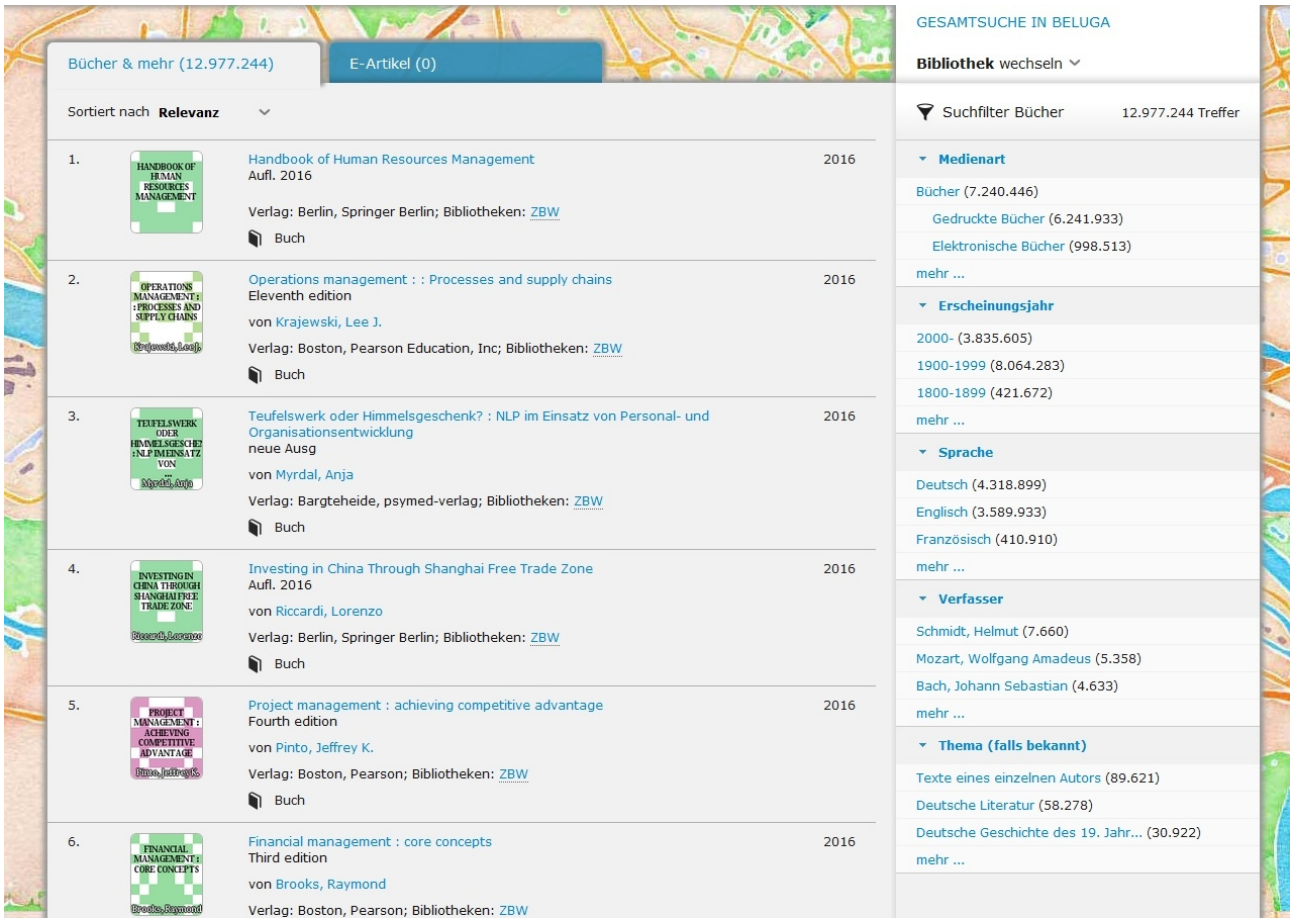
Relevantes aus der Bibliothek?

Ein heuristischer Ansatz zur Ermittlung der Relevanzsortierungsparameter auf Grundlage automatisierter statistischer Analysen.



Die relevantesten Bestände der 11 Hamburger Bibliotheken bei beluga

Staats- und Universitätsbibliothek Hamburg, 2015

Hajo Seng

Inhaltsverzeichnis

Der elektronische Bibliothekar....3
Suchmaschinen & Bibliothekskataloge....4
Der lange Weg der Metadaten....5
Eine erste Clusterung....6
Wörter zertrümmern....8
Der Dismax-Algorithmus....10
Erste heuristische Betrachtungen....13
Relevante Felder identifizieren und clustern....13
Autorenfelder....13
Titelfelder....13
Erschließungsfelder....14
volltextähnliche Felder....15
weitere Felder....15
Auf die inneren Werte kommt es an....16
Bewertungen innerhalb der Cluster....18
Bewertungen der Cluster zueinander....18
Einzelfallbetrachtungen....18
Heuristische Betrachtung mit statistischen Werkzeugen....21
Arbeiten mit statistischen Betrachtungen....24
Spezielle Anforderungen....32
Dabei sein ist alles....32
Ergebnisse nach oben treiben....32
Abfragen boosten: Medientypen....33
Funktionswerte boosten: Aktualität....36
Phrasen dreschen....37
Hauptsache das stemming stimmt....40
Finale Justierungen....41
To tie or not to tie....41
Will man wirklich alles finden?....41
Auf der Zielgeraden....42
Fazit....48
Wie es weiter geht....49
Quellen und Links (nicht nach Relevanz sortiert)....50
Suchmaschinen und Bibliothekskataloge....50
Datenschema-Dokumentationen....51
Software-Dokumentationen....51

Der elektronische Bibliothekar

Die Relevanzsortierung von Suchergebnissen über (hochgradig strukturierte) bibliothekarische Metadaten ist ein recht komplexes Thema. Bereits der Begriff „Relevanz“ lässt sich nur schwer fassen, weil er keine absolute Eigenschaft eines Mediums darstellt. Vielmehr stellt sich Relevanz in einem vielschichtigen Bedeutungsnetzwerk her, nicht nur relativ zum Suchbegriff, sondern auch zu den Erwartungen des Suchenden, ihren oder seinen Suchstrategien, zur Verfügbarkeit und Erschließung eines Mediums und vielen anderen Aspekten mehr. Selbst wenn alle in Betracht kommenden Aspekte objektiv darstellbar wären, wären sie und ihre wechselseitigen Abhängigkeiten so komplex, dass sie unmöglich adäquat behandelt werden können. Daher macht es wenig Sinn, die Relevanzsortierung als eine objektive Größe zu betrachten.

Suchmaschinenalgorithmen aber berechnen Größen und stellen damit eine Objektivität her, unabhängig davon, ob es Sinn macht oder nicht. Das trifft auch auf die Relevanzbewertung von Dokumenten zu. Die Berechnungen für eine solche Relevanzbewertung bauen auf einer maschinellen Analyse der Suchanfrage mit den gefundenen Dokumenten unter Berücksichtigung des gesamten Dokumentenbestands auf. Dafür haben sich mittlerweile Standards herausgebildet, auf denen weitgehend alle Suchmaschinen aufbauen. Die Algorithmen, die diesen Standards zu Grunde liegen, sind mit vielen Parametern justierbar. Wie auch der Begriff der Relevanz ist auch die Berechnung dieser Relevanzparametern eine ziemlich komplexe Angelegenheit, die sich bei größeren Datenbeständen nicht mehr „per Hand“, ohne Zuhilfenahme von Maschinen, behandeln lässt.

Beide Perspektiven, die eher semantisch geprägte und die der Maschine, verleiten zu dem Schluss, dass die Aufgabe, eine Relevanzsortierung zu optimieren, ausgeschlossen erscheint. Zum einen gibt es zu viele schwer oder überhaupt nicht vernünftig fassbare Faktoren, die in die Relevanz eines Dokuments (relativ zu einer Suche) eingehen, zum anderen sind selbst die objektivierbaren Faktoren am Ende so komplex zu berechnen, dass sie kaum auf einen vernünftigen Relevanzbegriff zu beziehen scheinen. Dennoch ist dieses Thema zu wichtig, als dass die Betrachtungen an diesem Punkt enden sollten.

Eine beliebte Vorgehensweise im Umgang mit komplexen und/oder unklaren Größen ist die Heuristik: Die Schwierigkeiten im Umgang mit diesen Komplexitäten und Unklarheiten reduzieren sich hier auf die Aufgabe einer geeigneten Darstellung. Geeignet heißt so, dass die relevanten Aspekte der untersuchten Größen in dieser Darstellung entdeckt werden können. Hier wird ein solcher Ansatz vorgestellt, der die Perspektive der Maschine, besser gesagt des Algorithmus, so dargestellt wird, dass in diesen Darstellungen die relevanten Aspekte der Relevanzsortierung entdeckt werden können. Dafür werden einzelne Aspekte des Algorithmus, einzelne Teilberechnungen, statistisch ausgewertet und die Auswertungen graphisch dargestellt. Dabei ergibt sich am Ende ein überraschend konsistentes Bild davon, wie sich relativ zu - zufällig ermittelten - Suchabfragen über den Datenbestand die Relevanzwerte der Dokumente zusammensetzen. Überraschend ist dabei vor allen Dingen, dass für die praktische Anwendung die Ergebnisse von den zufällig ermittelten Suchabfragen unabhängig sind.

Um dieses heuristische Ergebnis nachvollziehen zu können, sind als Vorbereitung eine Reihe von Schritten notwendig, um das Vorgehen und Funktionieren des Ranking-Algorithmus zu verstehen.

Suchmaschinen & Bibliothekskataloge

Die Suche nach Literatur kann vielfältige Formen annehmen und das Auffinden der gewünschten Literatur kann bisweilen zu einem komplexen Unterfangen geraten. Die Suche ist auch daher schwer zu fassen, weil sie in der Regel von mehreren Parametern abhängt, die in realen Situationen nie alle zugleich bekannt sind: Da ist gibt es die Suchgewohnheiten des Suchenden, die zu findende Literatur, ihre Sichtbarkeit, ihre Verfügbarkeit und nicht zuletzt die zur Suche verwendeten Instrumente. Bibliothekare versuchen hier Brücken ins Unbekannte zu bauen, indem sie wissenschaftliche Literatur erschließen. Die Metadaten, in die eine solche Erschließung am Ende gerinnt, stellen eine dem Suchenden meist unbekannte Steuerung beim Suchen dar; eine unsichtbare Hand, die ihn im Idealfall zu dem lenkt, was er zu finden erwartet. Oder auch, was er nicht zu finden erwartet, aber dennoch finden sollte, weil es in Hinblick auf seine Recherche relevant sein könnte.

Die „klassischen“ Bibliothekskataloge, wie die auf pica basierenden Bibliothekssysteme, sind ausgefeilte Suchsysteme, die zielgenaue Suchen mit Hilfe einer Reihe miteinander kombinierbarer Suchschlüssel ermöglichen. Im Idealfall liefern die Suchen in diesen Katalogen kleine aber „gute“ Treffermengen. Besonders gut eignen sich solche Kataloge für Suchen, bei denen die Suchenden bereits konkrete Vorstellungen von dem haben, was sie finden wollen - und wie sie es finden können. Eine neuere Art von Bibliothekskatalogen sind die Index-basierten Kataloge. Anders als die hochgradig strukturierten Metadaten eines pica-Katalogs, werden die Metadaten hier in einer flachen Struktur, dem Index, vorgehalten. Die Suchen liefern hier im Idealfall große Ergebnismengen, die aber „nach Qualität“ geordnet dargestellt werden („Relevanzsortierung“) und sukzessive eingegrenzt werden können („Facettierung“). Index-basierte Kataloge sind ein heuristisches Suchwerkzeug, das sich besonders gut eignet, in bestimmten Suchbereichen zu stöbern und dabei Literatur zu entdecken, die anfangs nicht erwartet wurde. Natürlich kann man in den klassischen Katalogen auch stöbern und in Index-basierten Katalogen gezielt suchen - die jeweiligen Stärken sind in diesen beiden Katalogtypen aber unterschiedlich.

Im Folgenden werden als Beispiele Suchen im Index (findex) der Verbundzentrale Göttingen (VZG) auf der Grundlage des darauf basierenden Konsortialkatalogs Beluga analysiert. Beluga beinhaltet die Katalogdaten von 11 Hamburger Bibliotheken mit insgesamt fast 13 Millionen Datensätzen (Stand Mai 2015). Die Bibliotheken sind im Einzelnen:

Bibliothekssystem Universität Hamburg	***8.601.229 Datensätze***
Zentralbibliothek Wirtschaftswissenschaften	***4.852.720 Datensätze***
Helmut-Schmidt-Universität	***1.837.414 Datensätze***
Technische Universität Hamburg-Harburg	***855.485 Datensätze***
Hochschule für angewandte Wissenschaften	***404.864 Datensätze***
Commerzbibliothek	***348.096 Datensätze***
Staatsarchiv	***234.842 Datensätze***
Hamburger Lehrerbibliothek	***221.517 Datensätze***
Hochschule für Musik und Theater	***188.142 Datensätze***
Hafencity Universität	***179.632 Datensätze***
Hamburg Media School	***16.573 Datensätze***

Der lange Weg der Metadaten

Die Katalogmetadaten liegen ursprünglich in einem pica-Format vor und werden von diesem in ein „solr-marc“-Format, d.h. ein für die Indexierung angepasstes Marc-Format übertragen. Aus diesen Marc-Daten werden dann die (flachen) Felder des Index befüllt. Die VZG legt bis dato die pica-marc-Konkordanz, die ihrer Datenübertragung zu Grunde liegt, nicht offen. Sie wird weitgehend der pica-marc-Konkordanz der Deutschen Nationalbibliothek entsprechen, aber Abweichungen davon sind unbekannt. Die Übertragung der solr-marc-Daten in den Index ist in einem als Datei veröffentlichtem Mapping offengelegt, zu dem mehrere - ebenfalls offengelegte - Skripte gehören. Diese Beschreibungen sind sehr technisch gehalten. Für eine Beurteilung der Relevanzsortierung genügt es völlig, sich das direkte Mapping, ohne die Skripte, anzusehen. Hier ein paar Beispiele für diesen Datentransfer:

„Alsterdampfer : die weisse Flotte am Jungfernstieg“, Titel- und Autorendaten

pica:
021A $a *@Alsterdampfer* ***$d*** *die weisse Flotte am Jungfernstieg* ***$h*** *Hans H. Müller*
037A $a *Frühere Ausg. u.d.T.: Müller, Hans H.: Chronik der weißen Flotte*
044K $8 *Alster, Fahrgastschifffahrt, Binnenschifffahrt*
244Z/01 $8 *Alster: Schiffahrt*
245Z/02 $a *Bibliothek Gerhard Ahrens*

marc 21:
100\|a *Müller, Hans H.*
245\|a *Alsterdampfer* ***\|b*** *die weisse Flotte am Jungfernstieg* ***\|c*** *Hans H. Müller*
500\|a *Frühere Ausg. u.d.T.: Müller, Hans H.: Chronik der weißen Flotte*
650\|a *Fahrgastschifffahrt, Binnenschifffahrt*
651\|a *Alster*
982\|a *Firmen-Festschrift, FA-NEU, Alster: Schiffahrt, Alster, 21, Hamburg, Schiffahrt*
982\|a *Bibliothek Gerhard Ahrens, Alster, Buch, Neuzeit <i.G., um 1500 bis zur Gegenwart>, ...*

index:
title_short *Alsterdampfer*
title *Alsterdampfer die weisse Flotte am Jungfernstieg*
title_sub *die weisse Flotte am Jungfernstieg*
title_full *Alsterdampfer die weisse Flotte am Jungfernstieg Hans H. Müller*
author *Müller, Hans H.*
abstract *Frühere Ausg. u.d.T.: Müller, Hans H.: Chronik der weißen Flotte*
topic *misc HE669.Z7* ***(marc 050),***
sdnb 21a ***(marc 084),***
20 Firmen-Festschrift, 20 FA-NEU, 22 Alster: Schiffahrt, 33 Alster, 33 21, 203 Hamburg, 203 Schiffahrt ***(marc 982)***
gbv Fahrgastschifffahrt, gbv Binnenschifffahrt ***(marc 650),***
gbv Alster ***(marc 651)***
class *HE669.Z7, 21a sdnb* ***(marc 50 und 84)***
22 02 00 Bibliothek Gerhard Ahrens,
33 00 00 (DE-601)050610511 x95 \|g\| Alster,
33 00 10 (DE-601)050611437 524070 \|s\| Alsterschifffahrt,
33 00 20 (DE-601)050625446 y70 \|f\| Buch,
33 00 25 (DE-601)050592521 z50 \|z\| Neuzeit <i.G., um 1500 bis zur Gegenwart>, ... ***(marc 983)***

An diesem Beispiel lassen sich einige Effekte erkennen, die für diesen beschriebenen Datentransfer typischer Weise vorkommen können. In den pica-Datensätzen kommen beispielsweise 244Z und 245Z nicht in allen Katalogisaten der an beluga beteiligten Bibliotheken vor. Global vergebene Schlagwörter sind im solr-marc in den Feldern 6XX zu finden, während die 9XX-Felder lokal vergebene Schlagwörter bzw. Klassifikationen enthalten. Beide Arten von Schlagwörtern befinden sich am Ende aber in einem Indexfeld, nämlich topic. Auch die Transformierung des Inhalts von 500a in das Feld abstract scheint nicht ganz treffend zu sein. Solche Effekte kommen nicht nur daher, dass sich die Formate, pica, marc und solr-index, nicht eins zu eins aufeinander abbilden lassen. Sie haben ihre Ursache auch darin, dass diese Daten maschinell übersetzt werden und lokale Katalogisierungsstrategien vieler einzelner Bibliotheken in ein für alle gültiges Schema übertragen werden. Mit dieser Erfahrung im Hinterkopf ermöglichen die nun - zumindest grob - bekannten Übertragungswege eine erste Strukturierung der Indexfelder.

Eine erste Clusterung

Für eine sinnvolle Betrachtung der einzelnen Indexfelder und ihrer Abhängigkeiten ist es sinnvoll, sie zunächst in einzelne Cluster zu unterteilen. Als Grundlage für eine erste Clusterung dient das Mappung der marc-Daten in die Indexfelder. Die folgenden Beispiele zeigen Ausschnitte aus dem Mapping der VZG. Dabei werden hier nur Felder betrachtet, die Metadaten beinhalten, die für eine allgemeine Katalogsuche relevant sein könnten. Felder, die beispielsweise Sprachcodes, ISBN-Nummern oder Signaturen enthalten, werden für Spezialsuchen relevant sein, die dann nur auf dezidierten Feldern suchen, aber nicht für eine allgemeine, das Gros der Felder einschließende Suche.

Es gibt bei der VZG, wohl aber auch sonst, ein Feld, das weitgehend alle im solr-marc vorhandenen Daten beinhaltet, dass ***allfields***-Feld:

allfields = custom, getAllSearchableFields(001, 999)

Dieses Feld für die allgemeine Suche mitzuverwenden hat sich als problematisch erwiesen, da es Metadaten mit einschließt, die bei den Suchen nicht berücksichtigt werden sollten. Wenn etwa ein Buch über Schnecken von Erich Ziegelmeier die Signatur „BI 46 Ziege“ hat, wird es auch von Nutzern gefunden, die eigentlich etwas über Ziegen erfahren wollen.

Der Metadatenbereich, der am einfachsten zu identifizieren ist, sind die Titeldaten. Hier führt die VZG folgendes Mapping durch:

title = 245ab:440ap:800abcdfpqt:830ap, first
title_sub = 245b, first
title_short = 245ap, first
title_full = custom, getAllSubfields(245, " ")
title_auth = 245ab, first
title_alt = 130adfgklnpst:240a:246a:730adfgklnpst:740a
title_old = 780ast
title_new = 785ast
series = 440ap:800abcdfpqt:830ap
series2 = 490a
journal = 773t

Dabei lassen sich folgende Abhängigkeiten feststellen (> steht für „enthält“, + für „Vereinigung ohne Überschneidung“ bzw. „disjunkte Vereinigung“ und >~ für „enthält weitgehend“):

title = series + title_auth
title_full > title_auth > title_sub
title_full > title_short

title_alt, title_old, title_new, series2 und ***journal*** sind jeweils eigenständige Felder, die keine Überschneidung mit anderen Feldern haben.

Analog die Autorendaten:

author	***= 100abcd, first***
author2	***= custom, removeTrailingPunct(110ab:111ab:700abcd:710ab:711ab)***
author_additional	***= 505r***

Hier erweisen sich alle drei Indexfelder als voneinander unabhängig. Zu beachten ist hier, dass im Feld author2 sowohl Personen- als auch Institutionendaten zu finden sein können.

Etwas umfangreicher fallen die Mappings der Schlagwort- und Klassifikationsdaten aus:

topic_title	***= custom, getAllSubfields(050:060:082:084:982:245:440:800:830: 600:610:630:650:689, " ")***
topic	***= custom, getAllSubfields(050:060:082:084:982:600:610:630:650:689, " ")***
class	***= custom, getAllSubfields(050:060:082:084:983, " ")***
class_local	***= custom, getAllSubfields(983, " ")***
geographic	***= custom, getAllSubfields(651, " ")***

und die etwas spezielleren Felder:

dewey-ones	***= getDeweyNumber(082a:083a, 1)***
dewey-tens	***= getDeweyNumber(082a:083a, 10)***
dewey-hundrets	***= getDeweyNumber(082a:083a, 100)***
bklname	***= script(bcl.bsh), getBklName (verwendet 082)***

Die Skripte für bklnumber und bklname (die Basisklassifikation) werten das marc-Feld 084 aus. Die hier genannten dewey-Felder sind die einzigen, die auch mit Text befüllt werden. Bei den Schlagwörtern und Klassifikationen ist zu beachten, dass die lokalen Schlagwörter (marc 982 und 983) nicht von den globalen getrennt werden. Es ergibt sich folgendes Abhängigkeitsbild:

topic_title > topic und topic_title > title
class > class_local
topic >~ class
class > bklname und topic > bklname
class > dewey-xyz und topic > dewey-xyz

Schließlich gibt es noch Felder, die mutmaßlich viel Volltext enthalten können, in diesem Fall abstract und contents, die beide eigenständig sind:

abstract	***= 520a:500a:787t***
contents	***= 505a:505t***

Insgesamt legt diese erste Betrachtung des marc-index-Mappings vier verschiedene Feldcluster nahe: ***Titeldaten***, ***Autorendaten***, ***Schlagwörter und Klassifikationen*** und ***volltextähnliche Felder***. Bevor diese Clusterung weiter analysiert wird, macht es Sinn ein paar grundlegende Mechanismen der solr-basierten Suche darzulegen.

