Indizieren von Text

René Pfeiffer <pfeiffer@luchs.at>

CaT

31. August 2008

Inhaltsübersicht - Wovon reden wir?

Inhaltsübersicht - Wovon reden wir?

- Wieso Text?
- Was ist eigentlich Text?
- Bevor man indiziert, räumt man auf
- „Richtiges” Indizieren
- Backends - Google@home
- Suchen - passende Frontends
- Wege zur Verbesserung der Ansätze

Dokumente und Text

Dokumente und Text

- Sammlungen von Dokumenten sehr verbreitet
  - Office Dokumente
  - Adobe® Portable Document Format (PDF)
  - Webseiten, E-Mails, Notizen, Quellcode, . . .

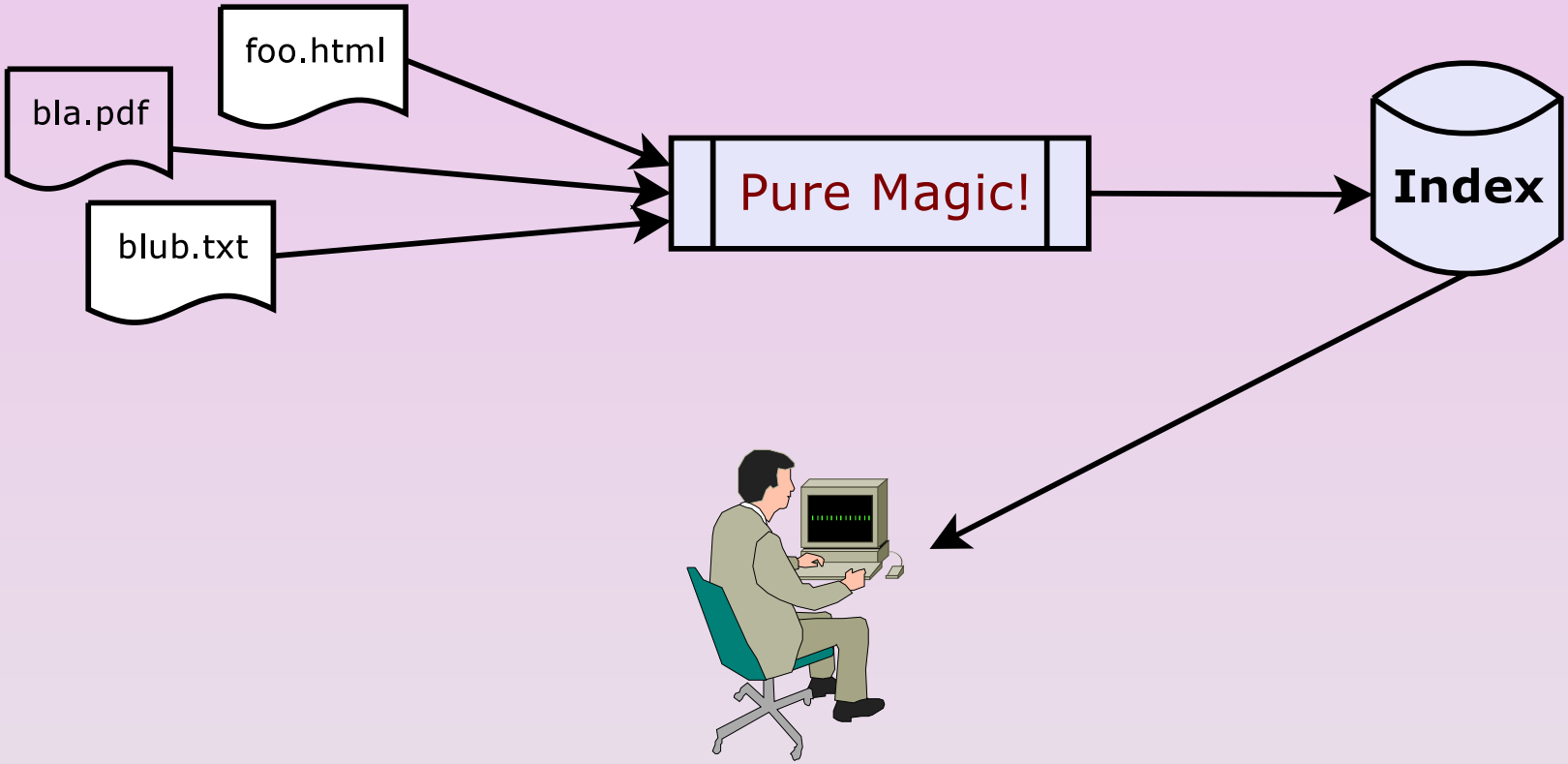
Dokumente und Text

- Sammlungen von Dokumenten sehr verbreitet
  - Office Dokumente
  - Adobe® Portable Document Format (PDF)
  - Webseiten, E-Mails, Notizen, Quellcode, ...
- Textdarstellung universell
  - leicht wandelbar
  - formatunabhängig
  - gut zu verarbeiten

