recoll]

[14] Xapian project website.
[https://xapian.org/]

Der Autor

Dr. Harald Jele ist Mitarbeiter an der Universität Klagenfurt. 1993 stieß er durch einen glücklichen Zufall auf Linux. Seitdem kann er sich dieses gleichermaßen am Server wie am Destop nicht mehr wegdenken.