Wörter zertrümmern

Die Feldinhalte werden von solr nicht als Ganzes analysiert, sondern vor der Analyse in einzelne Bestandteile zerlegt, den sog. Tokens. Diese Tokens sind in der Regel ganze Wörter - in der Regel. Die Art und Weise, wie die Feldinhalte zerlegt werden, kann in solr sehr flexibel konfiguriert werden. Welche Konfiguration vorliegt, ist der schema.xml zu entnehmen. Interessant ist dort die Definition der Text-Felder (im Gegensatz zu den String-Feldern, deren Inhalte grundsätzlich nicht zerlegt werden). Der entsprechende Eintrag im Index sieht wie folgt aus:

```
<fieldType name="text" class="solr.TextField" positionIncrementGap="100">
  <analyzer type="index">
    ...
    <tokenizer class="solr.ICUTokenizerFactory"/>
      <filter class="solr.WordDelimiterFilterFactory"
                generateWordParts="1"
                generateNumberParts="1"
                catenateWords="1"
                catenateNumbers="1"
                catenateAll="0"
                splitOnCaseChange="1"/>
      ...
      <filter class="solr.SnowballPorterFilterFactory" language="English"/>
      ...
  </analyzer>
  <analyzer type="query">
    ...
      <filter class="solr.WordDelimiterFilterFactory"
                generateWordParts="1"
                generateNumberParts="1"
                catenateWords="0"
                catenateNumbers="0"
                catenateAll="0"
                splitOnCaseChange="1"/>
      ...
  </analyzer>
</fieldType>
```

Das sind jetzt nur die Ausschnitte, die im folgenden erörtert werden. Der Teil ***<analyzer type="index">*** beschriebt die Bearbeitung der Feldinhalte vor der Analyse, ***<analyzer type="query">*** die Bearbeitung der Suche. Idealer Weise sind beide Teile identisch oder korrespondieren in geeigneter Weise miteinander. Hier sind die Parameter ***catenateWords*** und ***catenateNumbers*** unterschiedlich, die anzeigen, ob Wort- und Zahlenbestandteile verbunden werden sollen oder nicht.

Der findex der VZG verwendet für Textfelder den ICU-Tokenizer; das ist immer eine gute Wahl, wenn man es mit unterschiedlichen Sprachen und unterschiedlichen Schriftzeichen zu tun hat. Der ICU-Tokenizer ermittelt Wortgrenzen nach dem Unicode-Standard, d.h. er erkennt Leerzeichen und Interpunktionen (außer dem .) als Trennzeichen und analysiert die Worte in ihrer jeweiligen Kleinschreibung. Außerdem erkennt der Tokenizer Wortgrenzen auch in vielen Sprachen, in denen einzelne Wörter nicht durch Leerzeichen getrennt werden, etwa fernöstliche Sprachen.

Neben den nach Unicode definierten Wortgrenzen (bei Buchstabenschriften Leerzeichen), werden Wörter auch an Übergängen von Klein- zu Großschreibung (***splitOnCaseChange=1***) und an Übergängen von Buchstaben zu Zahlen und umgekehrt (***splitOnNumerics=1*** per default) in Wortteile getrennt, wenn wie hier die Parameter ***generateWordParts*** und ***generateNumberParts*** gesetzt sind. Im Index-Teil sind. ***catenateWords*** und ***catenateNumbers*** jeweils auf 1 gesetzt, was bedeutet, dass jeweils eine maximale Anzahl von (nicht an Leerzeichen getrennten) Wortteilen eines Wortes zusätzlich zusammengesetzt werden, wenn das Wort insgesamt nur aus Buchstaben oder nur aus Ziffern besteht (***catenateWords=0***). So ergeben sich als Beispiele folgende Zerlegungen:

„JavaScript 2.x ist mega-cool“ im Index:

java	***script***	***2.x***	***ist***	***mega***	***cool***
	javascript				***megacool***

oder

„JavaScript 2-0 ist mega-cool“ im Index:

java	***script***	***2***	***0***	***ist***	***mega***	***cool***
	javascript		***20***			***megacool***

Zu beachten sind hier auch die Reihenfolgen der Tokens, die insbesondere für die Phrasensuche wichtig sind, was später noch besprochen wird. Durch spezielle Mappings (die ebenfalls der schema.xml zu entnehmen sind), wird beim findex das „-“ in „minus“ übersetzt und fungiert daher nicht mehr als Worttrenner. Ich tue hier und im Folgenden so, als gäbe es diese Mappings nicht. Im query-Teil sind die Parameter ***catenateWords*** und ***catenateNumbers*** jeweils auf 0 gesetzt, sodass sich hier eine andere Zerlegung ergibt:

„JavaScript 2.x ist mega-cool“ in der Anfrage:

java	***script***	***2.x***	***ist***	***mega***	***cool***

bzw.

„JavaScript 2-0 ist mega-cool“ in der Anfrage:

java	***script***	***2***	***0***	***ist***	***mega***	***cool***

Zur Verarbeitung der Feldinhalte trägt auch der stemming-Algorithmus bei, hier der Snowball-Porter-Stemmer. Stemming bedeutet, dass die Flektierung von Wörtern neutralisiert wird, indem die Wörter auf grammatikalische Grundformen reduziert werden. So wird beispielsweise aus Hauses und Häuser die Grundform Haus. Der stemming-Algorithmus arbeitet auf Grundlage der Sprache, in der er die Wörter vermutet. Der Snowball-Porter-Stemmer arbeitet musterbasiert - im Unterschied zu tabellenbasiert. D.h., er benutzt bestimmte - von der jeweiligen Sprache abhängige - Muster, um die Grundformen zu raten. Das ist sehr performant, unter Umständen aber nicht so präzise, wie ein tabellenbasierter Stemmer.

Neben diesen Verarbeitungsmechanismen gibt es noch eine (im Fall der VZG kleine) Liste an Zeichen, die vor dem Aufsplitten ersetzt werden (z.B. „-“ durch „minus“) und eine (hier ebenfalls kleine) Liste von Wörtern, die synonym behandelt werden sollen (z.B. „one“, „1“ und „i“).

Was mit einem Suchbegriff bei einer solr-Abfrage passiert, lässt sich in den weiteren Angaben zur Ergebnismenge ablesen, die man erhält, indem man den debugquery-Parameter auf on stellt. Diese Darstellung erscheint auf dem ersten Blick etwas kryptisch, lässt sich aber doch leicht entschlüsseln:

„JavaScript 2.x ist mega-cool“:

```
<str name="querystring">JavaScript 2.x ist mega-cool</str>
<str name="parsedquery">
    (+((DisjunctionMaxquery((allfields:"java script"))
    DisjunctionMaxquery((allfields:"(ii 2 two) x"))
    DisjunctionMaxquery((allfields:ist))
    DisjunctionMaxquery((allfields:"mega cool")))~4))/no_coord
</str>
<str name="parsedquery_toString">
    +(((allfields:"java script")
    (allfields:"(ii 2 two) x")
    (allfields:ist)
    (allfields:"mega cool"))~4)
</str>
<str name="QParser">ExtendedDismaxQParser</str>
```

Da kein bestimmtes Feld befragt wird, wird hier eine „AllFields“-Abfrage durchgeführt, die das Feld „allfields“ abfragt. Die Suchanfrage wird demnach in die Teilanfragen „java script“, „(ii 2 two) x“, „ist“ und „mega cool“ aufgeteilt; „~4“ bedeutet, dass Tokens mit dem Abstand bis zu 4 noch als benachbart angesehen werden, sodass dann auch ein Feldeintrag „Wie mega-cool ist JavaScript 2.x?“ einen Treffer erzeugen würde. Am Ende lässt sich erkennen, dass die Suchanfrage von dem Extended Dismax Parser bearbeitet wurde. Dismax ist eine Algorithmus-Implementierung, die das Matching von Suchanfrage und Feldinhalt, bzw. dem Dokument insgesamt, bewertet.

Der Dismax-Algorithmus

Dismax steht für einen „disjunction-maximum“ Suchalgorithmus, das ist ein Algorithmus, der gleichzeitig über verschiedene Felder sucht, sie getrennt bewertet („disjunction“) und die maximale Bewertung berücksichtigt („maximum“).

Oben wurde ja bereits angemerkt, dass es nicht angeraten ist, eine „Allfields“-Suche durchzuführen, um unerwünschte Suchergebnisse zu vermeiden. Statt dessen können bestimmte Felder für die Suche spezifiziert werden, etwa title und topic, die dann obendrein noch in Hinsicht auf ihre Bewertung gewichtet werden können; title etwa mit dem Faktor 2. Das Ergebnis wäre dann folgendes:

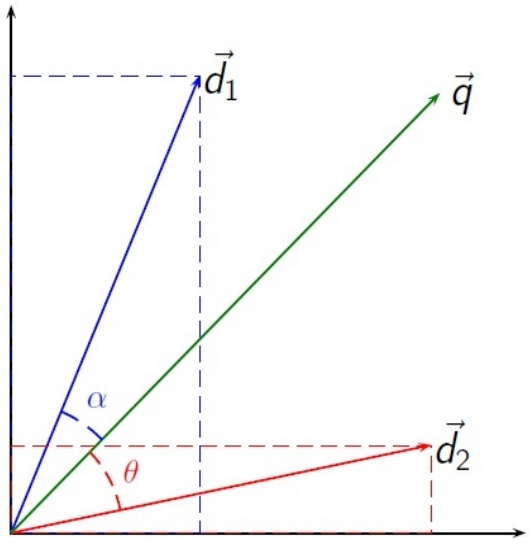
„JavaScript 2.x ist mega-cool“:

```
<str name="parsedquery_toString">
    +(((topic:"java script" | title:"java script"^2.0)
    (topic:"(ii 2 two) x" | title:"(ii 2 two) x"^2.0)
    (topic:ist | title:ist^2.0)
    (topic:"mega cool" | title:"mega cool"^2.0))~4)
</str>
```

Ein Match im Feld ***title*** ist demnach doppelt so hoch gewichtet wie im Feld ***topic***. Doch wie wird ein solcher Treffer überhaupt bewertet?

Der Algorithmus zur Bewertung von Ähnlichkeiten zwischen Suchbegriffen und Feldinhalten basiert auf dem Vektorraummodell für automatische Indexierungen. Über den Ursprung gibt es tatsächlich einige Kontroversen; auf jeden Fall spielen die Arbeiten von Gerard Salton dabei eine tragende Rolle, der 1989 eine umfassende Beschreibung des Algorithmus geliefert hat. Erste Betrach-

tungen dazu fanden bereits in den frühen 1970-er Jahren statt. In diesem Vektorraummodell werden sowohl die Anfrage als auch die Dokumente (bzw. Feldinhalte) als Vektoren dargestellt und mit einer speziellen Metrik ihre Entfernungen voneinander berechnet.



Haben die Anfrage q und die Dokumente d_j die Form q = (q_1,q_2,q_3,...) und d_j = (d_{i1},d_{i2},d_{i3},...), so ist der Kosinus des Winkels zwischen den Vektoren das Maß für die Ähnlichkeit:

$$\|d_i, q\| = \cos(\alpha) = \frac{d_i \cdot q}{|d_i| \cdot |q|} = \frac{\sum_j d_{ij} \cdot q_j}{\sqrt{\sum_j d_{ij}^2} \cdot \sqrt{\sum_j q_j^2}}$$

wobei die euklidische Metrik zu Grunde gelegt wird:

$$|q| = \sqrt{\sum q_j}$$

Die einzelnen Vektoren d_j = (d_{j1},d_{j2},d_{j3},...) werden nach dem sog. „tf-idf-Maß“ (tf steht für Termhäufigkeit und idf für inverse Dokumenthäufigkeit) wie folgt berechnet:

$$d_{ij} = tf_{t,d} \cdot idf(t)$$

wobei $tf_{t,d}$ die Termhäufigkeit des Terms t im Dokument d bezeichnet und idf(t) die inverse Dokumentenhäufigkeit bezüglich des Terms t. Die beiden Größen werden wie folgt berechnet:

$$tf_{t,d} = 0.5 + \frac{0.5 \cdot f_{t,d}}{max\{f_{x,d} \, mit \, x \in d\}} \quad \text{und} \quad idf(t) = \log\left(\frac{|D|}{|\{d' \in D \, mit \, t \in d'\}|}\right)$$

D steht dabei für die Gesamtmenge der Dokumente.

Ein Match wird als umso besser bewertet, je häufiger die Suchterme im Dokument vorkommen, je weniger Terme überhaupt im Dokument vorkommen und je seltener die Suchterme in der Gesamtmenge der Dokumente zu finden sind. Das lässt sich ohne weiteres auf Suchterme in Bezug auf einzelne Dokumentenfelder übertragen. Der Dismax-Algorithmus nimmt dann als Bewertung für das gesamte Dokument die höchste Bewertung für ein einzelnes Feld.

Bei einer solr-Abfrage kann die konkrete Bewertung kann mit Hilfe des Parameters debugquery=on im Detail eingesehen werden. Beispielsweise bei einer Suche nach JavaScript in den Feldern ***title*** und ***topic***, wobei ***title*** doppelt so hoch gewichtet werden soll wie ***topic***:

12.075157 = (MATCH) max of:
4.346499 = (MATCH) weight(topic:"java script" in 248021) [DefaultSimilarity], result of:
4.346499 = score(doc=248021,freq=2.0), product of:
0.5640543 = queryWeight, product of:
21.795341 = idf(), sum of:
10.256164 = idf(docFreq=1442, maxDocs=15106620)
11.539179 = idf(docFreq=399, maxDocs=15106620)
0.02587958 = queryNorm
7.7058167 = fieldWeight in 248021, product of:
1.4142135 = tf(freq=2.0), with freq of:
2.0 = phraseFreq=2.0
21.795341 = idf(), sum of:
10.256164 = idf(docFreq=1442, maxDocs=15106620)
11.539179 = idf(docFreq=399, maxDocs=15106620)
0.25 = fieldNorm(doc=248021)
12.075157 = (MATCH) weight(title:"java script"^2.0 in 248021) [DefaultSimilarity], result of:
12.075157 = fieldWeight in 248021, product of:
1.0 = tf(freq=1.0), with freq of:
1.0 = phraseFreq=1.0
19.320251 = idf(), sum of:
9.268343 = idf(docFreq=3874, maxDocs=15106620)
10.0519085 = idf(docFreq=1769, maxDocs=15106620)
0.625 = fieldNorm(doc=248021)

In Hinblick auf die Suche nach ***JavaScript***, was der Tokenizer zu ***java script*** umformt, und die Gewichtung des title-Feldes hat das am höchsten bewertete Dokument den Wert 12.075157 erhalten. Es ist das Buch „JavaScript“ von Thomas Theis. Ohne dem ***boosting***, der Höherbewertung, um den Faktor 2 hätte ebenfalls der Treffer im title-Feld mit dem Wert 6.0375785 den Gesamtwert des Treffers bestimmt.

In solr/lucene ist - im Unterschied zu den theoretischen Berechnungen oben:

$$tf_{t,d} = \sqrt{f_{t,d}} \quad \text{und} \quad idf(t) = 1 + \log\left(\frac{|D|}{|\{d' \in D \text{ mit } t \in d'\}| + 1}\right)$$

$$queryNorm \approx \frac{1}{\sqrt{\sum_{t \in q} idf(t)^2}} \quad \text{und} \quad fieldNorm \approx \frac{1}{\sqrt{\{f_{x,d} \text{ mit } x \in d\}}}$$

D bezeichnet hier alle Felder eines Typs und d ein spezifisches Feld. Die ***queryNorm*** sollte das Ranking nicht beeinflussen, wohl aber die Werte, die ein Match erhält. Die ***fieldNorm*** sorgt dafür, dass Felder mit vielen Einträgen schlechter bewertet werden als welche mit nur wenig Einträgen. Zu diesen Normierungen kommen noch sog. ***boostings***, Faktoren, die sowohl auf der Anfrage- als auch auf der Dokumentenseite die Werte skalieren. Genau ist dies bei der Klasse search::Similarity der lucene API-Dokumentation beschrieben (siehe die Literaturliste).

In diesem skizzierten Beispiel ist die Einstellung so, dass sämtliche Token (hier java und script) vorkommen müssen. Mit dem Parameter „mm“ („minimum should match“) kann dies flexibel geändert werden, dass beispielsweise mindestens zwei der Token gefunden werden müssen oder 75%. Dieser Parameter wird am Ende der Betrachtungen noch näher beleuchtet.

Erste heuristische Betrachtungen

Die hier skizzierten Berechnungen machen deutlich, dass für eine Optimierung der Relevanzsortierung der Weg, die einzelnen Relevanzberechnungen nachzuvollziehen, nicht gangbar ist. Als Alternativer Ansatz sollen im Weiteren einige heuristische Überlegungen angestellt werden. Zunächst macht es Sinn, die oben begonnenen Betrachtungen zu den für die Relevanzsortierung relevanten Felder und deren Clusterung abzuschließen. Danach folgen ein paar einfache Einzelfallbetrachtungen, um sich den oben beschriebenen Distanzberechnungen zwischen Suchanfragen und Dokumenten qualitativ anzunähern. Schließlich werden die heuristischen Betrachtungen auf Grundlage statistischer Auswertungen vertieft.

Relevante Felder identifizieren und clustern

Nach den oben getätigten Überlegungen können nun die für die Suche und die Relevanzsortierung relevanten Felder identifiziert und in Cluster zusammengefasst werden. Als Kriterien für die Cluster dienen thematische Ähnlichkeiten (wie Titel und vorheriger Titel) und Abhängigkeiten, die sich aus dem marc-index-Mapping ermitteln lassen. Interessant sind im Wesentlichen nur Felder des Typs text (mit stemming) oder des Typs textProper (ohne stemming). Felder vom Typ string eignen sich nur für Phrasensuchen; das sollten typischer Weise Felder sein, in denen eine andere Suche keinen Sinn macht, etwa Format, Veröffentlichungsjahr oder Sprache.

Oben wurde bereits an Hand der Zuordnungen der marc-Daten eine Vorclusterung der Felder vorgenommen. Diese legt es nahe, folgende Cluster von Feldern getrennt zu betrachten: Autorenfelder, Titelfelder, Erschließungsfelder, „volltextähnliche“ Felder und den Rest. Die Feldabdeckungen wurden durch leere Abfragen ermittelt, wobei zu beachten ist, dass die von solr gemeldete Bearbeitungszeit unterhalb des Timeout des Index liegt.

Autorenfelder

Autorenfelder vom Typ text oder textProper gibt es in unserem Fall nur vier:

author	***68% Feldabdeckung***
author2	***38% Feldabdeckung***
author_fuller	***0% Feldabdeckung***
author_additional	***0% Feldabdeckung***

Die beiden Felder ***author*** und ***author2*** sind nach den obigen Betrachtungen voneinander unabhängig. Wie diese beiden Felder zueinander zu gewichten sind, ist in erster Linie eine inhaltliche Frage, wie „wichtig“ weitere Personen oder Institutionen im Vergleich zum Autor zu werten sind.

Titelfelder

Titelfelder vom Typ text oder textProper gibt es deutlich mehr; zunächst diejenigen, in die die marc-Felder 245, 440, 800 und 830 einfließen:

title	***94% Feldabdeckung***
title_auth	***92% Feldabdeckung***
title_short	***92% Feldabdeckung***
title_full	***92% Feldabdeckung***
title_sub	***38% Feldabdeckung***
series	***10% Feldabdeckung***

dann die weiteren, vorangegangenen und folgenden Titel:

title_alt	***10% Feldabdeckung***
title_old	***1% Feldabdeckung***
title_new	***1% Feldabdeckung***

und Titeldaten, die im Wesentlichen bei Artikeln (journal) oder Reihen (series2) eine Rolle spielen:

journal	***35% Feldabdeckung***
series2	***34% Feldabdeckung***

Die Felder in den zwei letzten thematischen Blöcken sind jeweils unabhängig von allen anderen; auch hier sind die Gewichtungen weitgehend inhaltlich zu entscheiden. Die Felder im ersten Block haben die bereits oben skizzierten Abhängigkeiten:

title = series + title_auth
title_full > title_auth > title_sub
title_full > title_short

Insbesondere ***series***, ***title_auth*** (marc 245ab) und ***title_sub*** (marc 245b) sind im Feld ***title*** vollständig aufgehoben, sowie ***title_short*** (marc 245ap) in ***title_full***. Da in Hinblick auf den Titel im marc-Feld 245 nur die Subfelder 245abp relevant sind, stellt sich hier die Frage, ob die Felder ***title, title_sub*** und ***title_full*** aus Redundanzgründen überhaupt in die Bewertungen einbezogen werden sollen. Es bleiben für die Betrachtung demnach noch ***series***, ***title_auth*** und ***title_short*** für die weitere Betrachtung übrig - was den ersten Block der Titeldaten betrifft.

Erschließungsfelder

Als Erschließungsfelder sind zunächst die allgemeinen Erschließungsfelder zu betrachten:

topic	***65% Feldabdeckung***
class	***48% Feldabdeckung***
class_local	***36% Feldabdeckung***
topic_title	***99% Feldabdeckung***
bklname	***14% Feldabdeckung***
geographic	***6% Feldabdeckung***

bklname und ***dewey-xyz*** sind beide vom Typ string und für die Relevanzsortierung nur bedingt geeignet. Die Aufnahme von Feldern vom Typ string in die Relevanzsortierung kann zu ungewünschten Effekten führen: So können diese Felder Suchen blockieren, die Zeichen beinhalten, die für text-Felder als Trennzeichen dienen. Das tun sie dadurch, dass diese Trennzeichen in einem der mit „und“ verknüpften Teilabfragen ausschließlich in den string-Feldern gesucht und dort nirgendwo gefunden werden. In unserem Fall würden wir aber gerne die BK einbeziehen, daher das Feld ***bklname*** mit bewerten. Bei Medien seit 1980 hat es immerhin eine Feldabdeckung von fast 19%, ab 2000 von 21,5% (bei den ganz aktuellen Medien geht die Abdeckung wieder leicht zurück). Mit den dewey-Feldern macht so etwas allerdings keinen Sinn, da sie immer eine Kombination aus Nummer und Begriff enthalten (was als komplette Phrase gefunden werden müsste).