Dokumente und Text

- Sammlungen von Dokumenten sehr verbreitet
  - Office Dokumente
  - Adobe® Portable Document Format (PDF)
  - Webseiten, E-Mails, Notizen, Quellcode, ...
- Textdarstellung universell
  - leicht wandelbar
  - formatunabhängig
  - gut zu verarbeiten
- Konzentration auf Dateien
  - Verzeichnisbäume mit Dateien verbreitet
  - Wandeln in Dateien oft möglich
    - ★ „Abspeichern”
    - ★ „Exportieren”

Die Idee - Wir suchen Dokumente/Inhalte



Die Probleme

- Wie stellt man interessante Dateien zusammen?
- Konvertierung PDF, HTML, PostScript®, .doc, .odt, . . . $\longrightarrow$ .txt
- Was ist eigentlich Text genau?
- Wer erstellt/verwaltet den Index?
- Wie macht man indizierte Daten zugänglich?

Finden interessanter Dateien

Finden interessanter Dateien

- Typische Fileserver halten beliebige Dateien
  - Strukturierte Bibliothek? Think monasteries!
  - Komposthaufen? Think dirt!

Finden interessanter Dateien

- Typische Fileserver halten beliebige Dateien
  - Strukturierte Bibliothek? Think monasteries!
  - Komposthaufen? Think dirt!
- Identifikation/Destillation von Textdateien schwierig
  - Bitte aufzeigen wer alle Extensions kennt!
  - `file` kennt auch nicht alles
  - Konverter müssen existieren (oder geschrieben werden)

Finden interessanter Dateien

- Typische Fileserver halten beliebige Dateien
  - Strukturierte Bibliothek? Think monasteries!
  - Komposthaufen? Think dirt!
- Identifikation/Destillation von Textdateien schwierig
  - Bitte aufzeigen wer alle Extensions kennt!
  - `file` kennt auch nicht alles
  - Konverter müssen existieren (oder geschrieben werden)
- Auswahl fällt auf Benutzer
  - Vorgabe von Extensions
  - keine Inspektion des Inhaltes durch „uns”

Finden interessanter Dateien

- Typische Fileserver halten beliebige Dateien
  - Strukturierte Bibliothek? Think monasteries!
  - Komposthaufen? Think dirt!
- Identifikation/Destillation von Textdateien schwierig
  - Bitte aufzeigen wer alle Extensions kennt!
  - `file` kennt auch nicht alles
  - Konverter müssen existieren (oder geschrieben werden)
- Auswahl fällt auf Benutzer
  - Vorgabe von Extensions
  - keine Inspektion des Inhaltes durch „uns”
- Ermittlung rekursiv durch Tree Walk
  - Dateisysteme führen (leider) keine Listen

Konverter

Konverter

- Zahlreiche Werkzeuge existieren:
  `pdftotext`, `antiword`, `unrtf`, `ooo_as_text`, `lynx`, `strings`, ...

Konverter

- Zahlreiche Werkzeuge existieren:
  `pdftotext`, `antiword`, `unrtf`, `ooo_as_text`, `lynx`, `strings`, ...
- Ausgabe ist „plain text”
  - Wir bevorzugen UTF-8 Kodierung
  - Nicht alle Konverter unterstützen das
  - Shell Pipes und anschließend `iconv` geht dann meist doch

Konverter

- Zahlreiche Werkzeuge existieren:
  `pdftotext`, `antiword`, `unrtf`, `ooo_as_text`, `lynx`, `strings`, ...
- Ausgabe ist „plain text”
  - Wir bevorzugen UTF-8 Kodierung
  - Nicht alle Konverter unterstützen das
  - Shell Pipes und anschließend `iconv` geht dann meist doch
- Konvertierter Inhalt ist ausreichend
  - Suche referenziert die Quelle (Pfad, Dateiname)
  - Quelle wird konvertiert/kopiert
  - Originale bleiben unverändert

Text? Was ist Text?

Text ist vielfältig kodiert!

Text ist vielfältig kodiert!

- `.txt` ist größte Herausforderung

Text ist vielfältig kodiert!

- `.txt` ist größte Herausforderung
- Kodierung ermitteln ist nicht trivial
  - US ASCII leicht, ISO-8859 Familie schwierig
  - Multibyte-Kodierungen leichter (aber nicht immer)
  - herstellerspezifische Kodierungen eher spezifischer

Text ist vielfältig kodiert!

- `.txt` ist größte Herausforderung
- Kodierung ermitteln ist nicht trivial
  - US ASCII leicht, ISO-8859 Familie schwierig
  - Multibyte-Kodierungen leichter (aber nicht immer)
  - herstellerspezifische Kodierungen eher spezifischer
- Erkennungmethoden
  - Perls `Encode::Guess`
  - Mozillas Universal Charset Detection
  - ICUs Character Set Detection

Text ist vielfältig kodiert!