Die Abhängigkeiten der anderen Felder voneinander wurde oben bereits untersucht:

topic_title > topic + title
class > class_local
class > bklname
topic >~ class

Die Abhängigkeiten insbesondere der Felder class, class_local und topic sind in Hinblick auf eine Relevanzbewertung der Dokumente unglücklich gewählt:

marc-Felder	topic	class	class_local	geographic	bklname
050, 060	X	X			
082, 084	X	X			X
651				X	
600, 610, 630, 650, 689	X				
982	X				
983		X	X		

Die lokalen Schlagwort- und Systematikfelder (marc 982 und 983) lassen sich so nicht von den globalen Feldern trennen. ***topic_title*** ist weitgehend redundant, vor allen Dingen in Hinblick darauf, dass zusätzlich zu den in ***title*** und ***topic*** erfassten Feldern keine für die Suche relevanten Felder hier aufgenommen werden. Ebenso das Feld ***class_local***.

volltextähnliche Felder

Zu den volltextähnlichen Felder im Index können folgende Felder gezählt werden:

contents	***1% Feldabdeckung***
abstract	***29% Feldabdeckung***
fulltext	***6% Feldabdeckung***
allfields	***100% Feldabdeckung***

Die Felder ***contents*** und ***abstract*** sind von allen anderen unabhängig. ***fulltext*** ist ein - in Hinblick auf die marc-Daten - virtuelles Feld; hier werden die in den Daten gefundenen Links aufgelöst und nach volltextähnlichen Textteilen durchsucht; oft finden sich hier die Inhalte von Inhaltsverzeichnissen, Abstracts oder Buchbesprechungen wieder. ***allfields*** ist ein Feld, in das alle im marc vorhandenen Daten kopiert werden.

weitere Felder

Weitere Felder, die vielleicht für spezielle Suchen, nicht aber für die Gewichtung einer allgemeinen Suche geeignet sind:

id	***100% Feldabdeckung***
format	***100% Feldabdeckung***
publishDate	***99% Feldabdeckung***
ctrlnum	***84% Feldabdeckung***
signature	***69% Feldabdeckung***
publishPlace	***57% Feldabdeckung***
normlink	***54% Feldabdeckung***
publisher	***49% Feldabdeckung***
isbn	***21% Feldabdeckung***
issn	***20% Feldabdeckung***
dewey-xyz	***17% Feldabdeckung***
genre	***< 1% Feldabdeckung***

Dazu kommen noch Felder, die von den oben genannten abhängig sind und für die Suchen irrelevante zusätzliche Informationen enthalten, und andere Felder, die nicht weiter betrachtet werden müssen. Die Suchen nach ***id*** (ppn), ***normlink*** (Normdaten-Ids), ***ctrlnum*** (enthält u.a. die ZDB-Id), ***isbn***, ***issn***, ***signature***, ***publisher*** und ***publishPlace*** können gut auch als Spezialsuchen implementiert sein; ***dewey-xyz***, ***bklname***, ***publishDate***, ***genre*** und ***format*** eignen sich eventuell als Facetten. Da ids in ctrlnum, ISBN und ISSN aus sehr speziellen Zeichenketten bestehen und es denkbar wäre, dass über die allgemeine Suche auch nach Verlagen oder Verlagsorten gesucht wird, können diese Felder mit einer entsprechend niedrigen Skalierung in die Relevanzbewertung eingehen; konkret mit der Bewertung 0. Ansonsten würden Einträge in diesen Feldern nicht gefunden werden. Von der Einbeziehung der Signaturen haben wir allerdings abgesehen, da manche sprechende Signaturen zu unerwarteten Ergebnissen führen können.

Auf die inneren Werte kommt es an

Damit sind hier vier Cluster von Indexfeldern identifiziert, die sowohl intern als auch zueinander gewichtet werden müssen; der Titelcluster wird auf zwei aufgeteilt, weil er viele Felder enthält, einen mit Haupt- und einen mit Nebentitelangaben:

Autorencluster:	***author, author2***
Titelcluster (1):	***series, title_auth, title_short, series2, journal***
Titelcluster (2):	***title_alt, title_old, title_new***
Erschließungscluster:	***topic, class, (class_local), geographic, dewey-xyz, bklname***
Volltextcluster:	***contents, abstract, fulltext, allfields***
gering bewertete Felder:	***isbn, issn, publisher, publishPlace, ctrlnum***

author2, ***series***, ***topic*** und ***class*** sind Felder, die viele marc-Daten beinhalten, sodass es sich hier lohnt genauer nachzusehen, was diese Felder genau beinhalten:

author2: *Besteht aus den Feldern **110ab**, **111ab**, 7**00abcd**, **710ab**, **710ab**.*
*Da die Felder Personendaten (**700**), Institutionsdaten (**110**, **710**) und Kongressdaten (**111**, **711**) beinhalten., wird es hier selten Überlappungen von Einträgen geben.*
Zu beachten ist hier, dass Haupteinträge (1xx) und weitere Einträge (7xx) ins selbe Indexfeld übertragen werden.

series: *Besteht aus den Feldern **440ap**, **800abcdfpqt** und **830ap.***
*Im Feld **800** sind lediglich die Subfelder **a, q, p und t** für die Suchen relevant. Ist in **440a** etwas eingetragen, so soll in den **8xx**-Feldern nichts eingetragen sein.*
*Als Personen- und Titelfelder enthalten auch **800** und **830** voneinander verschiedene Einträge. Zu beachten ist hier, dass insbesondere das Feld **800** nicht nur Titel- sondern auch Personendaten enthalten kann.*
Es besteht in sofern eine Abhängigkeit zum Feld series2, dass ein Eintrag in 490a einen Eintrag in 830a voraussetzt; oft findet sich der Eintrag in 830a in 490a wieder.

topic: *Besteht aus den Feldern **050**, **060**, **082**, **084**, **600**, **610**, **630**, **650**, **689** und **983**.*
*Die Felder **050** und **060** sind für die Suchen nicht relevant, da sie lediglich Bezeichner beinhalten, ebenso das Feld **082** für die Dewey-Klassifikation.*

*Das Feld **084** für andere Klassifikationen kann auch Klartext enthalten, in unserem Beispiel regelhaft dann, wenn es eine BK enthält.*
*Die **6xx**-Felder sind für die (global vergebenen) Schlagwörter: **600** Personen, **610** Institutionen, **630** „klassische" Schlagwörter, **650** „kanonische" Schlagwörter. Diese Felder enthalten erwartungsgemäß verschiedene Einträge.*
*Das Feld **689** enthält Erläuterungen zu vergebenen Schlagwörtern, die sich darin in der Regel doppeln. **Überhaupt sind Mehrfachnennungen von Schlagwörtern in den 6xx-Feldern nicht selten**, beispielsweise wenn ein Schlagwort durch verschiedene Institutionen - etwa des Bibliothekenverbunds oder der Nationalbibliothek - qualifiziert ist. Diese Mehrfachnennungen werden auch in den Index übernommen.*
*Das Feld **982** schließlich enthält lokal vergebene Schlagwörter. Da am hiesigen Verbund sehr viele Bibliotheken beteiligt sind, sind hier auch viele Schlagwörter zu finden, nicht selten auch welche, die eigentlich keine hohe globale Relevanz haben. **Dieser Umstand wertet das topic-Feld insgesamt leider ab**.*

class: *Enthält nach den obigen Betrachtungen lediglich die Felder **084** und **983** als „sprechende" Felder.*
Die Einträge in 983 sind wiederum lokal und teilweise recht umfangreich, was auch hier zu einer Abwertung des Feldes insgesamt führt.

Weiterhin sollte beachtet werden, dass es Felder gibt, die bevorzugt in Zusammenhang mit bestimmten Medientypentypen auftreten. In dem hier dargelegten Fall sind es die Felder ***series*** und, ***series2*** (Zeitschriftenbände, bzw. Bände in Reihen oder Serien), sowie ***journal*** (Artikel). Die beiden speziellen Felder ***fulltext*** und ***allfields*** beinhalten viele unkontrollierte Daten; in der Praxis hat sich erwiesen, dass die Berücksichtigung von ***allfields*** zu unerwünschten Effekten führt. Das ***fulltext***-Feld kann dagegen mit einer geringen Bewertung verwendet werden; meist enthält es allerdings Daten aus den Feldern ***abstract*** und ***contents***. sodass wir entschieden haben, sie für die Relevanzsortierung nicht zu berücksichtigen. In diesen Feldern sind manchmal auch Verzeichnisse, Abstracts oder Besprechungen von mehreren Werken enthalten. Dadurch konnte es in beluga passieren, dass die Suche nach „erotische Literatur Lateinamerika" auch „Tim und Struppi in Tibet" fand. Speziell zu beachten ist hier auch, dass die Einträge in den ***dewey-xyz*** Feldern englischsprachig sind und jeweils sowohl die Nummern als auch den Text beinhalten.

Um die Parameter für eine Relevanzsortierung geeignet zu setzen bedarf es mehrerer Entscheidungen:

Welche Felder sollen bei der Suche überhaupt berücksichtigt werden?
Wie sollen diese Felder gewichtet werden?
Welche globalen Bewertungsparameter sollen gesetzt werden?

Die zu berücksichtigenden Felder wurden bereits identifiziert und zu geeigneten Clustern zusammengefasst. Für die Gewichtung der Cluster zueinander und der Felder jeweils innerhalb der Cluster sollten die bislang gemachten Überlegungen in Betracht gezogen werden:

Abhängigkeiten zwischen Feldern
Feldabdeckungen
Qualität der jeweiligen Feldinhalte

Auf dieser Grundlage ergeben sich erste Ideen, wie die Cluster und die Felder in den Clustern zueinander bewertet werden sollten. In dem hier dargestellten Beispiel der Hamburger Bibliotheken in beluga sähen diese Ideen wie folgt aus:

Bewertungen innerhalb der Cluster

Autorencluster: - ***author*** sollte höher bewertet sein als ***author2***.

Titelcluster (1): - ***title_auth*** und ***title_short*** sollten höher bewertet sein als ***series*** und ***series2***, aber nicht zu sehr, da sie sich überlappen.
- ***series*** und ***series2*** sollten höher bewertet sein als ***journal*** (da eher die Zeitschriften gefunden werden sollen, als enthaltene Artikel).

Titelcluster (2): - ***series***, ***title_auth*** und ***title_short*** sollten höher bewertet sein als ***title_alt***, ***title_old***, ***title_new***.

Erschließungscluster:- ***class*** sollte höher bewertet werden als ***topic***.
- ***class_local*** könnte zur Abwertung lokaler Systematikvergaben verwendet werden.
- ***geographic*** sollte wegen der geringen Feldabdeckung nicht zu hoch bewertet werden.
- ***bklname*** sollte wegen der geringen Feldabdeckung ebenfalls nicht zu hoch bewertet werden.

Volltextcluster: - ***contents*** und ***abstract*** können ähnlich hoch bewertet werden.
- ***fulltext*** wird nur gering bewertet
- ***allfields*** wird nicht verwendet.

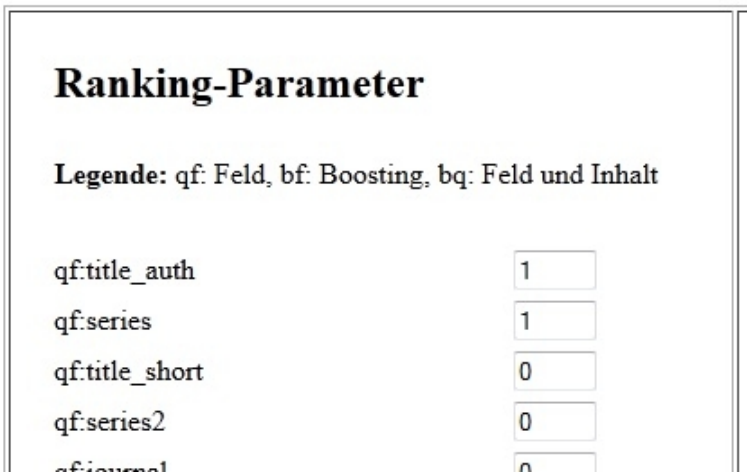
Bewertungen der Cluster zueinander

Wird nach Namen gesucht, sollten diese im ***Autorencluster*** höher bewertet sein als im ***Titel***- und im ***Erschließungscluster***. Ansonsten sind Treffer im ***Titelcluster*** höher zu bewerten als im ***Erschließungscluster*** (Qualität der Daten) und in diesen beiden Clustern höher als in den ***Volltextclustern*** (Kontrolle über die Inhalte). Die ***gering zu bewertenden Felder*** sollten deutlich geringer bewertet werden als die anderen Felder, am besten mit 0. Es ergibt sich - in der Tendenz - folgendes Bild:

Autorencluster = Titelcluster > Erschließungscluster > Volltextcluster > Rest = 0

Einzelfallbetrachtungen

Die Berechnungen der einzelnen Relevanzwerte ist relativ komplex und insbesondere auf größeren Datenstrukturen kaum mehr händisch nachzuvollziehen. Hier macht es Sinn, zunächst Suchen mit wenig Ergebnissen zu betrachten, um das Verhalten der Bewertungsalgorithmen besser einschätzen zu können. Um diese möglichst bequem zu gestalten, verwenden wir eine kleine Webanwendung, die entsprechend parametrisierte Suchanfragen absetzt und die Auswertung der Ergebnisse anzeigt. Unten sind die Eingabemaske für die Parameter und die Einzelabfragen zu sehen; die Eingabemaske für die Parameter setzt sich nach unten fort:



Und hier ein Ausschnitt des Ergebnisses:

Solr-Abfrage

Rang:1 PPN:330582771 [Kapsel: Alsterdampfer 1,2] (Book,)

Rang:2 PPN:050351249 "St. Georg" neuer alter Alsterdampfer (Article, 1994)

```
6.7802777 = (MATCH) sum of:
  6.7802777 = (MATCH) max of:
    6.7802777 = (MATCH) weight(title_auth:alsterdampf in 441246) [DefaultSimilarity], result of:
      6.7802777 = score(doc=441246,freq=1.0), product of:
        0.9376728 = queryWeight, product of:
          16.527916 = idf(docFreq=2, maxDocs=16626823)
          0.05673267 = queryNorm
        7.230963 = fieldWeight in 441246, product of:
          1.0 = tf(freq=1.0), with freq of:
            1.0 = termFreq=1.0
          16.527916 = idf(docFreq=2, maxDocs=16626823)
          0.4375 = fieldNorm(doc=441246)
```

Rang:3 PPN:769621627 Bildband die Alsterdampfer in Hamburg (Book, 2013)

Rang:4 PPN:050542982 Der älteste Alsterdampfer kehrte zurück (Article, 1994)

Rang:5 PPN:504446479 Alsterdampfer auch nach Eilbek? Zur Diskussion gestellt (Article, 2004)

Rang:6 PPN:622429353 Mit dem Alsterdampfer nach Winterhude-Eppendorf (Article, 1968)

Rang:7 PPN:050573284 Alsterdampfer "St. Georg" zurück auf die Alster (Article, 1995)

In dem gerade gezeigten Beispiel wird ***series*** gegen ***title_auth*** getestet, beide mit dem gleichen Gewicht 1. Es fällt hier auf, dass die ersten 12 Ränge ihre Bewertung durch ***series*** erhalten haben, während die Einträge in ***title_auth*** erst ab Rang 13 zum Tragen kommen. Wird die Gewichtung von ***title_auth*** um nur 0,1 auf 1,1 erhöht, so ändert sich das Bild hier radikal: Die ersten 17 Ränge werden dann durch die Bewertung von ***title_auth*** vergeben, während der Rang 18 der erste ist, der durch den Wert von ***series*** erzielt wurde. Wird aber das Gewicht von ***series*** um 0,1 auf 0,9 verringert, so sind die ersten 10 Ränge immer noch durch ihren Wert im Feld ***series*** bestimmt, währen ***title_auth*** erst ab dem 11. Rang zum Tragen kommt. Bei einem Wert von 0,8 für ***series*** sind wiederum die ersten 16 Ränge durch ***title_auth*** bestimmt.

Etwa bei einem Verhältnis von 1:1,165 (***series***:***title_auth***), scheinen die Gewichtungen dieser beiden Felder zueinander ausgewogen zu sein. Es zeigt sich hier, dass im direkten Vergleich der Feldbe-

wertungen, die Ergebnisse in Hinblick auf das damit verbundene Ranking sich keineswegs linear verhalten. Sie ändern sich vielmehr an bestimmten Punkten schlagartig. Diese „Schwellwerte“ sagen zum einen etwas über die Felder selbst aus, sind aber auch von den gewählten Suchbegriffen abhängig. Obige Tabelle zeigt die Werte für den Suchbegriff Java. Für den Suchbegriff Hamburg liegt der Schwellwert dagegen zwischen 1:1,55 und 1:1,6.

Gewicht (title_auth)	**Gewicht (series)**	**Rang (title_auth)**	**Rang (series)**
1	1	> 12	1 - 12
1,1	1	1 - 17	> 17
1	0,9	> 10	1 - 10
1	0,8	1 - 16	> 16
1,165	1	2, 3, 4, 6, 7, ...	1, 5, 8, 9, 10, ...

Durch die Abhängigkeit der Ergebnisse von den Suchbegriffen, ist die Aussagekraft solcher Einzelfallbetrachtungen nur begrenzt. Sie sind aber nützlich, um eine sinnvolle Ausgangskonstellation von Gewichtungen zu ermitteln. Dabei ist es sinnvoll, zunächst die Schwellwerte im Vergleich von unterschiedlichen Feldern eines Clusters zueinander einzugrenzen, um dann die Cluster als Ganzes zueinander adäquat zu gewichten. Hier eine beispielhafte Auswahl von (grob) ermittelten Schwellwerten zum Suchwort Java:

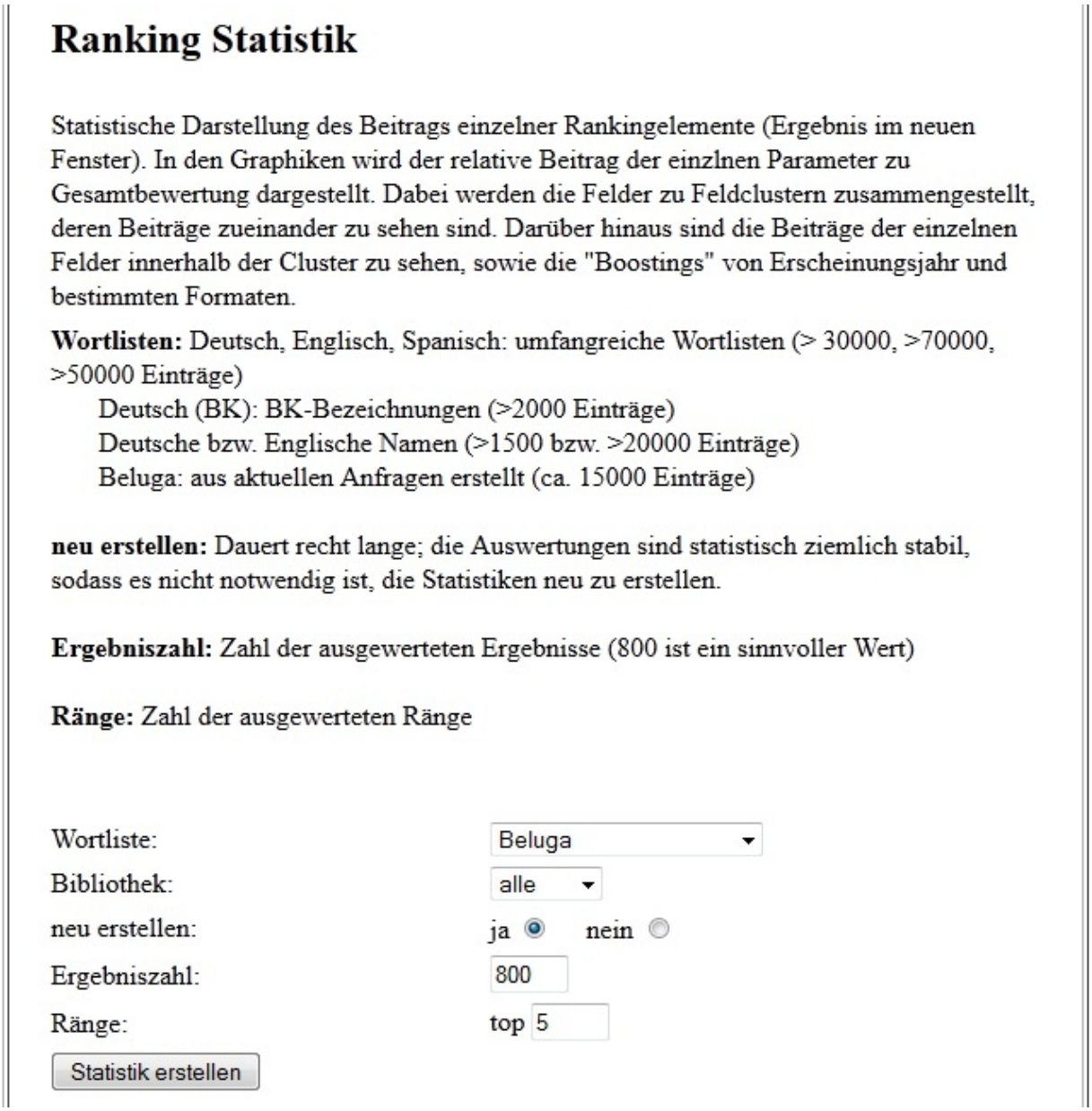
Feld 1	**Feld 2**	**Schwellwert**
series	title_auth	1 : 1,3
series	title_short	1 : 1,3
series2	series	1 : 1,4
title_alt	title_short	1 : 1,7
author	author2	1 : 1,0
author	title_short	1 : 1,4
title_short	topic	1 : 1,6
topic	class	1 : 1,5
title_short	abstract	1 : 1,8

Diese Schwellwerte spiegeln zum einen Gegebenheiten im Datenbestand der einzelnen Felder wider, wie viele Wörter sie im Durchschnitt enthalten und wie oft sie belegt sind, dies aber immer nur relativ zum Suchbegriff. Daher müssen auf diese Weise die Schwellwerte für verschiedene Suchbegriffe ermittelt werden; aber bei annähernd 13 Millionen Datensätzen ist die Auswahl einer geeigneten und dennoch kleinen Stichprobe kaum möglich. So ergeben sich mit diesen Betrachtungen vielleicht erste Anhaltspunkte. Hauptsächlich aber zeigen sie das qualitative Verhalten der Relevanzbewertung auf: Dass die Bedeutung einzelner Feldeinträge bei sich verändernden Gewichtung schlagartig zu- oder abnimmt. Das geht im Wesentlichen auf die Berechnung der fieldnorm zurück, die als Faktor in die Einzelbewertung eingeht, aber einen sprunghaften Verlauf hat. Die fieldnorm und die idf sind diejenigen Parameter, die Index-seitig die Schwellwerte bestimmen.