- `.txt` ist größte Herausforderung
- Kodierung ermitteln ist nicht trivial
  - US ASCII leicht, ISO-8859 Familie schwierig
  - Multibyte-Kodierungen leichter (aber nicht immer)
  - herstellerspezifische Kodierungen eher spezifischer
- Erkennungmethoden
  - Perls `Encode::Guess`
  - Mozillas Universal Charset Detection
  - ICUs Character Set Detection
- *XYZ* `.txt` $\longrightarrow$ UTF-8 `.txt` Konverter notwendig

Aufgaben eines Indexers

Aufgaben eines Indexers

- Ein Indexer ist auch ein Konverter.

Aufgaben eines Indexers

- Ein Indexer ist auch ein Konverter.
- Indexer nimmt Textanalyse vor
  - Eliminieren von Stopwörtern
  - Tokenisierung / *Stemming*
  - Sprachanalyse (sprich Linguistik++)

Aufgaben eines Indexers

- Ein Indexer ist auch ein Konverter.
- Indexer nimmt Textanalyse vor
  - Eliminieren von Stopwörtern
  - Tokenisierung / *Stemming*
  - Sprachanalyse (sprich Linguistik++)
- Indexer speichert behandelten Text

Aufgaben eines Indexers

- Ein Indexer ist auch ein Konverter.
- Indexer nimmt Textanalyse vor
  - Eliminieren von Stopwörtern
  - Tokenisierung / *Stemming*
  - Sprachanalyse (sprich Linguistik++)
- Indexer speichert behandelten Text
- Indexer führt Suche aus
  - Bewerten der Ergebnisse (*Ranking*)
  - Finden von ähnlichen Begriffen

Aufgaben eines Indexers

- Ein Indexer ist auch ein Konverter.
- Indexer nimmt Textanalyse vor
  - Eliminieren von Stopwörtern
  - Tokenisierung / *Stemming*
  - Sprachanalyse (sprich Linguistik++)
- Indexer speichert behandelten Text
- Indexer führt Suche aus
  - Bewerten der Ergebnisse (*Ranking*)
  - Finden von ähnlichen Begriffen
- Indexer sollte schnell sein
  - Fileserver haben typischerweise Gigabytes an Daten
  - Google hat Benutzer verwöhnt

Mögliche Indexer Backends

Mögliche Indexer Backends

- MySQL Datenbankserver
  - `TEXT` Datentyp
  - Volltextindex seit Version 3.23.23

Mögliche Indexer Backends

- MySQL Datenbankserver
  - `TEXT` Datentyp
  - Volltextindex seit Version 3.23.23
- PostgreSQL Datenbankserver
  - Versteht auch Texte ⌣̈
  - Volltextindex seit Version 8.3.0 integriert
  - Modul `tsearch2` für frühere Versionen verfügbar

Mögliche Indexer Backends

- MySQL Datenbankserver
  - `TEXT` Datentyp
  - Volltextindex seit Version 3.23.23
- PostgreSQL Datenbankserver
  - Versteht auch Texte ¨‿
  - Volltextindex seit Version 8.3.0 integriert
  - Modul `tsearch2` für frühere Versionen verfügbar
- Lucene / CLucene Indexer
  - *Apache Lucene is a high-performance, full-featured text search engine library*
  - Yahoo! ist! Premium! Sponsor! von! Lucene!
  - Noch Fragen? ¨‿

MySQL Volltextindex

MySQL Volltextindex

- Anforderungen an Volltextindex
  - ausschließlich MyISAM Tabellen
  - möglich für `CHAR`, `VARCHAR` und `TEXT`

MySQL Volltextindex

- Anforderungen an Volltextindex
  - ausschließlich MyISAM Tabellen
  - möglich für `CHAR`, `VARCHAR` und `TEXT`
- `FULLTEXT()` legt Volltextindex an

MySQL Volltextindex

- Anforderungen an Volltextindex
  - ausschließlich MyISAM Tabellen
  - möglich für `CHAR`, `VARCHAR` und `TEXT`
- `FULLTEXT()` legt Volltextindex an
- eingebaute Liste von Stopwörtern
- Fokus liegt auf englischer Sprache
- sehr leicht einzusetzen

MySQL Volltextindex - Anlegen

MySQL Volltextindex - Anlegen

```
CREATE TABLE URL (
  ID bigint(20) NOT NULL auto_increment,
  Created datetime NOT NULL default '0000-00-00 00:00:00',
  Modified datetime NOT NULL default '0000-00-00 00:00:00',
  Category int(11) NOT NULL default '0',
  Summary varchar(240) default 'empty',
  URL varchar(240) NOT NULL default '',
  `Status` enum('checked','invalid','unchecked') default 'unchecked',
  Content text,
  MimeType varchar(80) default NULL,
  PRIMARY KEY (ID,URL),
  FULLTEXT KEY TextIndex (Summary,Content),
) TYPE=MyISAM;
```

MySQL Volltextindex - Abfragen

MySQL Volltextindex - Abfragen

```
mysql> SELECT ID,LEFT(URL,30),
       MATCH(Summary,Content) AGAINST ('comedy') AS Rank
       FROM URL WHERE MATCH(Summary,Content) AGAINST ('comedy');
+------+--------------------------------+------------------+
| ID   | LEFT(URL,30)                   | Rank             |
+------+--------------------------------+------------------+
| 1419 | http://www.laugh.com/          |  4.5044388771057 |
| 1464 | http://www.brunching.com/      |  2.2119855880737 |
|  932 | http://www.mittermeier.de/.    |  1.8302870988846 |
|  865 | http://www.kevinandkell.com/20 |  1.2289597988129 |
| 1412 | http://www.harveysidfisher.com | 0.93927109241486 |
|  587 | http://www.atomfilms.com/      |  0.8789678812027 |
| 3176 | http://www.monochrom.at/taugsh |  0.7746199965477 |
| 1954 | http://www.zompist.com/swedcul | 0.63005137443542 |
| 1078 | http://www.satirewire.com/news | 0.61533695459366 |
|  634 | http://www.chickenhead.com/    | 0.53878200054169 |
| 2765 | http://kaedrin.com/fun/books/c | 0.52765518426895 |
...
```

PostgreSQL Volltextindex

PostgreSQL Volltextindex

- Indizierung basiert auf `TSearch2` Modul
  - Modul war ehemals optional
  - ab Version 8.3 fester Bestandteil

PostgreSQL Volltextindex

- Indizierung basiert auf `TSearch2` Modul
  - Modul war ehemals optional
  - ab Version 8.3 fester Bestandteil
- Modul verwendet Stopwörter, *Token* und *Lexeme*
  - Datentyp `tsvector` beschreibt indizierten Text
  - Verwendung von Wörterbüchern möglich
  - Parser erweiterbar

PostgreSQL Volltextindex

- Indizierung basiert auf `TSearch2` Modul
  - Modul war ehemals optional
  - ab Version 8.3 fester Bestandteil
- Modul verwendet Stopwörter, *Token* und *Lexeme*
  - Datentyp `tsvector` beschreibt indizierten Text
  - Verwendung von Wörterbüchern möglich
  - Parser erweiterbar
- Indizierung hat Beschränkungen
  - Größe `tsvector` $\leq$ 1 Megabyte
  - Länge eines Lexemes $\leq$ 2 kB
  - Anzahl der Lexeme $\leq 2^{64}$
  - ...

PostgreSQL Volltextindex - Anlegen

PostgreSQL Volltextindex - Anlegen

```
CREATE TABLE msn (
 id_documents serial,
 filename character varying(254) default null,
 path character varying(254) default null,
 resource character varying(254) default null,
 type character varying(254) default null,
 modified timestamp with time zone default now(),
 mtime timestamp with time zone default now(),
 content text default null,
 content_fti tsvector,
 UNIQUE (filename,path,resource) );

CREATE INDEX content_idx ON msn USING gin(content_fti);

CREATE TRIGGER content_update BEFORE UPDATE OR INSERT ON msn
 FOR EACH ROW EXECUTE PROCEDURE
 tsvector_update_trigger(content_fti,'pg_catalog.english',content);
```

PostgreSQL Volltextindex - Suchen

PostgreSQL Volltextindex - Suchen

```
lynx=> SELECT filename, ts_rank(content_fti,to_tsquery('bluetooth'))
  AS rank FROM msn
  WHERE content_fti @@ to_tsquery('bluetooth')
  ORDER BY rank DESC LIMIT 15;
                 filename                    |    rank
---------------------------------------------+-----------
 Blueprinting.pdf                            | 0.0983345
 Blueprinting.pdf                            | 0.0983345
 BlueSnarf_CeBIT2004.pdf                     | 0.0982876
 BlueSnarf_CeBIT2004.pdf                     | 0.0982876
 trifinite.presentation_it-underground.pdf   | 0.0972985
 trifinite.presentation_it-underground.pdf   | 0.0972985
 mws05_slides.pdf                            | 0.0972985
 mws05_slides.pdf                            | 0.0972985
 trifinite.presentation_22c3_berlin.pdf      | 0.0970351
 trifinite.presentation_22c3_berlin.pdf      | 0.0970351
 trifinite.presentation_whatthehack_ws.pdf   | 0.0968831
 trifinite.presentation_whatthehack_ws.pdf   | 0.0968831
 wicon_2004.pdf                              | 0.0954989
 wicon_2004.pdf                              | 0.0954989
 39t2462.pdf                                 | 0.0928561
(15 rows)
```

CLucene Index

CLucene Index

- CLucene ist C++ Portierung von Lucene
  - Portierungen gibt es auch für .NET, C, Objective C, Perl, Python, PHP5, Ruby & Delphi

CLucene Index

- CLucene ist C++ Portierung von Lucene
  - Portierungen gibt es auch für .NET, C, Objective C, Perl, Python, PHP5, Ruby & Delphi
- Eigenschaften von Lucene
  - Index ist ca. 20%-30% so groß wie der Originaltext
  - *sehr* schnell ($\approx$ 20 MB/min auf Pentium M 1,5 GHz)
  - Ranking der Ergebnisse
  - eigene Sprache für Queries
  - *JStreams* - Objekte zum Einlesen von Dateien