Heuristische Betrachtung mit statistischen Werkzeugen

Anstatt einzelne Suchabfragen zu betrachten, deren Ergebnisse den Nachteil haben, von den Suchen abzuhängen, gibt es auch die Möglichkeit viele Suchabfragen statistisch auszuwerten und die Ergebnisse zu betrachten. Bereits mit wenigen hundert Suchabfragen als Stichproben können so verlässliche Werte ermittelt werden, vorausgesetzt, die Suchabfragen ähneln im statistischen Mittel den Abfragen im „wirklichen Leben“. Ähnliches muss dabei auch für die ausgewerteten Ergebnisse, hier die ersten 20 Treffer, gelten. Die Suchabfragen können zum einen aus einem Pool spezieller Abfragen generiert werden, um spezifische Bewertungen zu analysieren, beispielsweise Namen für die Analyse von Autorensuchen, oder kontrollierte Vokabulare für die Analyse von Stichwortsuchen. Für das Allgemeine sind hier aber diejenigen Suchabfragen die besten, die dem wirklichen Katalogbetrieb entstammen.

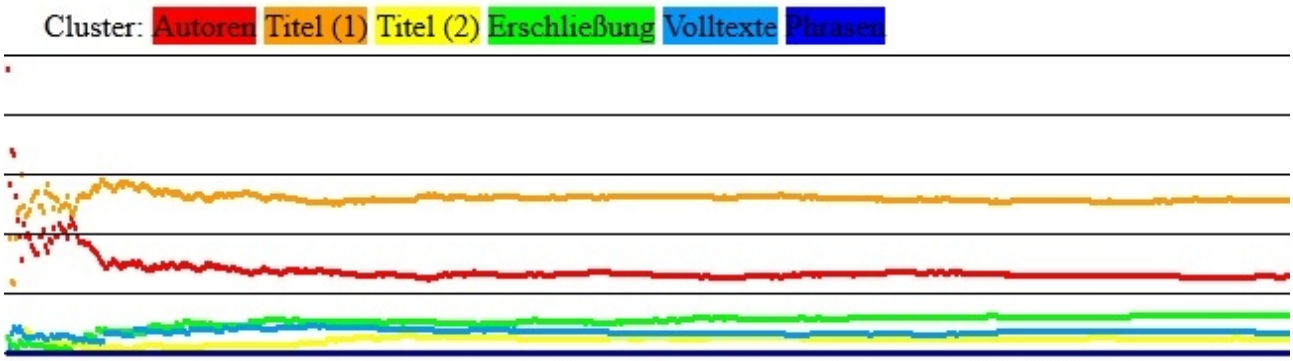
Um sich nicht in eine unübersichtliche Menge von Berechnungen zu verlieren, erscheint es sinnvoll, die statistischen Auswertungen graphisch darzustellen und die Analysen auf diese Darstellungen zu beziehen. Dafür benutzen wir ein einfaches, browsergestütztes Werkzeug:



Mit diesem Werkzeug lässt sich eine Wortliste auswählen, aus der zufällig eine Anzahl von Suchanfragen generiert werden. Dabei kann gewählt werden, wie viele Ergebnisse insgesamt ausgewertet werden sollen (hier 800) und welche Ränge (hier die ersten 5); Die Zahl der Ergebnisse entspricht dabei der Zahl der ausgewerteten Ränge multipliziert mit der Zahl der Suchabfragen (in diesem Fall werden also 160 Abfragen abgesetzt). Die Suche lässt sich auch auf einzelne Bibliotheken einschränken.

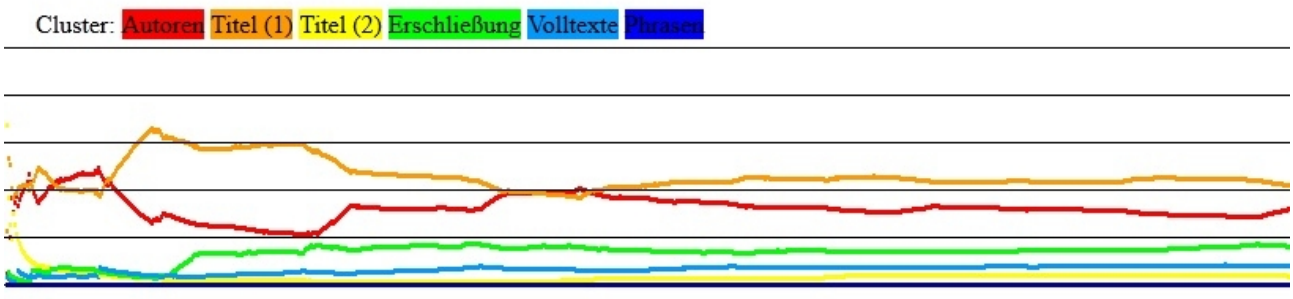
Die Ergebnisse werden in Form von Verlaufsdiagrammen angezeigt, in denen von links nach rechts für jeden angezeigten Aspekt die über die Suchanfragen kumulierten Werte dargestellt werden. Nach den Gesetzen der Statistik streben diese kumulierten Werte auf einen Mittelwert zu, der - mit einer gewissen Wahrscheinlichkeit - dem (hypothetischen) Mittelwert der durchschnittlichen Bewertung der Gesamtheit der Dokumente bei der Gesamtheit aller Suchabfragen entspricht. Bei einer Stichprobe von etwa 15000 Dokumenten und 800 ausgewerteten Ergebnissen ist am Ende (d.h. am rechten Rand des Diagramms) ein stabiler Wert zu erwarten. Im Folgenden eine Demonstration mit Ranking-Parametern, die den am Ende des hier dargestellten Prozesses ermittelten Parametern ähneln.

Zunächst wird nur der erste Rang ausgewertet:

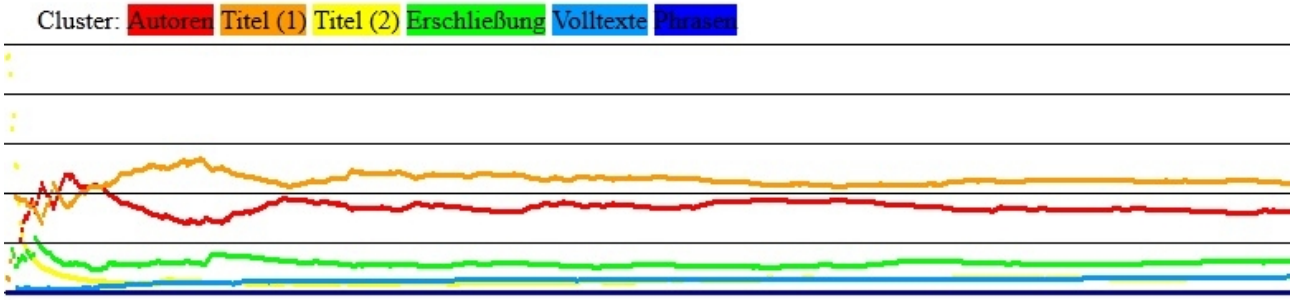


Hier ist die Ansicht für die Cluster zu sehen: Jeder Cluster hat eine eigene Farbe. Die unterste Linie ist die 0-Punkte-Linie, während die oberste die Summe aller Bewertungspunkte darstellt. Zusätzlich zu den Clustern wird hier (in dunkelblau) der Anteil der Werte dargestellt, der durch eine gesonderte Bewertung von Phrasentreffern erzielt wird. Hier ist er 0, weil Phrasen noch nicht gesondert bewertet werden; das kommt erst später. Zu sehen ist gut, wie die Linien, die anfangs (links) noch ziemlich schwanken, sich zum Ende (rechts) hin zunehmend begradigen. Zusätzlich zur Darstellung der Übersicht über die Cluster gibt es eine Reihe anderer Darstellungen, etwa der einzelnen Cluster und auch weitere Parameter, die weiter unten nach und nach erläutert werden.

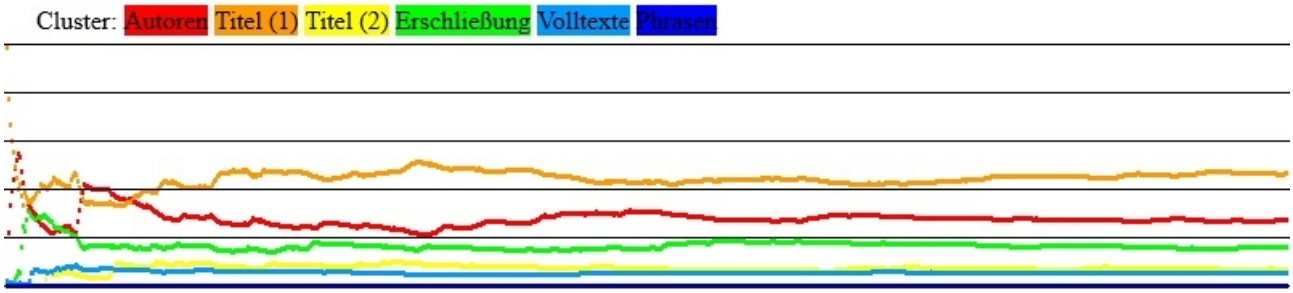
Die Auswertung der jeweils ersten zwanzig Ränge gibt ein etwas anderes Bild, da hier nur 40 Suchabfragen ausgewertet wurden:



Hier sind die Schwankungen auch nach rechts hin größer, die Werte, auf die die Linien zustreben, qualitativ aber sehr ähnlich. Für die weiteren Betrachtungen werden - als guter Kompromiss - immer die ersten fünf Ergebnisse ausgewertet. Das ergibt dann folgendes Bild:

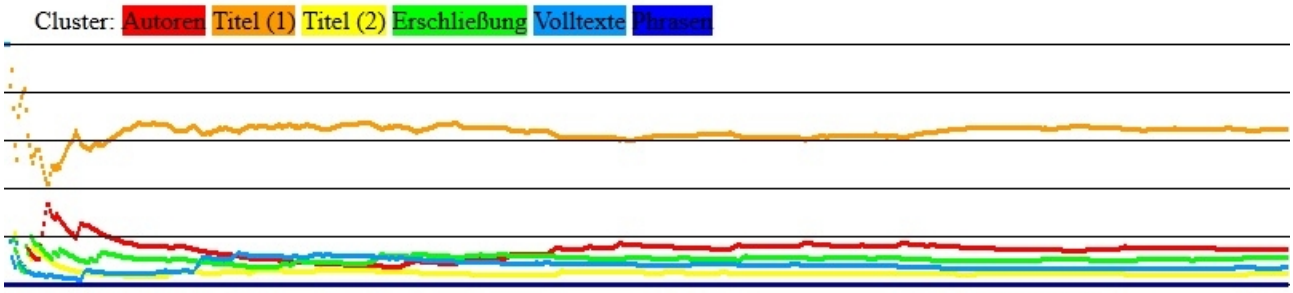


Zum Vergleich dieselbe Auswertung auf derselben Grundmenge. Da die Suchabfragen zufällig aus der Grundgesamtheit ermittelt werden, ist hier ein etwas anderes Bild zu erwarten:



Die Unterschiede sind im Wesentlichen in der linken Hälfte zu sehen, während die Werte, auf die die Diagrammlinien zustreben, sehr ähnlich sind; ähnlich genug jedenfalls für eine heuristische Betrachtung.

Unterschiede sind aber zu erkennen, wenn für die Abfragen eine andere Wortliste als Grundlage gewählt wird. Hier das Ergebnis über einer Liste mit etwa 30000 deutschen Wörtern.



Die Reihenfolge der Bewertungen ist zwar dieselbe wie oben, aber die Abstände der Bewertungen unterscheiden sich deutlich: Hier wird der erste Titelcluster deutlich aufgewertet, der Autorencluster dagegen abgewertet. Das ist nicht weiter verwunderlich, da dieser Wortliste aus gebräuchlichen Wörtern besteht und daher erwartungsgemäß weniger potenzielle Namen enthält. Ein Auszug aus dieser Wortliste sieht wie folgt aus:

Entflut
Entfluten
Entflügel
Entflügeln
Entform

Entformen
Entformung
Entformungsschräge

Es sind einzelne Wörter (keine Phrasen), die häufig auch in mehreren Formen vorkommen.

Die in diesem Artikel verwendete Wortliste enthält etwa 15000 verschiedene Suchanfragen, die in der Zeit vom 24.5. bis zum 2.6. 2015 bei beluga im Livebetrieb abgesetzt wurden. Für diese 15000 Abfragen werden die Logdateien von etwa 6 bis 8 Tagen ausgewertet. Hier ein kurzer Auszug aus der beluga-Wortliste:

organon der
Artistic Citizenship A Public Voice for the Arts.
Vegetationsgeographie
autobiographie
trauma und Kindheit
organon der heilkunst
Molekülphysik und Quantenchemie Einführung in die experimentellen und theoretischen Grundlagen
Marketing Event

Darin kommen aber auch Ausdrücke wie im Folgenden vor:

"Energiebewusstes Bauen"
fo fa gai lun
A1997/7913
10754:63
e-commerce* einzelandel*

Die Abfragen, wie sie „real“ gestellt werden sind also wesentlich heterogener als die Einträge irgendwelcher Wortlisten. Deswegen ist es sinnvoll, solche „realen“ Listen von Suchbegriffen als Grundlage für die statistischen Auswertungen zu nehmen. Dennoch sind die aus den Bewertungen erstellten Diagramme in Hinblick auf die Suchabfragen nachvollziehbar stabil und lassen damit auch eine Beurteilung der relativen Bewertungen nach Augenschein zu. Die rechten Enden der Kurven zeigen dabei jeweils die Werte an, auf die die Bewertung im statistischen Mittel zustrebt, die Kurvenverläufe geben einen Aufschluss über die zu erwartende Streuung der Werte.

Arbeiten mit statistischen Betrachtungen

Bevor mit der statistischen Auswertung der Relevanzgewichtungen begonnen werden kann, muss zum einen eine Startkonfiguration erstellt und zum anderen die Clusterung der Felder definiert werden. Beides ist in den Vorüberlegungen schon weitgehend geschehen. Hier die Tabelle mit den Startparametern (siehe dazu auch die Tabelle mit den Schwellwerten):

Titelcluster (1):

title_auth = title_short > series = series2 > journal

title_auth	***20***
title_short	***20***
series	***15***
series2	***10***
journal	***5***

Titelcluster (2): < Titelcluster (1)

series > title_alt = title_new = title_old

title_alt	*5*
title_new	*5*
title_old	*5*

Autorencluster: = Titelcluster (1)

author > author2; title_short ~ 1,4 x author

author	*15*
author2	*10*

Erschließungscluster: < Titelcluster (1)

class > topic > geographic ~ bklname; topic ~ 1,4 x title_short

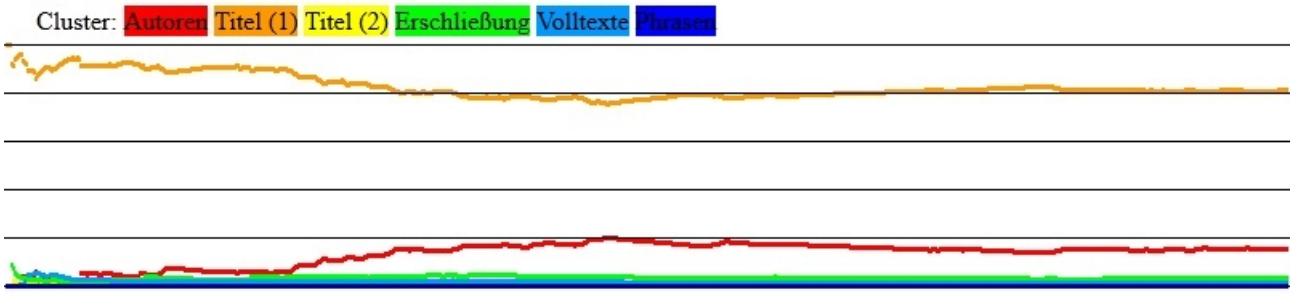
class	*20*
topic	*10*
geographic	*5*
bklname	*5*

Volltextcluster: < Erschließungscluster

abstract = contents > fulltext; abstract ~ 1,8 x title_short

abstract	*10*
contents	*10*
fulltext	*5*

Hier die dazu gehörige Cluster-Ansicht:



Zu sehen ist hier, dass im Vergleich der Cluster der Titelcluster deutlich überrepräsentiert und die Erschließungs- und Volltextcluster deutlich unterrepräsentiert sind. Daher werden im nächsten Versuch der Titelcluster etwas ab- und die anderen etwas aufgewertet; dabei erfährt der Erschließungscluster eine etwas höhere Aufwertung als die anderen:

Titelcluster (1):

title_auth	*18*
title_short	*18*
series	*13*
series2	*10*
journal	*5*

Titelcluster (2):

title_alt	*5*
title_new	*5*
title_old	*5*

Autorencluster:

author	***17***
author2	***12***

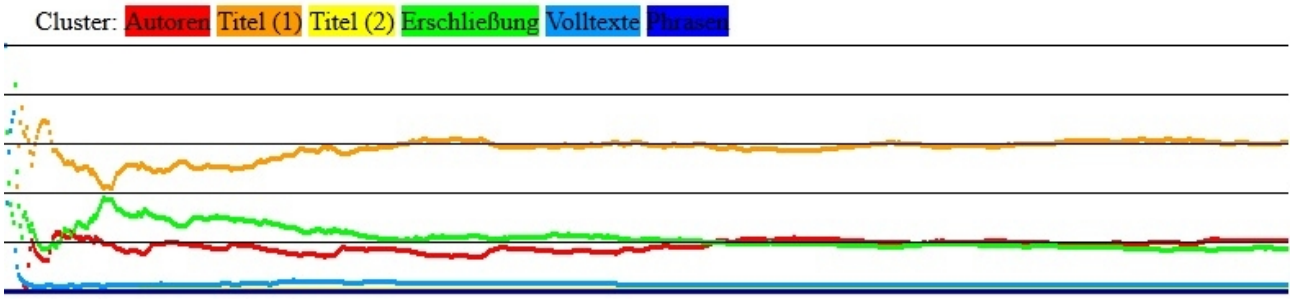
Erschließungscluster:

class	***30***
topic	***20***
geographic	***15***
bklname	***15***

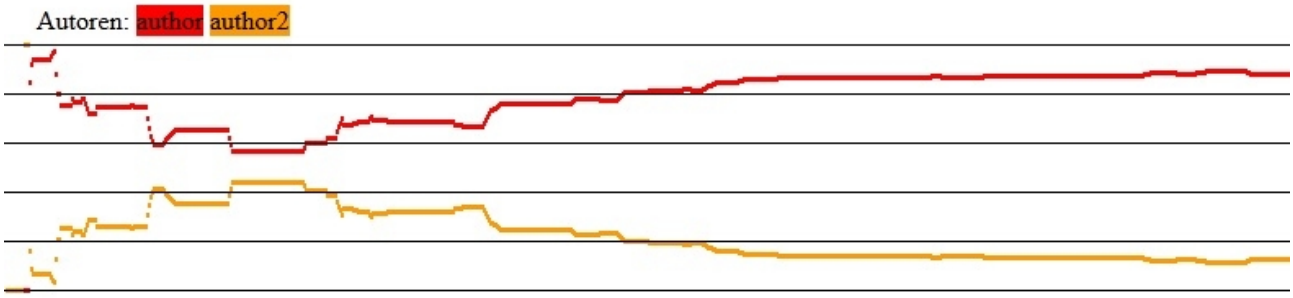
Volltextcluster:

abstract	***15***
contents	***15***
fulltext	***10***

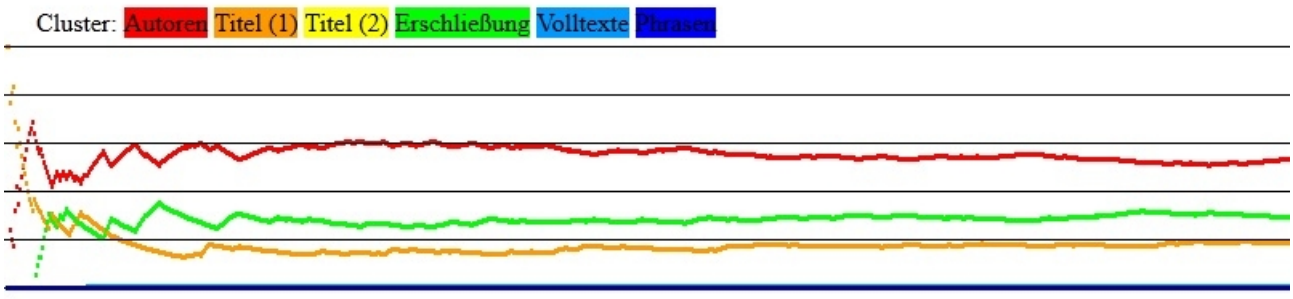
Die Cluster sehen dann im Vergleich zu einander so aus:



Zu sehen ist jetzt, dass die Gewichtung der Cluster zueinander bereits eher den Zielvorstellungen entspricht. Der Autorencluster könnte gegenüber dem Erschließungscluster etwas aufgewertet werden, aber dafür gibt es weiter unten noch etwas tiefer gehende Betrachtungen. Die Volltexte und die Nebentitelangaben sind allerdings deutlich unterrepräsentiert. Vor einer weiteren Anpassung der Parameter lohnt sich aber ein Blick in die einzelnen Cluster. Zunächst die Autoren:

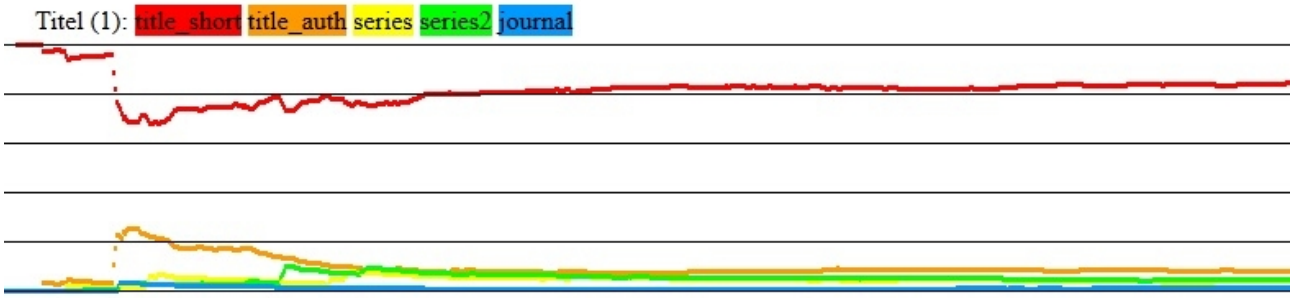


Hier scheint das Verhältnis von Haupt- und Nebenangaben passend zu sein. Um die Justierung der Parameter besser einschätzen zu können, kann auch die Auswertung über eine Liste von etwa 1500 deutschen Nachnamen betrachtet werden; zunächst die Cluster im Vergleich:



Hier ist der Autorencluster wie erwartet maßgeblich für die Bewertung der Dokumente. Die interne Bewertungsverteilung im Autorencluster sieht wie folgt aus:

Hier ist zu sehen, dass ***author2*** im Vergleich zu ***author*** fast verschwindet und daher etwas aufgewertet werden könnte. Die beiden Titelcluster zeigen sich so:

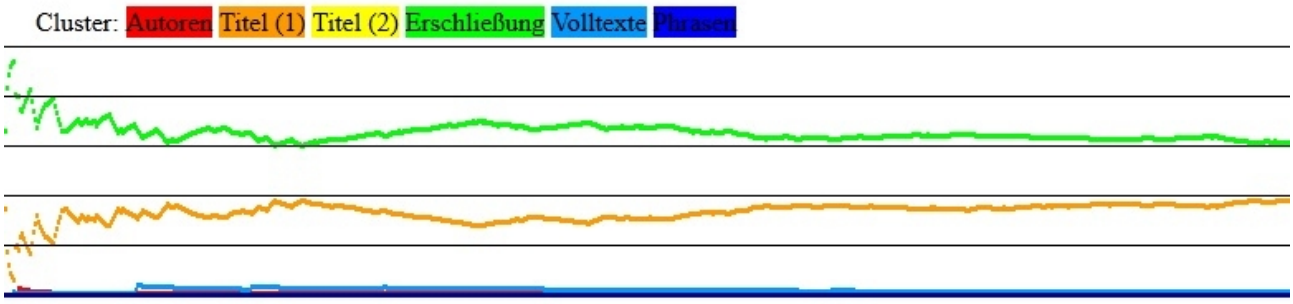


und

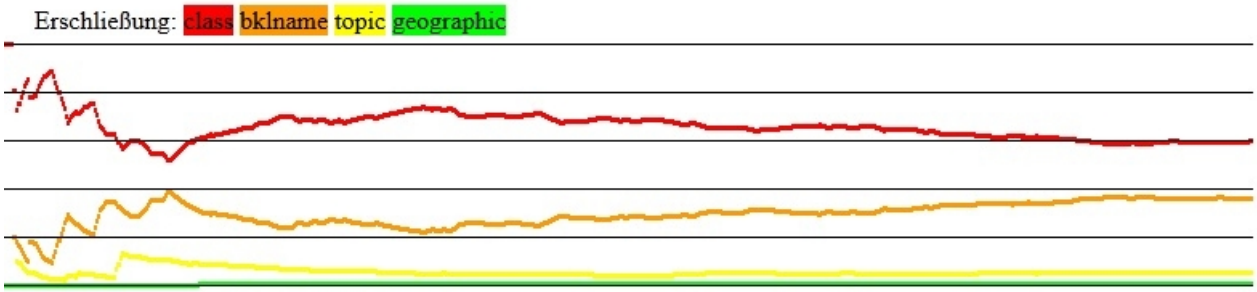
Hier erscheint ***title_short*** im Vergleich zu den anderen Feldern zu hoch bewertet, die Verhältnisse von ***title_alt*** zu ***title_old*** bzw. ***title_new*** zueinander auf Grund ihrer jeweiligen Feldabdeckungen plausibel. Nun der Erschließungscluster:

Das sieht insgesamt schon ganz brauchbar aus; vielleicht könnte das ***topic***-Feld gegenüber dem ***class***-Feld ein wenig zurücktreten. Um hier wie schon bei den Autoren eine weitere Perspektive ein-

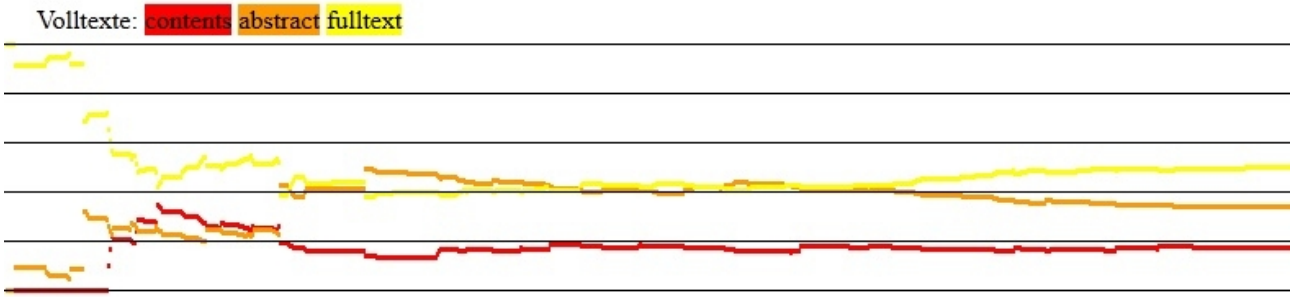
zubeziehen, kann als Grundlage der Suchausdrücke etwa die Liste der Basisklassifikationen gewählt werden. Bei den Clustern hat dann der Erschließungscluster sozusagen das Sagen:



Erwartungsgemäß spielt der Autorencluster hier keine Rolle mehr. Intern zeigt sich der Erschließungscluster wie folgt:



Hier ist zu sehen, dass bei vorliegen eines Treffers der BK, das Feld ***class*** besser bewertet wird als das Feld ***bklname***; das könnte eine Korrektur vertragen. Am Ende kommt jetzt noch ein Blick auf den Volltextcluster:



Das sieht ziemlich ausgewogen aus. Da aber dass ***fulltext***-Feld etwas suspekt ist, weil es Daten enthält, die in den Metadaten nicht explizit vorkommen, könnte dies noch ein wenig abgewertet werden.

Diese Justierungen der Parameter lassen sich auf diese Weise sukzessive den eigenen Anforderungen anpassen. Bereits nach ein paar wenigen Runden ist eine Verteilung der Gewichte gefunden, die als ein erstes Zwischenergebnis betrachtet werden kann. Sie sieht wie folgt aus:

Titelcluster (1):

title_auth	***19***
title_short	***17***
series	***15***
series2	***12***
journal	***10***

Titelcluster (2):

title_alt	***10***
title_new	***10***
title_old	***10***

Autorencluster:

author	***19***
author2	***17***

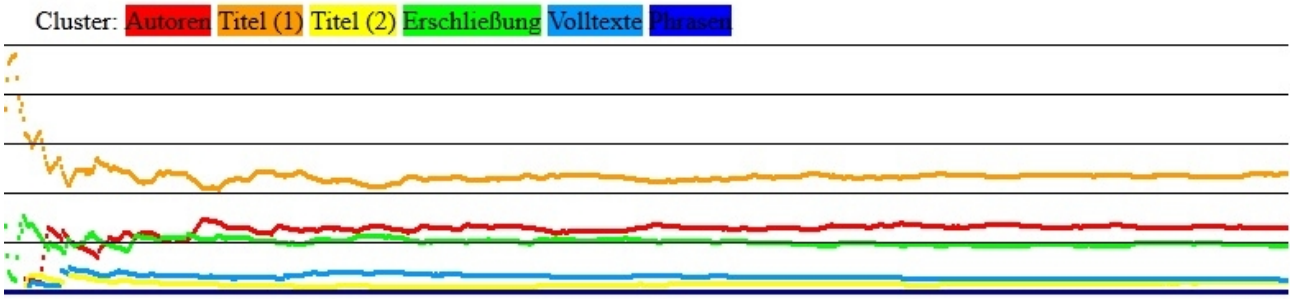
Erschließungscluster:

class	***35***
bklname	***20***
topic	***20***
geographic	***15***

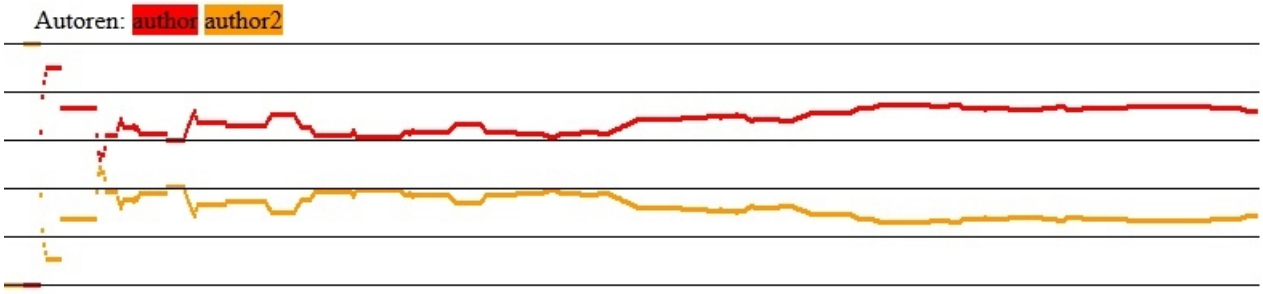
Volltextcluster:

abstract	***20***
contents	***20***
fulltext	***15***

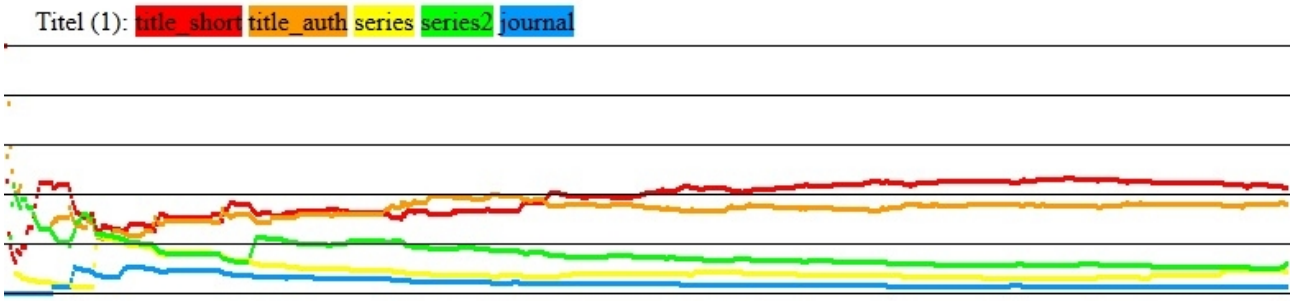
Als Ergebnis zeigt sich die Verteilung der Bewertungen auf die einzelnen Cluster so:



Das passt insgesamt schon sehr gut in die bislang erarbeiteten Vorüberlegungen: Maßgeblich ist der Titelcluster, gefolgt von Autoren- und Erschließungscluster, die - wie oben gezeigt - sehr gut greifen, wenn nach Namen, bzw. kontrolliertem Vokabular (zumindest in Hinblick auf die BK) gesucht wird. Die Volltexte tragen relativ wenig zur Gesamtbewertung bei, weil ihre Einträge als weit weniger relevant eingestuft wurden, die Nebentitelangaben, weil sie nur eine geringe Gesamtabdeckung haben. Beide tragen aber etwas zur Gesamtbewertung bei; d.h. zum Beispiel, dass hier einige Medien auf Grund ihrer Volltext-Einträge gefunden wurden. Nun folgt der Blick in die einzelnen Cluster:

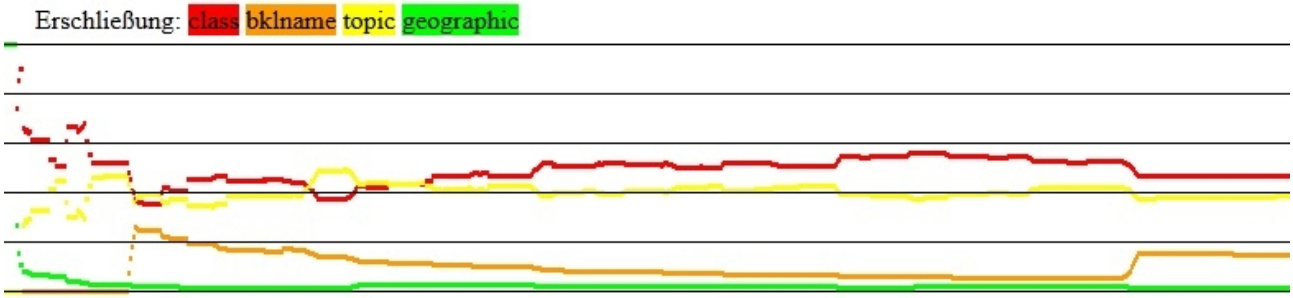


Im Autorencluster wurde - wie beabsichtigt - das Feld ***author2*** gegenüber ***author*** ein wenig aufgewertet.



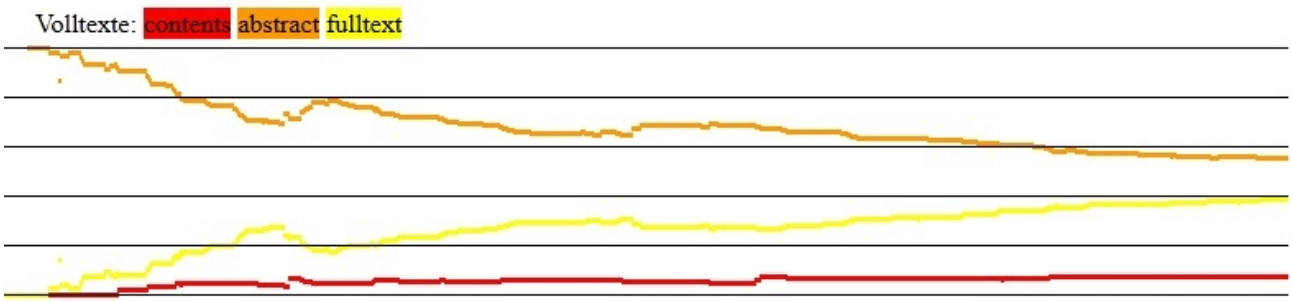
Im Haupttitel-Cluster sind jetzt die spezielleren Felder ***series***, ***series2*** und ***journal*** etwas besser bewertet. Interessant ist, dass die geringe Verschiebung der Paramter von ***title_auth*** und ***title_short*** von ***18:18*** zu ***19:17*** ein ganz anderes Bild ergibt und ***title_auth*** nun ähnlich hoch wie ***title_short*** bewertet. Hier wurde offenbar ein Schwellwert überschritten. Gerade weil er so bemerkenswert ist, wurde dieser Effekt mehrmals erfolgreich auf Reproduzierbarkeit getestet: Er zeigt sich unabhängig von der gewählten Stichprobe. Nun die Nebentitel mit dem erwarteten Bild:

Der Erschließungscluster:



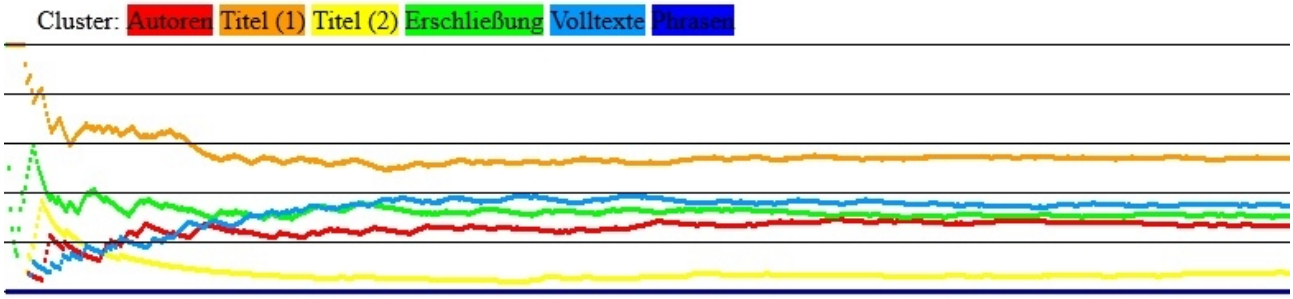
Hier wird nun ***topic*** ähnlich hoch wie ***class*** bewertet; ob das so gewünscht wird, wird bei der Schlussbetrachtung noch erörtert werden; ***geographic*** erscheint dagegen etwas unterbewertet.

Für den Volltextcluster ergibt sich schließlich folgendes Bild:



Ähnlich wie bei dem Cluster der Nebentitel ist hier auch zu erkennen, dass wegen der geringen Bewertung insgesamt, die statistische Auswertung des Clusters selbst ungenau wird. Hier wäre eine Erhöhung der betrachteten Suchanfragen angebracht.

Neben den Punktwerten lässt sich auch nur die Frage „Treffer oder nicht“ statistisch auswerten. Hier werden lediglich die Zahl der Treffer ausgewertet, die ein bestimmtes Feld oder ein bestimmter Cluster erzielt - bei einem Dismax-Algorithmus ist dies bei jedem Treffer genau ein Feld. Da die Höhe der vergebenen Ranking-Punkte die Bewertungen über den gesamten Datenbestand normiert, ist eine Auswertung unter Berücksichtigung der Punktezahl zwar zu berücksichtigen, aber eine Auswertungen der reinen Trefferzahlen geben unter Umständen noch weitere Hinweise auf die zu erwartende Punkteverteilungen. Im Vergleich der Cluster zueinander sieht eine Auswertung der reinen, unnormierten Trefferzahl so aus:



Das Ergebnis zeigt eindrucksvoll die wichtige Rolle, die die Normierung hierbei spielt: Da die Volltexte viel Text enthalten, tragen sie im Vergleich sehr häufig zum Erzielen eines Treffers bei. Aus demselben Grund werden diese Treffer aber sehr gering bewertet, was insgesamt zu einer geringen Bewertung der Volltexte führt. Auch beim Erschließungscluster ist ein solcher Effekt zu erkennen, wenn auch wesentlich schwächer.

Spezielle Anforderungen

Dabei sein ist alles

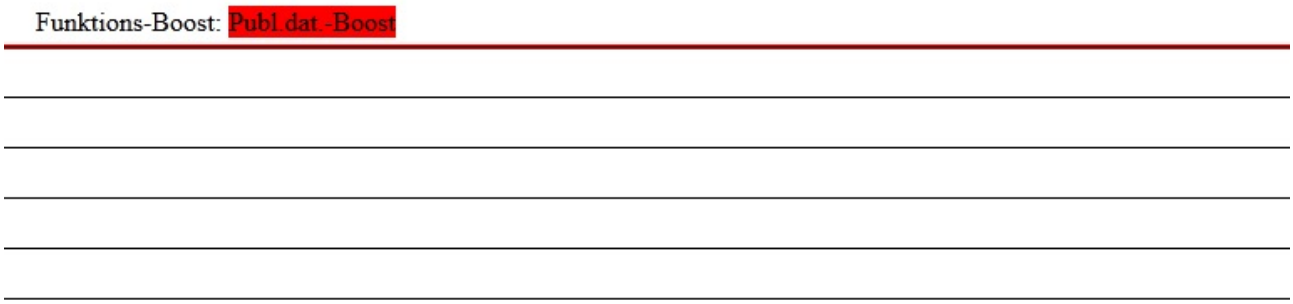
Oben wurde bereits angesprochen, dass es eine Reihe von Feldern gibt, die zwar durchsucht werden aber keinen Beitrag zum Ranking leisten sollen. Das können Felder mit speziellen Einträgen sein, etwa Normdaten-Ids oder andere Kennungen, wie hier ***isbn***, ***issn***, ***ctrlnum*** und ***normlink***; es können auch Felder sein, die dann zum Tragen kommen sollen, wenn die Ergebnismenge klein ist, wie hier ***publisher*** und ***publishPlace***.

Ergebnisse nach oben treiben

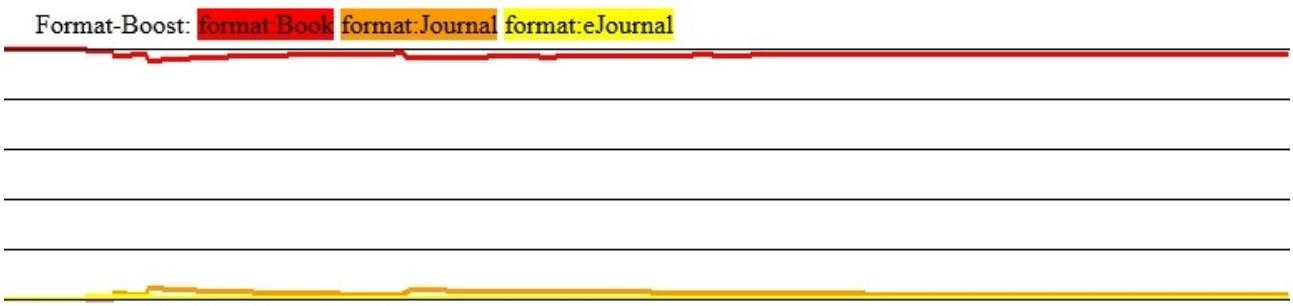
Solr bietet eine Reihe von Möglichkeiten, die Relevanzsortierung über die Gewichtung der Felder hinaus zu beeinflussen. Die bekannteste und wohl wichtigste ist das sogenannte „boosting“ („anheben“, „hochtreiben“). Neben dem hier betrachteten „querytime-boosting“, was durch eine Abfrage ausgelöst wird, gibt es auch ein „indextime-boosting“, das für einzelne Felder oder Dokumente bei der Indexierung festgelegt wird. Das multiplikative boosting, bei dem die Bewertung mit einem Faktor multipliziert wird, wird hier mangels Erfahrung damit auch nicht betrachtet. Beim additiven boosting werden zu den durch die Felder errechneten Bewertungen weitere Werte addiert. Diese Werte werden entweder durch Abfragen oder durch Funktionen über Feldwerte ermittelt. Additives boosting hat den Vorteil, dass der Beitrag, den es zur Gesamtbewertung leistet mit den anderen Beiträgen ermittelt werden kann.