CLucene Index

- CLucene ist C++ Portierung von Lucene
  - Portierungen gibt es auch für .NET, C, Objective C, Perl, Python, PHP5, Ruby & Delphi
- Eigenschaften von Lucene
  - Index ist ca. 20%-30% so groß wie der Originaltext
  - *sehr* schnell ($\approx$ 20 MB/min auf Pentium M 1,5 GHz)
  - Ranking der Ergebnisse
  - eigene Sprache für Queries
  - *JStreams* - Objekte zum Einlesen von Dateien
- Index ist dateibasiert
  - Lucene Index „lebt” in einem Verzeichnis
  - binärkompatibel zwischen Plattformen und Sprachen

CLucene Index - Anlegen? Suchen?

CLucene Index - Anlegen? Suchen?

- CLucene bietet ein Framework, also APIs, keine CLIs
  - Luke ist ein Java Werkzeug
  - Luke ermöglicht Inspektion eines Index

CLucene Index - Anlegen? Suchen?

- CLucene bietet ein Framework, also APIs, keine CLIs
  - Luke ist ein Java Werkzeug
  - Luke ermöglicht Inspektion eines Index
- Anlegen eines *Document* Objekts
  - Anlegen von *Label*/*Content* Paaren
  - Anhängen dieser Paare an *Document* Objekt

CLucene Index - Anlegen? Suchen?

- CLucene bietet ein Framework, also APIs, keine CLIs
  - Luke ist ein Java Werkzeug
  - Luke ermöglicht Inspektion eines Index
- Anlegen eines *Document* Objekts
  - Anlegen von *Label*/*Content* Paaren
  - Anhängen dieser Paare an *Document* Objekt
- Speichern des *Document* Objekts im Index

CLucene Index - Anlegen? Suchen?

- CLucene bietet ein Framework, also APIs, keine CLIs
  - Luke ist ein Java Werkzeug
  - Luke ermöglicht Inspektion eines Index
- Anlegen eines *Document* Objekts
  - Anlegen von *Label*/*Content* Paaren
  - Anhängen dieser Paare an *Document* Objekt
- Speichern des *Document* Objekts im Index
- Suche geht über beliebige *Label*/*Content* Paare

CLucene Index - API

CLucene Index - API

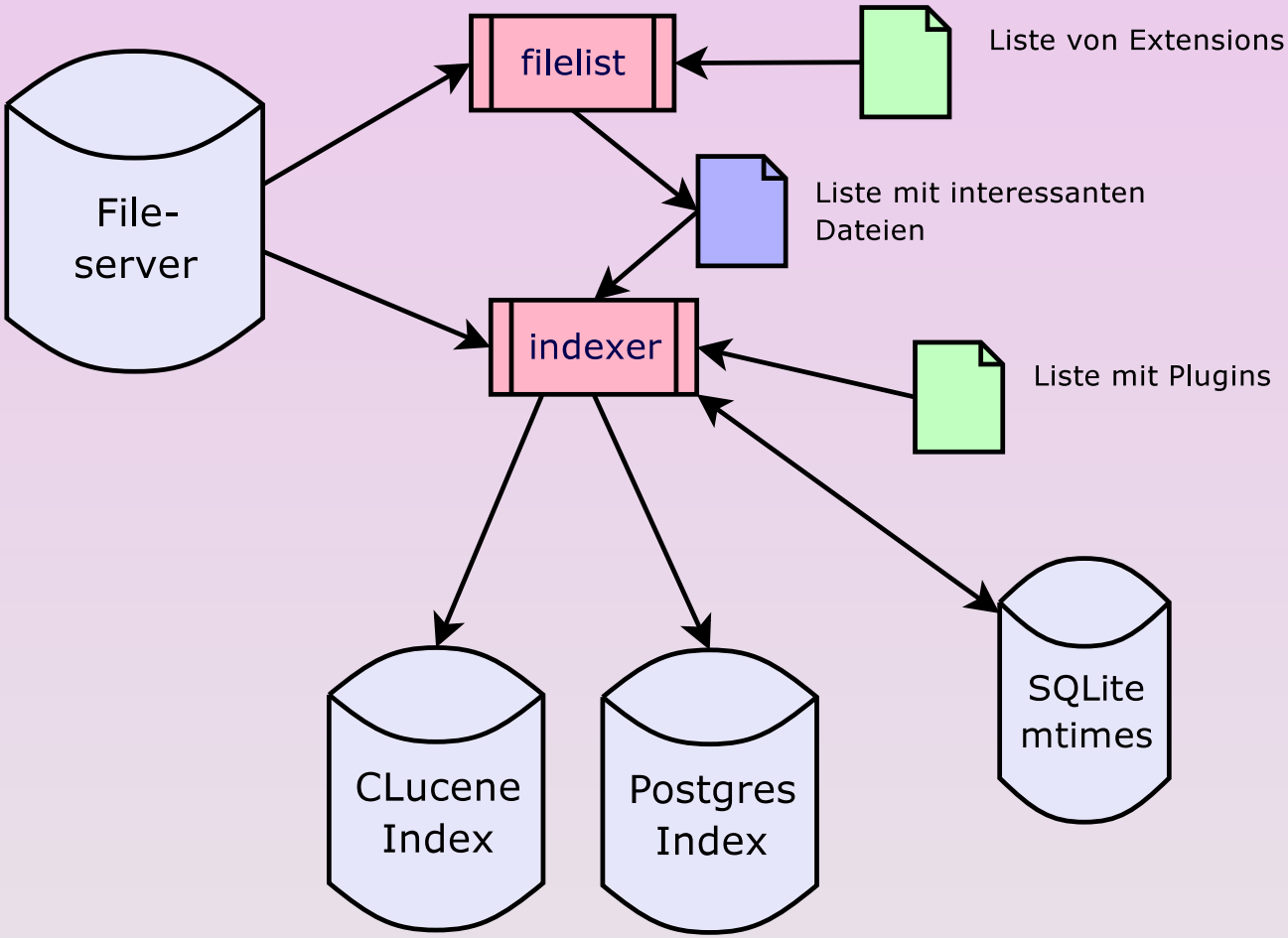
```
index_repository = new IndexWriter::IndexWriter( index_path, &analyser,
                                                 new_index, true );

filename_wide  = to_wstring(filename);
field_filename = new Field::Field( (const wchar_t*) L"filename",
                 filename_wide.c_str(),
                 true, true, true, false );

file_document = new lucene::document::Document;
file_document->add( *field_filename );

index_repository->optimize();
index_repository->close();
```

Indexstrategie

Indexstrategie

1. rekursives Durchqueren von Verzeichnissen
   1. Einschränkung aufgrund Dateiendung
   2. Aufschreiben interessanter Dateien
2. List interessanter Dateien durchgehen
   1. Vergleich der `mtime` mit einer Datenbank
   2. bereits bearbeitete Dateien ignorieren
   3. neue oder veränderte Datei bearbeiten
      1. Dokument-zu-Text Konverter aufrufen
      2. extrahierten Text in Index werfen
3. Index aufräumen
   - `VACUUM FULL ANALYZE msn;`
   - `index_repository->optimize();`

The Big Picture



Dokument-zu-Text Konverter filelist

Dokument-zu-Text Konverter filelist

- Kommando `filelist` durchquert Verzeichnis rekursiv

Dokument-zu-Text Konverter filelist

- Kommando `filelist` durchquert Verzeichnis rekursiv
- Konfigurationsdatei beschreibt interessante Dateien

```
extension = [ ".doc", ".htm", ".html", ".odp" ]
nice      = 13;
output    = "filelist.txt";
```

Dokument-zu-Text Konverter filelist

- Kommando `filelist` durchquert Verzeichnis rekursiv
- Konfigurationsdatei beschreibt interessante Dateien

```
extension = [ ".doc", ".htm", ".html", ".odp" ]
nice      = 13;
output    = "filelist.txt";
```

- *output* Datei enthält Liste
  - pro Zeile eine Datei mit Pfad
  - # leitet Kommentar ein

Dokument-zu-Text Konverter filelist

- Kommando `filelist` durchquert Verzeichnis rekursiv
- Konfigurationsdatei beschreibt interessante Dateien

```
extension = [ ".doc", ".htm", ".html", ".odp" ]
nice      = 13;
output    = "filelist.txt";
```

- *output* Datei enthält Liste
  - pro Zeile eine Datei mit Pfad
  - # leitet Kommentar ein
- Ja, ein `bash` Skript hätte es auch getan. Ich weiß.

Indexer Kommando

Indexer Kommando

- `indexer` liest von `filelist` generierte Dateiliste

Indexer Kommando

- `indexer` liest von `filelist` generierte Dateiliste
- `indexer` bedient
  - CLucene Backend
  - PostgreSQL Backend
  - SQLite Datenbank für `mtime` der Dateien (unabhängig vom Backend)

Indexer Kommando

- `indexer` liest von `filelist` generierte Dateiliste
- `indexer` bedient
  - CLucene Backend
  - PostgreSQL Backend
  - SQLite Datenbank für `mtime` der Dateien (unabhängig vom Backend)
- `indexer` konvertiert Dokumente
  - Konfigurationsdatei für externe Plugins
  - Plugins sind Shell Kommandos
  - häßlich, spart aber Libraries und *JStreams*