Im Folgenden wurde - zunächst nur zur Demonstration - ein Funktions-Boost ***ord(publishDateSort)*** eingeführt, das den Rang des Publikationsjahrs nach aufsteigender Sortierung aller in Datenbestand vorhandener Publikationsjahre betrachtet, und zwar mit dem Gewicht 10. Dazu ein boosting des Feldes ***format***, wenn es ***Book*** beinhaltet, mit dem Gewicht 200, wenn es ***Journal*** beinhaltet, mit dem Gewicht 50 und, wenn es ***eJournal*** beinhaltet, mit dem Gewicht 25. Diese Werte werden unten noch weiter erörtert. Das Analysewerkzeug zeigt nun als Übersicht folgendes Bild:

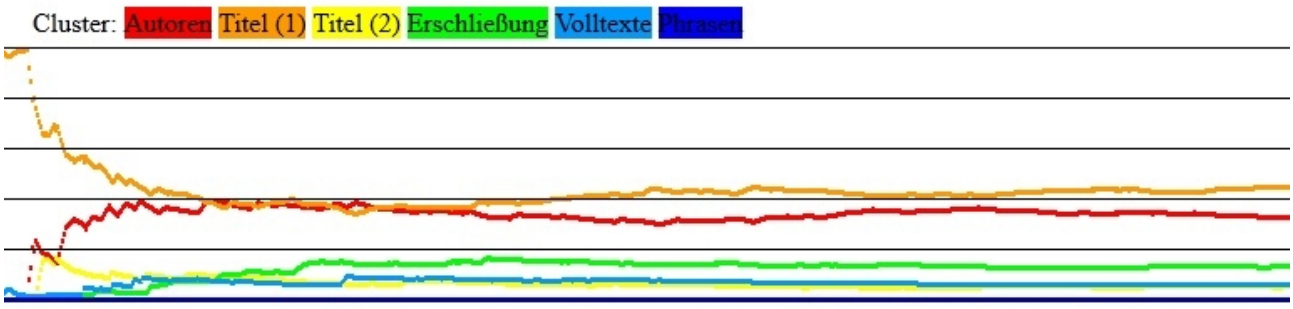
In rot ist hier der Beitrag aller Feldgewichte zu sehen, in orange das Funktions-boosting (Publikationsjahr) und in gelb das Format-boosting. Die Anzeige des Funktions-boostings ist redundant, da es nur eines davon gibt und dessen Anteil am gesamten Funktions-boosting daher 1 beträgt. Beim Format-boosting gibt es drei Parameter zu vergleichen, ***Book***, ***Journal*** und ***eJournal***; wie sie sich verteilen, ist im übernächsten Diagramm zu sehen. Zunächst das Funktions-boosting:



dann das Format-boosting:



und schließlich mit dieses zusätzlichen Einstellungen die Anzeige der Cluster:



Die Bewertungen der Cluster wird von diesen boostings nur wenig beeinflusst; zumindest aber scheinen aber die Autorenfelder in neueren Publikationen oder aber in Buch- oder Zeitschriftenpublikationen reichhaltiger zu sein, sodass hier dieser Cluster entsprechend mehr zur Gesamtbewertung der Felder beiträgt.

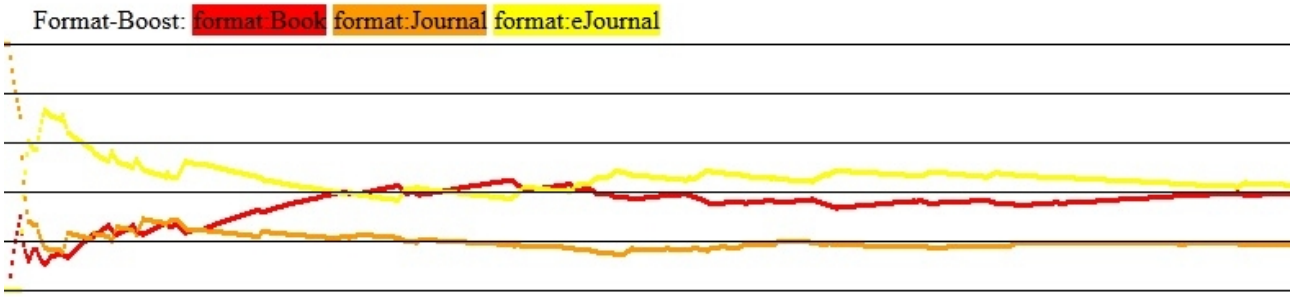
Abfragen boosten: Medientypen

Ein typisches Einsatzfeld für Abfrage-basiertes boosting auf bibliothekarisch erfassten Daten ist das Anheben der Bewertung von bestimmten Medientypen. In der Betrachtung hier sind es Bücher, Zeitschriften und elektronische Zeitschriften. Im Index sind sie mit ***Book***, ***Journal*** oder ***eJournal*** im Feld ***format*** bezeichnet. Es ist für dieses Abfrage-basierte boosting auf jeden Fall angeraten, dies nur über Felder einzurichten, deren Inhalte kontrolliert sind. Stünde etwa für Bücher in dem ***format***-Feld manchmal ***Book***, manchmal aber ***Books*** oder ***Buch*** oder ***Bücher***, wäre so ein boosting deutlich unzuverlässiger und als Relevanzkriterium weit weniger geeignet.

Sollen diese boostings bei allen drei Medien ungefähr gleich sein, so sollten die statistischen Werte in etwa die Häufigkeit widerspiegeln, in der die jeweiligen Medien im Datenbestand vorkommen. Im vorliegenden Beispiel ist dies:

Book	***48,2%***
Journal	***2,7%***
eJournal	***0,7%***

Dadurch, dass die Häufigkeiten so ungleich verteilt sind, ist es schwierig, die Gewichtung der boostings rein nach dem Diagramm zu ermitteln. Hier macht es u.U. Sinn, sich die Rohdaten anzusehen, die von dem Analysewerkzeug ermittelt werden und die die Grundlage der Diagramme bilden. Um die Ausgangslage zu beurteilen, macht es hier Sinn, erst einmal die Verteilung zu betrachten, wenn alle drei Medien gleich hoch bewertet werden - am besten auch sehr hoch, weil dann dieses Formate-boosting für die Gesamtbewertung ausschlaggebend ist. Das Ergebnis sieht wie folgt aus:



Da die von solr vorgenommenen Normierungen die Häufigkeit der Einträge ausgleichen soll und es hier, im Format-Feld, um wohldefinierte Einträge geht, wäre zu erwarten, dass alle drei Medien etwa gleich viel zur mittleren Bewertung der Suchergebnisse beitragen. Eine Betrachtung der Verteilung der Medientypen bei einzelnen Suchabfragen - auch mit den bei allen drei Medientypen gleichen boosting-Parametern - zeigt aber, dass sich je nach Suche die Medientypen der Ergebnisse stark unterscheiden. Mal sind da fast nur Bücher zu finden, mal nur E-Zeitschriften. Die Bewertungen mit Ranking-Punkten ist dagegen einigermaßen einheitlich, wie sich nach Begutachtung einiger Einzelergebnisse zeigt:

	Punkte	***zum Quadrat***	***mult. mit der Häufigkeit***
Book	***ca. 0,7***	***0,5***	***24***
Journal	***ca. 3,1***	***9,6***	***26***
eJournal	***ca. 5,2***	***27,0***	***19***

Die Größenordnungen in der letzten Spalte sind in etwa gleich; dass hier Schwankungen auftreten, ist nach den Beobachtungen zu den Einzeltreffern und angesichts der großen Häufigkeitsunterschiede nachvollziehbar. Was heißt das nun für die boosting-Faktoren für die einzelnen Medientypen? Damit alle drei in gleicher Weise aufgewertet werden, sollten die Wurzeln der relativen Häufigkeiten die Verhältnisse der boosting-Faktoren bestimmen, also ***0,8 : 1,6 : 6,9***, oder beginnend mit 25 in etwa: ***25 : 50 : 200***.

Als weitere Anforderung gab es bei beluga den Wunsch, Zeitschriften- oder Reihenbände gegenüber Zeitschriften bzw. Reihen aufzuwerten. Die Bände haben im Index fast immer das Format ***Book***; sie geben sich durch die Angabe eines ***ppnlink*** im Index zu erkennen, mit dem die zugehörige Reihe ermittelt werden kann. Hier die wesentlichen Abdeckungen für Medien mit ***ppnlink***:

alle	***64%***
Aufsätze	***47%***
Bücher	***36%***
Zeitschriftenbände	***12%***

Lösen lässt sich diese Anforderung mit einem boosting für Datensätze mit einem ***ppnlink***-Eintrag und dem Format ***Book***, etwa ***ppnlink:* AND format:Book***. Die solr-eigene Analyse des Rankings gibt hierzu beispielhaft folgende Erläuterung:

1.4504247 = (MATCH) weight(format:Book^200.0 in 490426) [DefaultSimilarity], result of:
1.4504247 = score(doc=490426,freq=1.0), product of:
0.57571125 = queryWeight, product of:
200.0 = boost
2.5193613 = idf(docFreq=3667583, maxDocs=16758311)
0.0011425738 = queryNorm
2.5193613 = fieldWeight in 490426, product of:
1.0 = tf(freq=1.0), with freq of:
1.0 = termFreq=1.0
2.5193613 = idf(docFreq=3667583, maxDocs=16758311)
1.0 = fieldNorm(doc=490426)
0.23576689 = (MATCH) sum of:
0.0072521227 = (MATCH) weight(format:Book in 490426) [DefaultSimilarity], result of:
0.0072521227 = score(doc=490426,freq=1.0), product of:
0.002878556 = queryWeight, product of:
2.5193613 = idf(docFreq=3667583, maxDocs=16758311)
0.0011425738 = queryNorm
2.5193613 = fieldWeight in 490426, product of:
1.0 = tf(freq=1.0), with freq of:
1.0 = termFreq=1.0
2.5193613 = idf(docFreq=3667583, maxDocs=16758311)
1.0 = fieldNorm(doc=490426)
0.22851476 = (MATCH) ConstantScore(ppnlink:*^200.0)^200.0, product of:
200.0 = boost
0.0011425738 = queryNorm

In der Bewertung wird dieser Term ***ppnlink:* AND format:Book*** tatsächlich in zwei Terme aufgeteilt, die dann summiert werden. Da aber der Term ***format:Book*** nicht geboostet wird, trägt er so gut wie nichts zur Gesamtwertung bei. Insbesondere wertet er auch nicht die Bücher zusätzlich auf. Der Wert, den dieser Term insgesamt einbringt, ist vergeichsweise niedrig, sodass in der Schlussbetrachtung über eine Erhöhung des „Bände-boostings“ nachgedacht werden kann. Im Übersichtsdiagramm zeigt sich die Situation nun so:

Der Beitrag des Bände-boostings ist tatsächlich vernachlässigbar; „Pi mal Daumen“ wird er daher vorerst von 200 auf 500 erhöht. Aber jetzt weiter zu weiteren Aspekten der Relevanz-Bewertung.

Funktionswerte boosten: Aktualität

Eine weitere typische Anforderung ist die Aufwertung aktuellerer Bestände. Dies kann über ein funktionsbasiertes boosting geschehen. Die Reihenfolge des Publikationsjahres ist daher eine gute Wahl, weil diese Größe unabhängig von der Verteilung der Medien auf die einzelnen Publikationsjahre ist und eine dem Datenbestand angepasste Skala darstellt. Die Alternative, den Abstand zwischen Veröffentlichungsjahr zum aktuellen Jahr zu bewerten, hat den Nachteil, dass dies insbesondere in den noch nicht so sehr vergangenen Jahren schlecht skaliert: In Relation zu 2015 würde etwa 2014 doppelt so hoch bewertet werden wie 2013.

Allerdings sind hierbei auch einige Aspekte der Katalogisierung zu beachten, insbesondere auch der Umstand, dass Zeitschriften immer das Ersterscheinungsjahr als Publikationsdatum erhalten. Zeitschriften, die es schon lange gibt, wie etwa „nature“ wären hier im Nachteil. Um dies auszugleichen, kann etwa den Zeitschriften unterstellt werden, dass sie immer aktuell sind und entsprechend geboostet werden. An Stelle der oben verwendeten Funktion ***ord(publishDate)*** wird dann

if(exists(query({!v=format:Journal})),737,ord(publishDateSort))

verwendet. 737 entspricht aktuell etwa dem Rang des Jahres 2015 (was sich bei Neuindexierungen aber immer ändern kann). Die Beiträge der boostings und der Gesamtheit der Felder sehen dann wie folgt aus:

Der Unterschied zu oben ist erwartungsgemäß nicht sehr groß. Zu beachten ist hier, dass in der solr-Ausgabe diese Funktion etwas anders aussieht:

if(exists(query(format:Journal,def=0.0)),const(737),ord(publishDateSort))

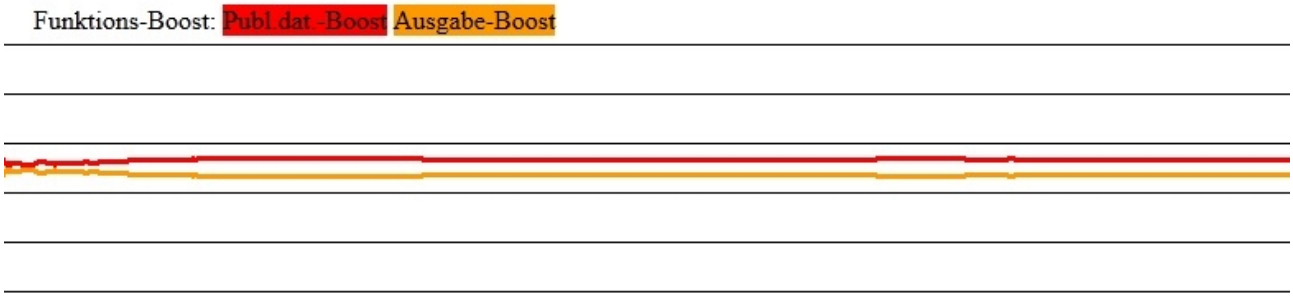
Anders als die „boosting-queries“, also das boosting der Formate oder der Bände, erreicht das boosting über den Rang des Veröffentlichungsjahrs in aller Regel nicht den Wert 0. Um eine Vergleichbarkeit dennoch zu gewährleisten, wird hier der jeweils kleinste Betrag aller analysierten Suchen abgezogen.

Eine weitere Anwendung für das boosting mit Hilfe einer Funktion ist das Aufwerten aktueller gegenüber weniger aktuellen Ausgaben - wo mehrere Ausgaben existieren. Das Feld ***edition*** ist im Index bei etwa 11% des Datenbestandes gesetzt. Unter der Annahme, dass Werke mit mehreren Ausgaben einigermaßen aktuell sind, kann dies realisiert werden, indem bei Medien, die mehrere Ausgaben haben, das Publikationsjahr um einen weiteren Betrag aufgewertet wird:

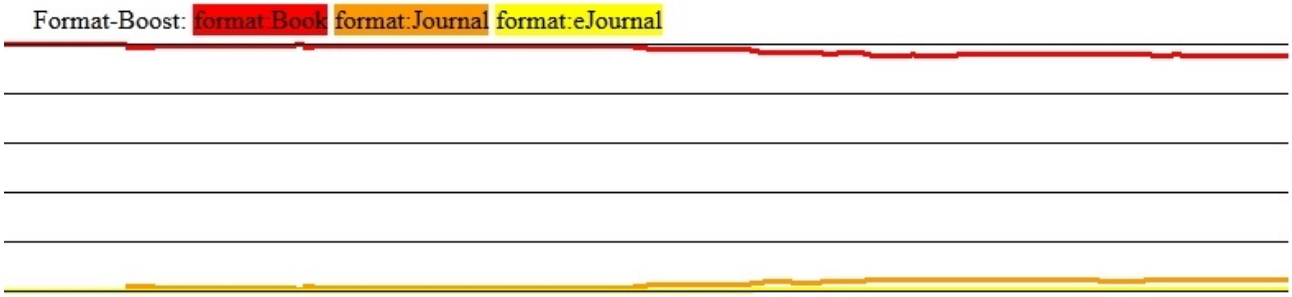
if(exists(query({!v=edition:*})),ord(publishDateSort),737)

Im Ergebnis zeigt sich nun das folgende Übersichtsdiagramm. Die Feinabstimmung wird dann nach Betrachtung (weniger) weiterer Gesichtspunkte am Ende vorgenommen. Hier die Übersicht:

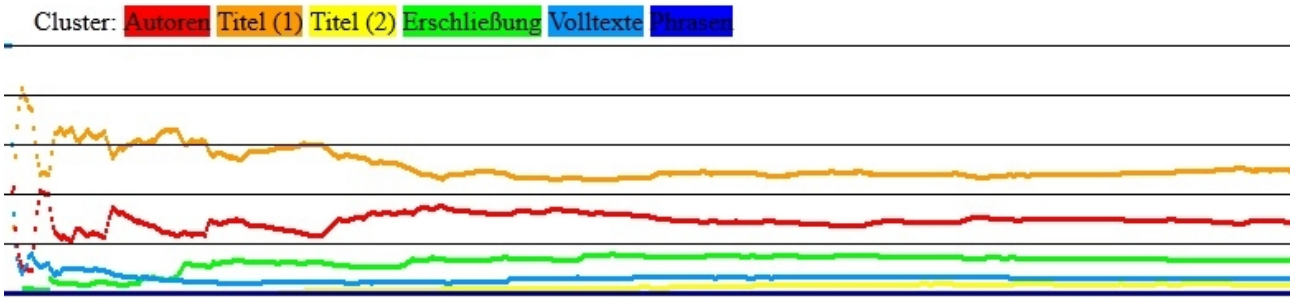
die beiden boosting-Funktionen:



das Formate-boosting:



und der Vollständigkeit halber auch die Cluster:



Phrasen dreschen

Die Suche nach einer Signatur sollte idealer Weise ein eindeutiges Ergebnis hervorbringen, wenn die Signatur im Suchraum eindeutig ist. Obwohl im Indexfeld ***signature*** die Signaturen aller im GBV-Index vorhandener Bibliotheken aufgelistet sind, sind die darin gelisteten Signaturen erstaunlich häufig eindeutig. Dennoch gestaltet sich eine Signaturensuche bei genauerer Betrachtung komplexer als zunächst angenommen. In der Hamburger Staatsbibliothek werden Signaturen gerne mal in Groß- und Kleinschreibung umgesetzt, obwohl sie auch kleingeschrieben suchbar sein sollten. So

sollte die Signatur ***F WiSo 260/3*** auch mit dem Term ***f wiso 260/3*** gefunden werden. Die einfache Suche nach ***f wiso 260/3*** liefert 3 Ergebnisse:

Rang 1 hat 87 Einträge im Feld ***signature*** und wird mit 3 Teiltreffern gefunden:

H WiSo 087/5, V 27-1-260(3), F b 38

Rang 2 hat 57 Einträge im Feld ***signature*** und wird mit einem Teiltreffer gefunden:

F WiSo 260/3

Rang 3 hat 74 Einträge im Feld ***signature*** und wird mit einem Teiltreffer gefunden:

Ex H WiSo 260/3, L oek 2685/003(6)f

Gefunden werden sollte aber nur der Treffer mit dem Rang 2; vielleicht hilft ja eine Phrasensuche. Die Suche nach ***„F WiSo 260/3“*** liefert wie erwartet das Medium mit der gesuchten Signatur als einzige Medium. Aber die Suche nach ***„f wiso 260/3“*** liefern kein Ergebnis; warum?

Als regulärer Term wird ***F WiSo 260/3*** wie folgt in einzelne Tokens zerlegt:

(+((DisjunctionMaxquery((signature:f)) DisjunctionMaxquery((signature:"wi so")) DisjunctionMaxquery((signature:"260 (iii 3 three)")))~3) ())/no_coord

bzw. in Kurzform:

+(((signature:f) (signature:"wi so") (signature:"260 (iii 3 three)"))~3) ()

Zur Erinnerung: Der Suchterm wird an Leerzeichen, Übergängen von Klein- zu Großschreibung und an Interpunktions- und Sonderzeichen (außer dem .) in Tokens aufgeteilt. In einer Synonymliste werden auch ***iii***, ***3*** und ***three*** als Synonyme benannt. ***f wiso 260/3*** wird daher wie folgt zerlegt:

+(((signature:f) (signature:wiso) (signature:"260 (iii 3 three)"))~3) ()

Beide Suchterme finden also die drei oben genannten Ergebnisse (weil alle drei den Term ***wiso*** enthalten; gäbe es ein Medium, das eine Signatur mit dem Term ***wi***, eine andere mit dem Term ***so*** enthielte, würde es vom ersten, nicht aber vom zweiten Suchterm gefunden).

Die Phrasensuche nach ***„F WiSo 260/3“*** wird dagegen wie folgt bearbeitet:

+(signature:"f wi so 260 (iii 3 three)") ()

und findet die Signatur ***F WiSo 260/3***, die vom Tokenizer wie folgt zerlegt wird (zu beachten die im Index-Teil der Konfiguration gesetzten catenate-Attribute):

Position:	***1***	***2***	***3***	***4***	***5***
	f	***wi***	***so***	***260***	***3***
			wiso		***2603***

zum Vergleich die Abfrage ***F WiSo 260/3***, ***f wiso 260/3*** und ***„F WiSo 260/3“***:

Position:	***1***	***2***	***3***	***4***	***5***
	f	***wi***	***so***	***260***	***3***
	f	***wiso***	***260***	***3***	
	f	***wi***	***so***	***260***	***3***

Die Suche nach ***„f wiso 260/3“*** findet aber nichts:

Position:	***1***	***2***	***3***	***4***	***5***
	f	***wiso***	***260***	***3***	

Denn anders als in der Abfrage befindet sich hier das Token ***wiso*** an der Position 3 und ist vom ***f*** an Position 1 um 2 Positionen entfernt. Die Phrasensuche bedeutet aber, dass sich die Entfernung der Positionen voneinander nicht unterscheiden darf. Für diesen Fall gibt es einen weiteren Parameter, der bei der Suche angegeben werden kann, nämlich der ***query-slop***-Parameter ***qs***. Er gibt die maximale Distanz an, mit der die einzelnen Terme einer Phrasensuche „vermanscht" werden können, um als Treffer zu zählen. Mit einem ***query-slop*** von 1 wird die gesuchte Signatur auch mit der kleingeschriebenen Phrasensuche gefunden.

Eine andere Möglichkeit, die Signaturensuche zu verbessern, wäre, die einfache Suche so zu gestalten, dass die gewünschte Signatur durch die Relevanzsortierung an erste Stelle rücken würde. Bei den 3 Treffern der Suche nach ***f wiso 260/3*** errechnen sich die Relevanzwerte wie folgt:

Rang 1:

2.2839546 = (MATCH) sum of:
0.06823718 = (MATCH) weight(signature:f in 166808) [tf = 1,732]
0.124108694 = (MATCH) weight(signature:wiso in 166808) [tf = 1,0]
2.0916088 = (MATCH) weight(signature:"260 (iii 3 three)" in 166808) [tf = 1,414]

Rang 2:

1.9622086 = (MATCH) sum of:
0.09192576 = (MATCH) weight(signature:f in 72807) [tf = 2,0]
0.14479347 = (MATCH) weight(signature:wiso in 72807) [tf = 1,0]
1.7254894 = (MATCH) weight(signature:"260 (iii 3 three)" in 72807) [tf = 1,0]

Rang 3:

1.6624798 = (MATCH) sum of:
0.06755061 = (MATCH) weight(signature:f in 187830) [tf = 1,732]
0.11958483 = (MATCH) weight(signature:wiso in 187830) [tf = 1,0]
1.4753443 = (MATCH) weight(signature:"260 (iii 3 three)" in 187830) [tf = 1,0]

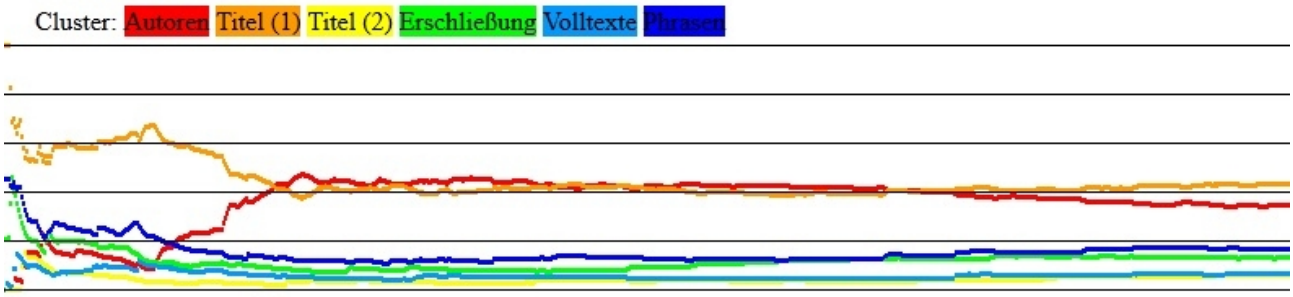
Dass der Term ***260 (iii 3 three)*** so viel höher bewertet wird als die anderen beiden, liegt an der vergleichsweise hohen idf, die dadurch zu Stande kommt, dass dieser Term aus 4 Subtermen besteht. Der erste Rang wird also dem Medium vergeben, das 3 Signaturen hat, die den - hoch bewerteten - Subterm ***260*** enthalten. Eine Möglichkeit, dieses eher unerwünschte Verhalten zu korrigieren, wäre, das Auffinden der ganzen Phrase mit Extrapunkten bei der Relevanzsortierung zu „belohnen". Dafür kann der Parameter ***pf=signature*** gesetzt werden, mit dem das Auffinden der Phrase im Suchfeld zusätzlich bewertet wird. Um allerdings diese Extrabewertung auch für die kleingeschriebene Suche zu erhalten, muss auch hier die Phrase „vermanscht" werden und zwar mit dem Parameter ***ps***, der daher auf 1 gesetzt wird. Damit rückt das gewünschte Ergebnis an die erste Stelle:

2.664291 = (MATCH) sum of:
1.1520983 = (MATCH) sum of:
0.053973623 = (MATCH) weight(signature:f in 72807)
0.08501456 = (MATCH) weight(signature:wiso in 72807
1.0131102 = (MATCH) weight(signature:"260 (iii 3 three)" in 72807)
1.5121927 = (MATCH) weight(signature:"f wiso 260 (iii 3 three)"~1 in 72807)

Da die idf auch für die Phrase als Summe der idfs aller Teilabfragen berechnet wird, erfährt die Phrase hier die höchste Bewertung von allen Teilbewertungen und hebt das entsprechende Medium mit großem Abstand an die erste Stelle.

Mit ***qf*** und ***qs*** lässt sich also die Art und Weise, wie Suchabfragen als Phrasen behandelt werden, beeinflussen; eine solche Beeinflussung ändert auch die Ergebnismenge. ***pf*** und ***ps*** beeinflussen die Art und Weise, wie die Feldeinträge als Phrasen behandelt werden und somit nur das Ranking, nicht die Ergebnisse. Insbesondere können damit Treffer aufgewertet werden, die als Phrasen treffen, obwohl keine Phrasensuche durchgeführt wird.

Wenn jeweils etwa ein Viertel der Feldbewertungen im Hauptitel-Cluster (***title_auth***, ***series***, ***title_short***, ***series2*** und ***journal)*** und im Feld ***topic***, sowie die Hälfte der Bewertungen in den Volltext-Feldern ***abstact*** und ***fulltext*** mit einem solchen „Phrasen-boosting“ versehen werden, zeigt sich in der Cluster-Analyse folgendes Ergebnis. Eine solche Verteilung des Phrasen-boostings macht daher Sinn, weil dies Felder sind, in denen entweder viel Text zu erwarten ist (Volltexte) oder in denen ein Phrasentreffer ein Qualitätsmerkmal des Ergebnisses darstellt. Phrasentreffer auf Feldern aufzuwerten, in denen normierte Einträge zu erwarten sind (etwa Autoren oder Klassifikationen), ist weniger sinnvoll.



Die Aufwertung gefundener Phrasen bewegt sich dann etwa in der Größenordnung der Bewertungen des Erschließungssclusters. Die hier angezeigte Phrasenkurve in Dunkelblau ist dem Diagramm zusätzlich hinzugefügt worden; sie beeinflusst die anderen Werte daher nicht. Zu sehen ist aber, dass durch die Konzentration der Phrasenaufwertung im Haupttitel-Cluster, die Bewertung des Clusters insgesamt im Mittel abnimmt. Auch das wird bei der Endbetrachtung korrigiert werden.

Hauptsache das stemming stimmt

Vor dem Vergleich von Suchabfragen mit Feldinhalten, werden beide Teile nicht nur durch den Tokenizer in einzelne Wortbestandteile zerlegt, sondern auch noch um ihre grammatikalischen Formen reduziert. Das macht Sinn, da eine Suche etwa nach dem „Hamburger Rathaus“ auch das Buch mit dem Titel „Das Rathaus in Hamburg“ finden möchte. Allerdings ist das ***stemming***, das Reduzieren von Wörtern auf ihre Grundformen, mit einigen Fallstricken behaftet, weswegen dem stemming nicht allzu viel Gewicht zugemessen werden sollte. Insbesondere sind die Bestände einer Bibliothek in der Regel vielsprachig, sodass eine adäquate Zuordnung eines passenden stemming-Algorithmus bei der Indexierung der Metadaten sehr aufwendig bis unmöglich sein wird.

Im Fall des findex der VZG werden die Felder ***title_full***, ***topic***, ***abstract***, ***fulltext*** und ***allfields*** auch als „_unstemmed“ angeboten. In den hier für die Relevanzbewertung verwendeten Feldern ***topic***, ***abstract*** und ***fulltext*** soll davon auch Gebrauch gemacht werden, indem die vollen („unstemmed“) Begriffsformen eine Aufwertung erfahren. Da in allen drei Feldern auch Phrasen aufgewertet werden sollen, macht es Sinn, dies zu kombinieren:

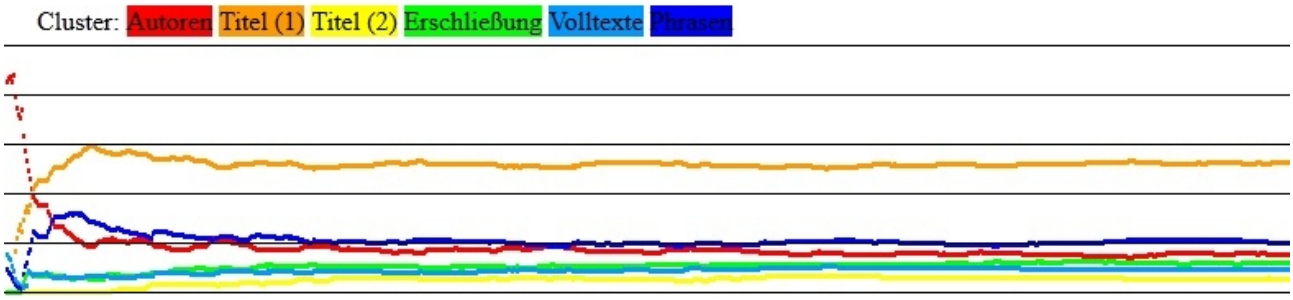
qf:topic^15
pf:topic_unstemmed^5

Finale Justierungen

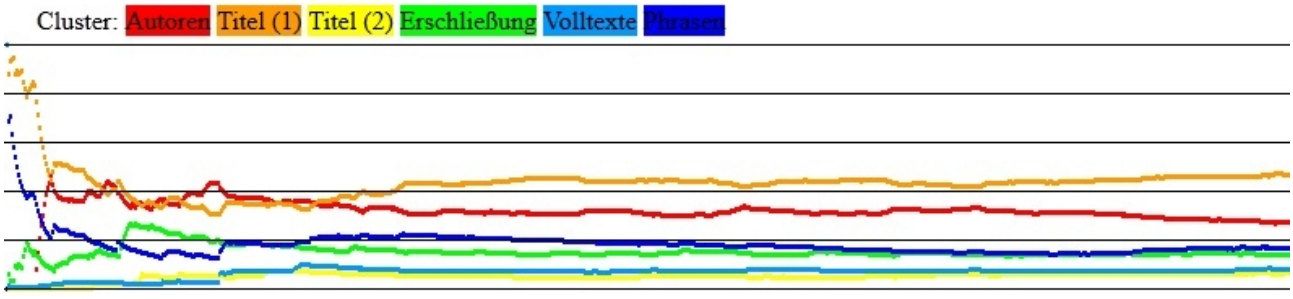
Bis hierhin wurden die meisten relevanten Aspekte einer Relevanz-Bewertung betrachtet. Es gibt nun noch ein paar wenige globale Einstellungen, die es Wert sind, bedacht zu werden, insbesondere der sogenannte ***tiebreaker*** und das Minimum der Einzelbegriffe eines Suchausdrucks, die ein Treffer enthalten sollte.

To tie or not to tie

Der Dismax-Algorithmus hat als Kern die Idee, dass für eine Relevanzbewertung ausschließlich das Feld in die Gesamtbewertung eingeht, das die höchste Bewertung erhält; die anderen Feldbewertungen bleiben unberücksichtigt. Das bedeutet der Wortteil „disjunction“. Es gibt aber auch die Möglichkeit, eine Kollisionsregel, engl. ***tiebreaker***, einzusetzen, der aber hier weniger die Aufgabe hat, bei einem Bewertungsgleichstand (der bei einer großen Grunddatenmenge sehr unwahrscheinlich ist) eine Entscheidung herbei zu führen, sondern vielmehr auch die anderen Feldbewertungen sozusagen mitspielen zu lassen. Der tiebreaker ist dabei der Faktor, mit dem die anderen Bewertungen dem maximalen Feldwert zugerechnet werden und der daher zwischen 0 und 1 liegt. Bei 0 haben wir den Dismax-Algorithmus wie bislang betrachtet, bei 1 hätten wir ihn in einen „ConMax“, also conjunction-maximum-Algorithmus verwandelt. Er weicht also die rigide „nur das Maximum zählt“-Regel auf. Würde er bei den bis hier erarbeiteten Einstellungen auf 1 gesetzt, sähe die Verteilung der Bewertungen der einzelnen Cluster so aus:



Die Verschiebungen im Vergleich zur Verteilung ohne tiebreaker sind deutlich zu erkennen. In der Literatur wird empfohlen, für den tiebreaker einen Wert um 0,1 zu wählen, was auch bei beluga zu guten Ergebnissen führt. Die Verteilung der Bewertungen auf die Cluster zeigt sich damit wie folgt:



Will man wirklich alles finden?

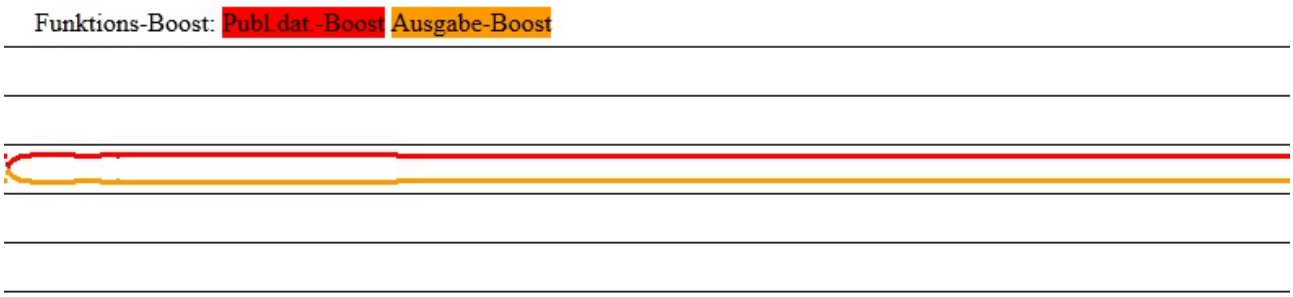
Schließlich gibt es noch Möglichkeiten, dass nur eine begrenzte Anzahl von Suchwörtern eines Mehrwortbegriffs treffen müssen, um ein Dokument als Treffer zu klassifizieren. Der entsprechende Parameter heißt ***mm*** („minimum should match“) und nimmt sowohl eine Zahl als auch einen Prozentwert an. Nach unseren Erfahrungen funktioniert er genauso wie in der Dokumentation beschrie-

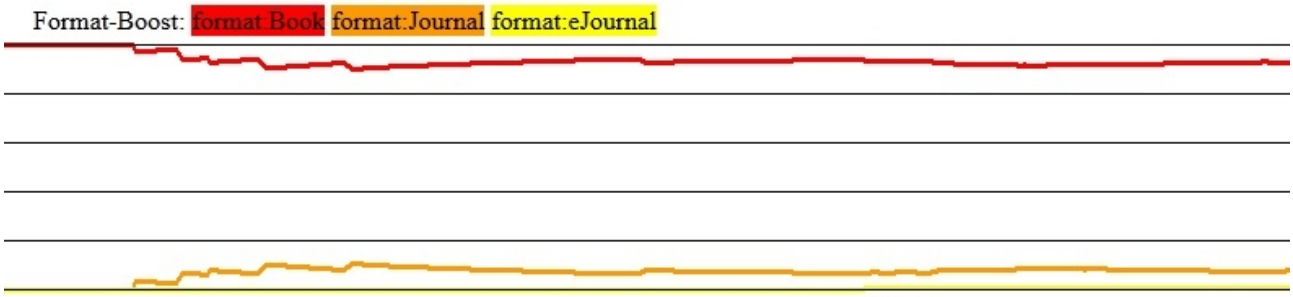
ben. Bei beluga kommt er allerdings bislang nicht zum Einsatz, da dadurch die „known-entity“-Suchen, d.h. Suchen, bei denen die Suchenden wissen, was sie finden wollen, und etwa einen ganzen Buchtitel eingeben, „verrauscht“ und damit erschwert werden. Ohne Parameter muss jeder Suchterm in mindestens einem der durchsuchten Indexfelder vorkommen.

Auf der Zielgeraden

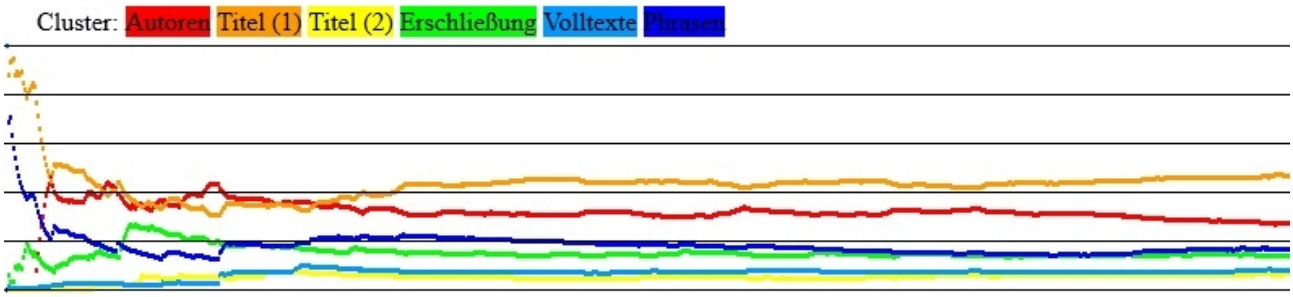
Zum Schluss wird die gesamte Einstellung noch einmal betrachtet und einer kritischen Bewertung unterzogen. Hier die Analyse-Diagramme der bislang erarbeiteten Konfiguration; erst die Übersicht:

die boostingfunktionen und das Formateboosting:





noch einmal die Cluster:



Vor der abschließenden Betrachtung dieser Diagramme sollte noch der Tatsache nachgegangen werden, dass sich im Vergleich zur einfachen Anfangskonfiguration die durchschnittliche Bearbeitungszeit der Abfragen merklich verlängert hat. Lag sie anfangs noch in einem Bereich bis zu etwa 0,2 Sekunden, liegt sie jetzt in der Regel zwischen 2 und 4 Sekunden. Das macht eine Laufzeitanalyse der Abfragen erforderlich. Dabei fällt auf, dass es im Wesentlichen die boostingqueries und boosting-Funktionen sind, die die Laufzeit verlängern. Die Bearbeitungszeit für den solr-Index - sie lässt sich ebenfalls aus dem Debug-Teil der Antwort entnehmen - beträgt jetzt im Mittel etwa 5 bis 10 Sekunden. Da mehrere Instanzen des Servers parallel betrieben werden, liegt die tatsächlich Antwortzeit unter diesem Wert. Die Analyse ergibt, dass einzig das Weglassen des Bände-boostings die mittlere Bearbeitungszeit auf etwa 1,5 Sekunden reduziert und damit die Antwortzeit auf etwa 0,5 Sekunden. Damit lässt sich leben.

Bei den Feldern zeigt sich, dass die Volltextfelder etwas stärker gewichtet werden könnten, ebenso die Nebentitel und das Feld ***author*** gegenüber ***author2***, sowie ***topic*** gegenüber ***class***. Auch scheinen beim Formate-boosting die E-Zeitschriften etwas zu kurz zu kommen. Über eine Analyse von Einzelabfragen kommt der Eindruck auf, dass das boosting des Veröffentlichungsjahrs etwas erhöht werden könnte. Die Abfrage etwa von „Alster Hamburg“ (1122 Treffer insgesamt) ergibt die ersten 10 Treffer (jeweils mit Medientyp und Veröffentlichungsjahr in Klammern nach dem Titel):

Rang:1 PPN:171031849 Hamburg an der Alster (Journal, 1990)
Rang:2 PPN:042819946 Alster Materialien für Aus- und Fortbildung (Book, 1988)
Rang:3 PPN:511917538 Hamburg Alster, Elbe, Metropole (Journal, 2005)
Rang:4 PPN:130190306 Jahrbuch des Alstervereins e. V (Journal, 1901)
Rang:5 PPN:039954730 Die schöne Hamburgerin (Book, 1982)
Rang:6 PPN:482306777 Sieh Dir an, wie Hamburg baut ... (Journal, 1963)
Rang:7 PPN:191056677 Alster-Rundschau Hamburger Monatszeitung (Journal, 1992)
Rang:8 PPN:184972108 Hamburg, rund um die Alster (Journal, 1994)
Rang:9 PPN:551473983 Hamburg, rund um die Alster (Journal, 1994)
Rang:10 PPN:166724262 Blickpunkt Hamburg (Journal, 1983)

und nach Erhöhen der boostings für das Veröffentlichungsjahr von (10/5) auf (17/8):

Rang:1 PPN:171031849 Hamburg an der Alster (Journal, 1990)
Rang:2 PPN:511917538 Hamburg Alster, Elbe, Metropole (Journal, 2005)
Rang:3 PPN:130190306 Jahrbuch des Alstervereins e. V (Journal, 1901)
Rang:4 PPN:184972108 Hamburg, rund um die Alster (Journal, 1994)
Rang:5 PPN:482306777 Sieh Dir an, wie Hamburg baut ... (Journal, 1963)
Rang:6 PPN:042819946 Alster Materialien für Aus- und Fortbildung (Book, 1988)
Rang:7 PPN:551473983 Hamburg, rund um die Alster (Journal, 1994)
Rang:8 PPN:191056677 Alster-Rundschau Hamburger Monatszeitung (Journal, 1992)
Rang:9 PPN:166724262 Blickpunkt Hamburg (Journal, 1983)
Rang:10 PPN:563735856 Jahresbericht des Allgemeinen Alster-Club Hamburg (Journal, 1905)

Die Bücher sind jetzt bis auf eines aus den „Top 10“ komplett verschwunden. Kein Wunder, da ja Zeitschriften durch das Boosten des Veröffentlichungsjahrs begünstigt werden; sie gelten ja immer als „brandaktuell“. Daher ist jetzt ein kleiner Griff in die Trickkiste erforderlich und das boosting der Zeitschriften wird reduziert, um einen Ausgleich zu schaffen, und zwar von 50 auf 30; mit einem Ergebnis, das deutlich besser aussieht:

***Rang:1** PPN:171031849 Hamburg an der Alster (Journal, 1990)*
***Rang:2** PPN:042819946 Alster Materialien für Aus- und Fortbildung (Book, 1988)*
***Rang:3** PPN:511917538 Hamburg Alster, Elbe, Metropole (Journal, 2005)*
***Rang:4** PPN:386843104 1859 - 1999 - eine kleine Chronik ... (Book, 1999)*
***Rang:5** PPN:562982035 Hamburg Entdeckungen an Elbe und Alster ... (Book, 2008)*
***Rang:6** PPN:039954730 Die schöne Hamburgerin (Book, 1982)*
***Rang:7** PPN:26535594X Hamburgs Alster Postkartenbuch (Book, 1998)*
***Rang:8** PPN:130190306 Jahrbuch des Alstervereins e. V (Journal, 1901)*
***Rang:9** PPN:314819428 Die Alster ein Alltagsmärchen in 48 Bildern und ... (Book, 2000)*
***Rang:10** PPN:351012214 Elbe, Alster, Jungfernstieg Hamburger Landgänge (Book, 2002)*

Hier als Ergebnis die abschließend ermittelten Konfigurations-Parameter:

Haupttitelcluster:
qf, title_auth^14
pf, title_auth^5
qf, series^11
pf, series^4
qf, title_short^12
pf, title_short^5
qf, series2^9
pf, series2^3
qf, journal^7
pf, journal^3
Nebentitelcluster:
qf, title_alt^12
qf, title_old^12
qf, title_new^12
Autorencluster:
qf, author^20
qf, author2^16
Erschließungscluster:
qf, topic^15
pf, topic_unstemmed^7
qf, geographic^15
qf, class^30
qf, bklname^20
Volltextecluster:
qf, abstract^10
pf, abstract_unstemmed^12
qf, contents^22
qf, fulltext^8
pf, fulltext_unstemmed^9
übrige Felder:
qf, publisher^0
qf, publishPlace^0
qf, publishDateSort^0
qf, isbn^0
qf, issn^0