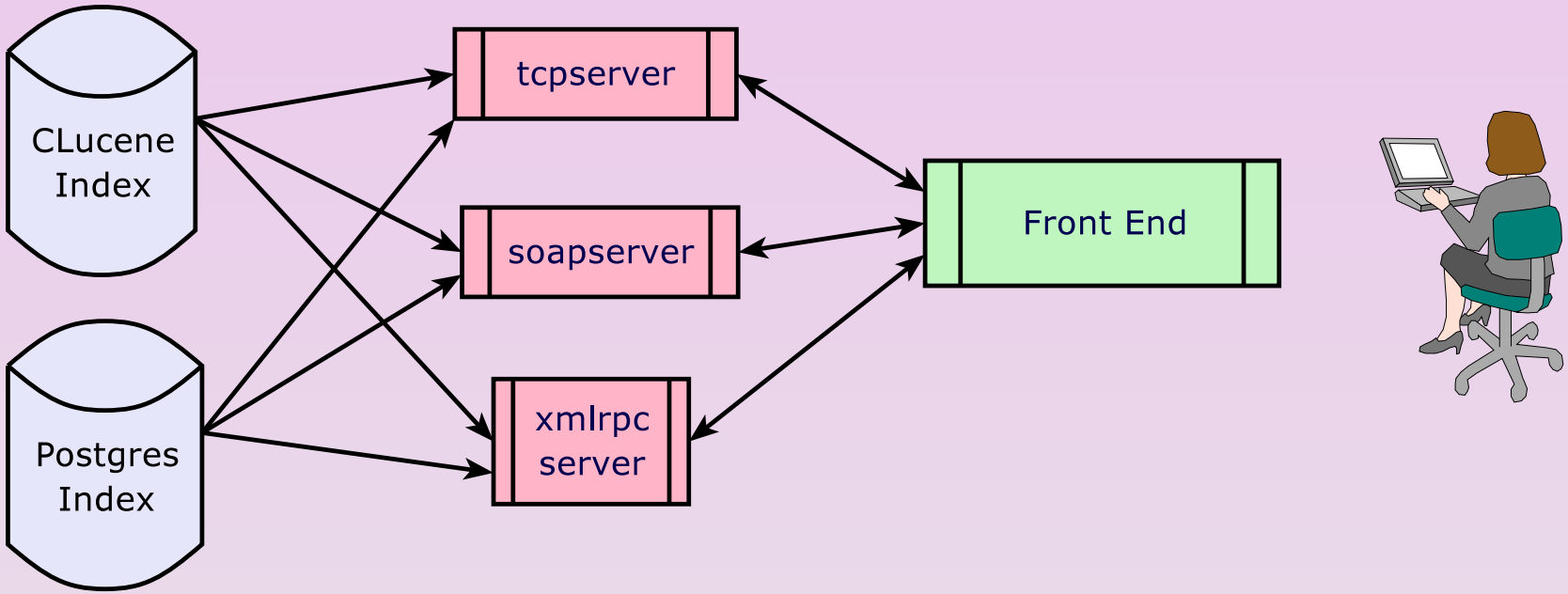
Indexer Plugins Konfiguration

Indexer Plugins Konfiguration

```
// List of known file extensions and their converters to
// plain text. This file is a configuration file of the
// indexer tool.
pdf=(pdftotext -q -nopgbrk -eol unix $IN - |
     iconv -f ISO-8859-1 -t UTF-8 -o $OUT -)
ps=(pstotext $IN | iconv -f ISO-8859-1 -t UTF-8 -o $OUT -)
doc=(antiword $IN > $OUT)
html=(elinks -dump -dump-charset ASCII $IN > $OUT)
htm=(elinks -dump -dump-charset ASCII $IN > $OUT)
odp=(ooo_as_text $IN > $OUT)
ods=(ooo_as_text $IN > $OUT)
odt=(ooo_as_text $IN > $OUT)
php=(elinks -dump -dump-charset ASCII $IN > $OUT)
rtf=(unrtf --nopict --text $IN > $OUT)
txt=(cat $IN > $OUT)
xls=(py_xls2txt $IN > $OUT)
xml=(cat $IN > $OUT)
```

Diese Datei wird von einem Boost Spirit Parser zerlegt und bearbeitet; die Syntax läßt sich leicht verändern oder erweitern.

Frontend für Suche



Das Front End muß nur Queries absetzen und die Ergebnisse aufbereiten können. Eine Abstraktionsschicht ermöglicht Front Ends in beliebiger Implementation.

Status des Projekts

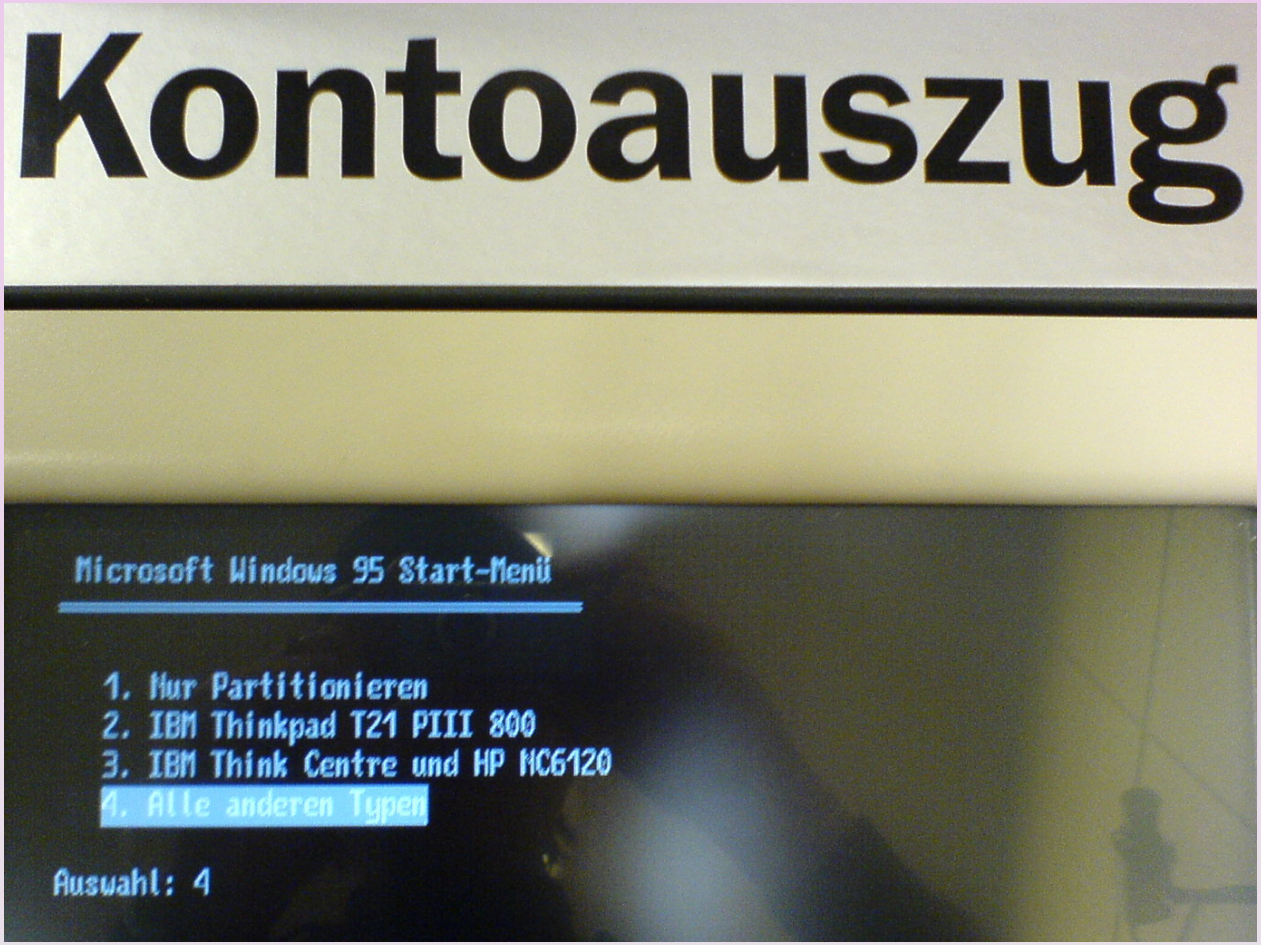
Status des Projekts

- Leicht außer Kontrolle geratene C++ Übung.
- Pre-α - *ready for production* ¨‿¨
  - Code indiziert brav in PostgreSQL & CLucene
  - Testläufe mit 500-600 Dokumenten & 1,3 GB Daten
  - fast keine Segfaults mehr ¨‿¨
- Frontends in Arbeit
  - eigenes TCP Protokoll mit TCP Server & PHP5
  - XML RPC API hängt an einem mysteriösen Bug (funktioniert nur mit US ASCII Daten)
  - SOAP API angedacht
- Code Cleanup dringend notwendig
  - sehr viel *debug stuff* noch enthalten
  - Code teilweise eine „Kann man das so machen?" Version
  - C++ Code sollte objektorientierter sein
- weitere Tests notwendig (Konverter, Dokumentauswahl)
- MySQL Support fehlt (noch)

Quellcode und Vorabversion

- `filelist`, `indexer`, `tcpserver` und weitere Codefragmente sind unter GPL 2 lizensiert. Quellcode wird gerne zur Verfügung gestellt. Patches & feedback welcome!
- Code benötigt relativ neue SQLite Library, Boost und CLucene.
- Teile des Codes sind in der Linuxgazette publiziert (Ausgabe 149 und 150).
- Das Strigi Projekt bietet sich zur Lektüre von CLucene im Einsatz an. Strigi verwendet JStreams zum Lesen von Dateien.

Noch Fragen?


Kontoauszug
Microsoft Windows 95 Start-Menü
1. Nur Partitionieren
2. IBM Thinkpad T21 PIII 800
3. IBM Think Centre und HP HC6120
4. Alle anderen Typen
Auswahl: 4

Über dieses Dokument

- Autor: René Pfeiffer
- Erstellt mit LaTeX und LaTeX Beamer Class
- Dokumentensammlung unter

  http://web.luchs.at/information/docs.php