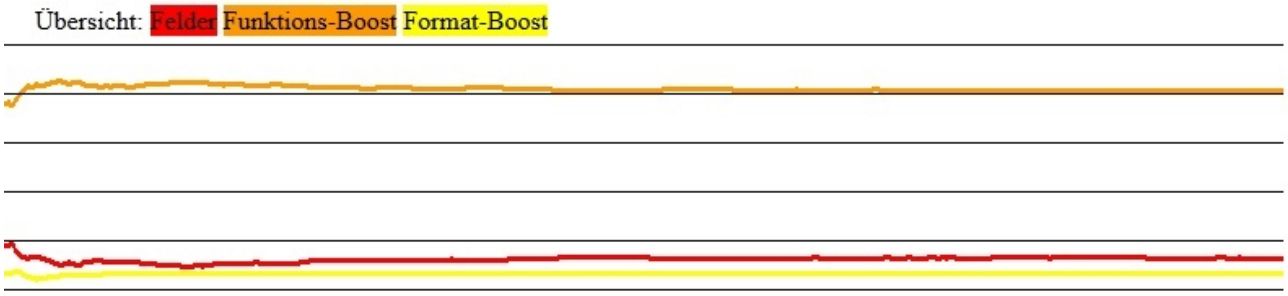
qf, ctrlnum^0
qf, normlink^0
qf, id^0

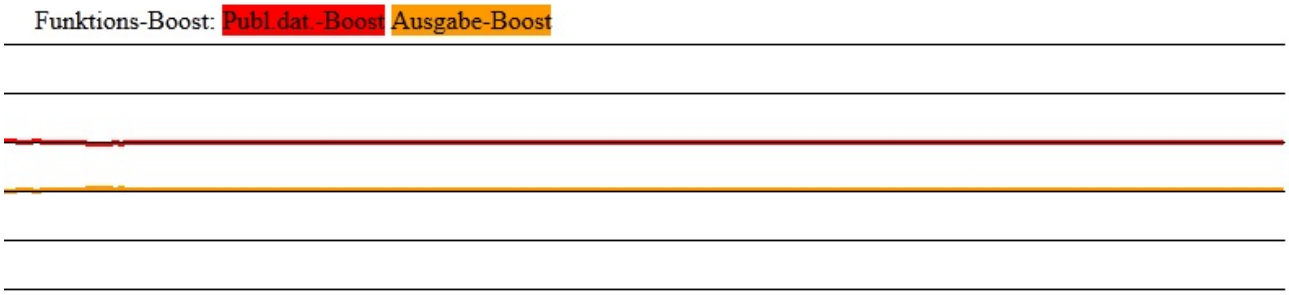
Formatboosting (boosting-query):
bq, format:eJournal^25
bq, format:Journal^30
bq, format:Book^200

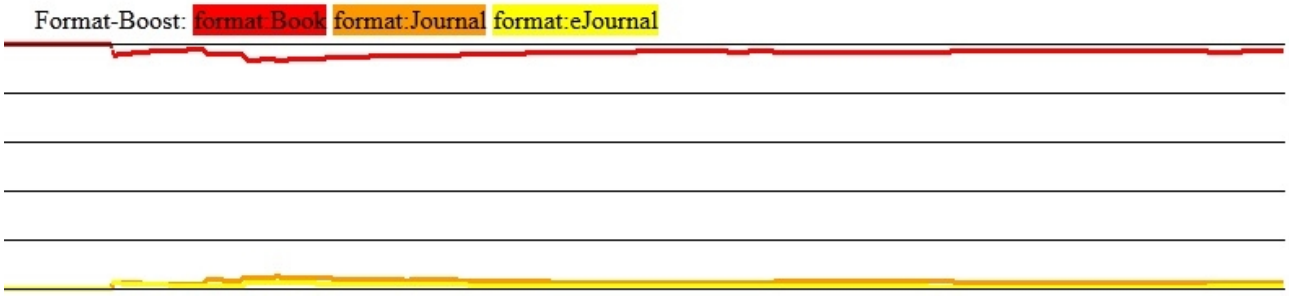
boostingfunktionen (Veröffentlichungsjahr, aktuelle Ausgaben):
bf, if(exists(query({!v=format:Journal})),737,ord(publishDateSort))^17
bf, if(exists(query({!v=edition:*})),ord(publishDateSort),737)^8

Tiebreaker:
tie, tie^0.1

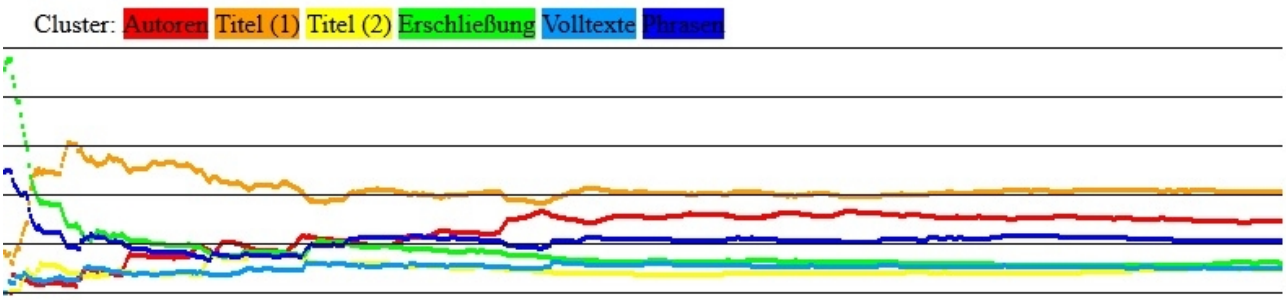
Die Übersicht sieht bei dieser Konfiguration Änderungen wie folgt aus:



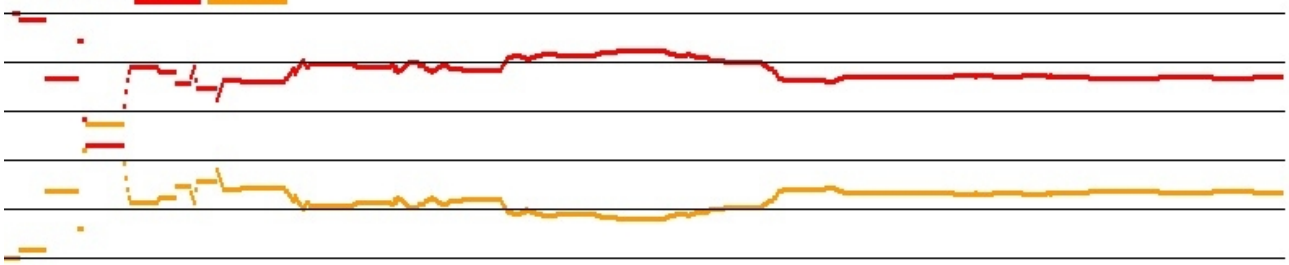




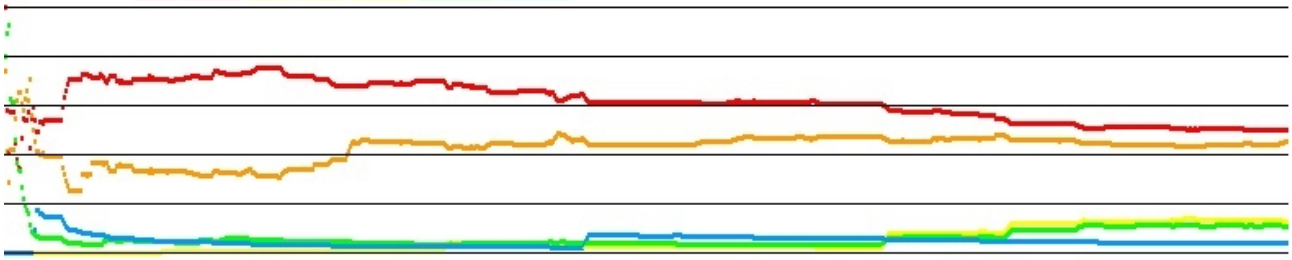
die Cluster und die Felder:



Autoren: author author2

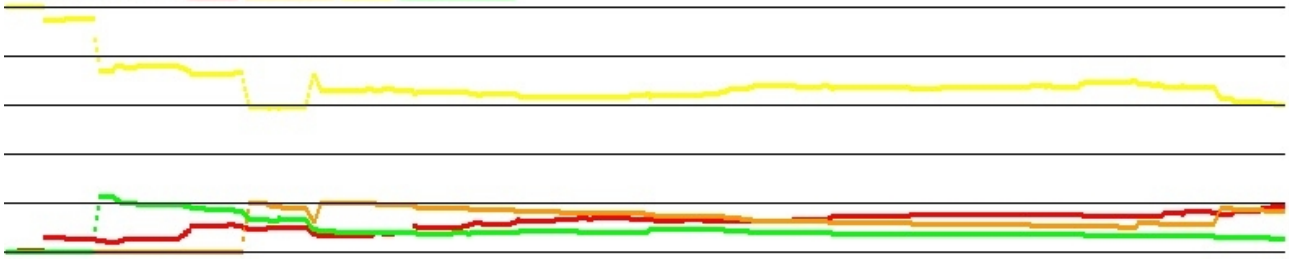


Titel (1): title_short title_auth series series2 journal

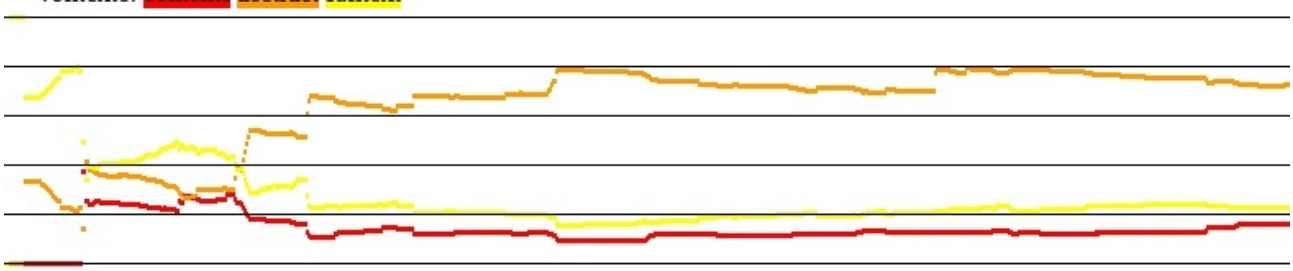


Titel (2): title_alt title_new title_old

Erschließung: class bklname topic geographic



Volltexte: contents abstract fulltext



Auch die einzelnen Cluster ergeben jetzt ein Bild, das vor dem Hintergrund der angestellten Vorüberlegungen dem entspricht, was an Feldbewertungen wünschenswert ist. Konkret sieht dies für das Beispiel „Alster Hamburg“ für den ersten Platz so aus:

26.27436 = (MATCH) sum of:
3.0974207 = (MATCH) sum of:
1.9671911 = (MATCH) max plus 0.1 times others of:
1.4058923 = (MATCH) weight(title_auth:alster^14.0 in 228168) [DefaultSimilarity], result of:
...
1.8266019 = (MATCH) weight(title_alt:alster^12.0 in 228168) [DefaultSimilarity], result of:
...
1.1302297 = (MATCH) max plus 0.1 times others of:
0.5145316 = (MATCH) weight(geographic:hamburg^15.0 in 228168) [DefaultSimilarity], result of:
...
0.49190438 = (MATCH) weight(title_auth:hamburg^14.0 in 228168) [DefaultSimilarity], result of:
...
0.28228942 = (MATCH) weight(class:hamburg^30.0 in 228168) [DefaultSimilarity], result of:
...
0.8152299 = (MATCH) weight(title_alt:hamburg^12.0 in 228168) [DefaultSimilarity], result of:
...
0.91983414 = (MATCH) weight(title_short:hamburg^12.0 in 228168) [DefaultSimilarity], result of:
...
0.9923673 = (MATCH) weight(format:Journal^30.0 in 228168) [DefaultSimilarity], result of:
...
15.085509 = (MATCH) Functionquery(if(exists(query(format:Journal,def=0.0)),const(737),ord(publishDateSort))), product of:
737.0 = if(exists(query(format:Journal,def=0.0)=5.2414784),const(737),ord(publishDateSort)=683)
...
7.0990634 = (MATCH) Functionquery(if(exists(query(edition:,def=0.0)),ord(publishDateSort),const(737))), product of:*
737.0 = if(exists(query(edition:,def=0.0)=0.0),ord(publishDateSort)=683,const(737))*
...

Für das Wort alster ist also das Feld ***title_alt*** mit einem Anteil von 0,9, das Feld ***title_auth*** mit 0,1 eingegangen, für das Wort hamburg ***title_short*** mit 0,9, ***title_alt***, ***class***, ***title_auth*** und ***geographic*** mit je 0,1. Dazu kommen dann die boostings in allen drei Bereichen. Zu sehen ist auch gut die doppelte Aufwertung, weil dieses Medium zum einen als Zeitschrift, zum anderen als „Medium ohne weitere Ausgaben“ als aktuell gewertet wird (Rang 737 statt 683).

Fazit

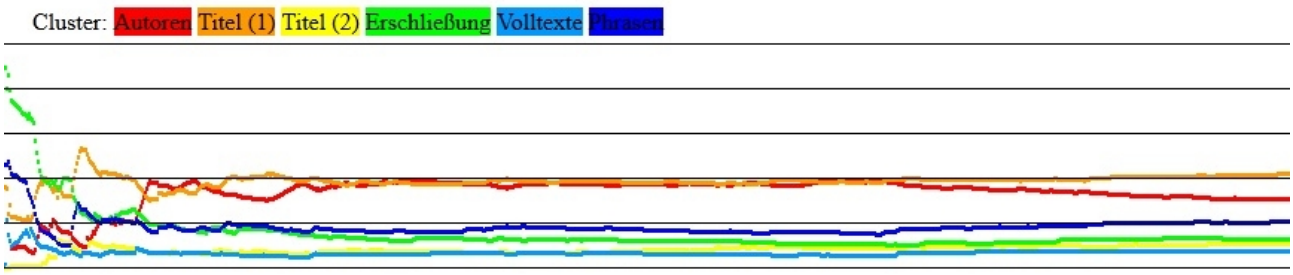
Mit der aktuellen Konfiguration führt eine leere Suche zu folgendem Ergebnis:

> ***Rang:1** PPN:788889680 Handbook of Human Resources Management (Book, 2016)*
> ***Rang:2** PPN:802100163 Operations management : Processes and supply chains (Book, 2016)*
> ***Rang:3** PPN:816726515 Teufelswerk oder Himmelsgeschenk? ... (Book, 2016)*
> ***Rang:4** PPN:823734730 Investing in China Through Shanghai Free Trade Zone (Book, 2016)*
> ***Rang:5** PPN:797284907 Project management achieving competitive advantage (Book, 2016)*
> ***Rang:6** PPN:799528900 Financial management core concepts (Book, 2016)*
> ***Rang:7** PPN:722134363 Wissenschaftstheorie und wissenschaftliches Arbeiten ... (Book, 2016)*
> ***Rang:8** PPN:79696727X Business a changing world (Book, 2016)*
> ***Rang:9** PPN:798452102 Understanding financial statements (Book, 2016)*
> ***Rang:10** PPN:819481149 Slavery in the circuit of sugar Martinique and ... (Book, 2016)*

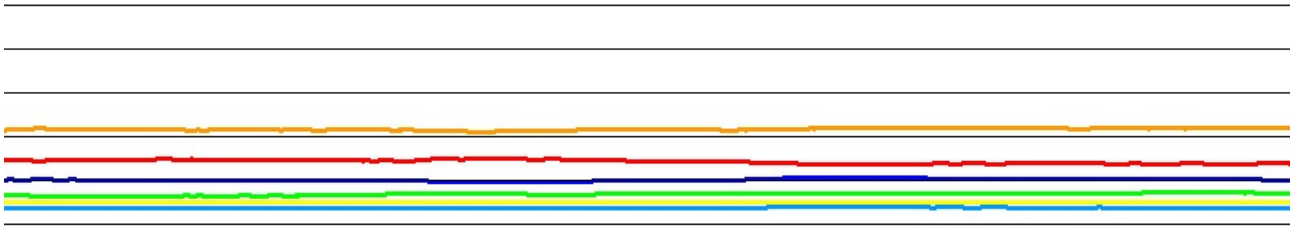
Wenn jetzt der Parameter für das boosting von E-Zeitschriften von 25 sukzessive erhöht wird, zeigt sich dasselbe Ergebnis für die Werte bis 28,95. Bei einem boosting-Wert von 29 zeigt sich aber etwas völlig anderes:

> ***Rang:1** PPN:816694591 Journal of service theory and practice (eJournal, 2015)*
> ***Rang:2** PPN:819365017 Statistical yearbook Curaçao (eJournal, 2015)*
> ***Rang:3** PPN:818042699 Hohenheim discussion papers in business, ... (eJournal, 2015)*
> ***Rang:4** PPN:826235999 Prego das Kulturmagazin von Edel (eJournal, 2015)*
> ***Rang:5** PPN:826757251 Diskussionspapiere / Hannover Center of Finance e.V (eJournal, 2015)*
> ***Rang:6** PPN:815915756 Journal of Aquaculture Engineering and Fisheries ... (eJournal, 2015)*
> ***Rang:7** PPN:815917015 Ocular oncology and pathology (eJournal, 2015)*
> ***Rang:8** PPN:817363181 Nuclear materials and energy (eJournal, 2015)*
> ***Rang:9** PPN:818041668 Heidelberger Zeitschrift für Iranistik (eJournal, 2015)*
> ***Rang:10** PPN:818041994 BMC nutrition (eJournal, 2015)*

Da scheint wieder so ein Schwellwert erreicht zu sein, der die gesamte Top-10-Liste beeinflusst. Das zeigt, dass es unmöglich ist, auf der Grundlage von Einzelfallbetrachtungen ein Ranking zu bewerten, geschweige denn die Parameter sinnvoll zu setzen. Anders bei einer statistischen Betrachtung: Die schwankenden und je nach Stichprobe auch immer anderen Verläufe der Kurven an den linken Enden der Diagramme spiegeln diese Situation der Einzelfallbetrachtungen wider. Zu den rechten Enden aber werden die Kurvenverläufe nicht nur glatter, sie nähern sich auch sichtbar bestimmten Werten und sind auch über verschiedene Stichprobenziehungen hinweg verlässlich. In diesem Artikel sind nach und nach zehn verschiedene Stichprobenziehungen verwendet worden, ohne die Konsistenz der hier gemachten Überlegungen zu beeinträchtigen.



Der linke Teil des Clusterdiagramms bei 2000 Top-5-Abfragen. Die Kurven nähern sich nach rechts zunehmend konstanten Werten an.



Der rechte Teil desselben Diagramms: Die Kurven sind annähernd konstant.

Dennoch sind die Werte, die bei den statistischen Betrachtungen erhoben werden, nicht absolut. Sie sind in den Diagrammdarstellungen daher auch so umgesetzt, dass sie auf die Höhe des Diagramms normiert werden und zwar für jede Auswertung. Eine solche Betrachtung passt aber sehr gut dazu, dass Relevanz eben nicht etwas Objektives oder auch nur Objektivierbares ist. Daher sind die Diagramme, die die Ergebnisse von statistischen Analysen darstellen, Werkzeuge, die einer heuristischen Betrachtung dienen. Ähnlich wie bei einer Einzelfallbetrachtung, bei der gesehen werden kann, ob die betrachteten Ergebnisse plausibel erscheinen, kann dies hier mit den statistischen Analysen geschehen. Auf diese Weise wird die Komplexität der der Relevanzbewertung zu Grunde liegenden Komplexität und die quasi diskontinuierlichen Effekte, die damit zusammenhängen, befriedigend bewältigt. Nichts desto trotz ist es gerade für eine gute heuristische Einschätzung der Ergebnisdiagramme von entscheidendem Vorteil, die der Bewertung zu Grunde liegenden bibliothekarischen und technischen Mechanismen zu kennen.

Die Einstellung der Relevanz-Bewertung von Suchergebnissen ist ein nie abgeschlossenes Projekt. Die hier angestellten Überlegungen werden immer wieder überprüft, ergänzt oder korrigiert, es werden weitere Betrachtungen angestellt und weitere Analysemöglichkeiten erschlossen. Diese Dynamik ist letztlich auch ein zentraler Aspekt eines suchmaschinenbasierten Katalogs. So wird mit Sicherheit das Phrasenboosten noch etwas herunter geregelt werden, weil es die betroffenen Felder bei Einterm-Suchen im Ranking schwächt und damit mit einer eher unerwünschten Verschiebung der Ränge einhergeht.

Wie es weiter geht

Hier wurde das Thema Relevanz-Bewertung und Relevanz-Sortierung im Wesentlichen von der Seite eines Nutzers eines vorhandenen solr-Index betrachtet. Obendrein wurde hier auch nur die allgemeine Suche exemplarisch konfiguriert und auch die Seite der Suchenden, also derjenigen, für die der Katalog bereitgestellt wird, wurde nur wenig berücksichtigt. Im weiteren wäre es also wichtig, auch die - hier teilweise angesprochenen - Optimierungen des Index zu konzipieren, beispielsweise die Zuordnung des Index zum solr-marc.

Es ist auch notwendig, das hier skizzierte Vorgehen auf spezialisierte Suchen, beispielsweise Titelsuchen oder Autorensuchen zu erweitern. Schließlich sollten die ermittelten Einstellungen auch in Hinblick auf das Nutzungsverhalten evaluiert werden. So ließe sich untersuchen, welche Treffer (insbesondere welche Ränge) Nutzer nach ihren Suchen nutzen, um eine Qualifizierung der Rangfolgen auch aus Nutzersicht zu haben.

Insbesondere macht es Sinn, „known-entity“- bzw. „known-item“-Suchen zu identifizieren und anders zu behandeln als die anderen Suchen. Die Arbeit von Imke Rulik (2015) greift dieses Thema auf und zeigt hierfür eine vielversprechende Lösung.

Es gibt auch noch weitere Themen, die in diesem Zusammenhang erörtert werden können und sollten: Sei es das multiplikative boosting oder Negativbewertungen von unerwünschten Einträgen. Zu der Relevanz-Sortierung gehört schließlich auch eine passende Facettierung. Die zu ermitteln wird mit ähnlichen Betrachtungen verbunden sein wie die hier vorgenommenen.

Quellen und Links (nicht nach Relevanz sortiert)

Suchmaschinen und Bibliothekskataloge

Timo Borst, Dirk Lewandowski: LibRank Projektseite, 2015. http://www.librank.info/; s.a. http://www.zbw.eu/de/ueber-uns/aktuelles/meldung/news/forschung-zu-intelligenteren-suchergebnissen-in-bibliothekskatalogen/

Imke Rulik: „Known-Item-Suchanfragen im Discoverysystem beluga: Retrievaleffektivität und Empfehlungen“. Bachelor-Arbeit, Hochschule für Angewandte Wissenschaften Hamburg, Department für Information, 2015. http://edoc.sub.uni-hamburg.de/haw/volltexte/2015/3023/pdf/Rulik_Imke_141202.pdf

Till Kinstler: „Hacking VuFind into german libraries“, 2009/2014. https://vufind.org/wiki/hacking_vufind_into_german_libraries

Stefan Winkler: „Relevance Ranking Revisited. Heterogene Datenquellen in VuFind“. Vortragsfolien für 15. BSZ-Kolloquium 30.9.2014. Bibliotheklsservice-Zentrum Baden-Württemberg (BSZ). http://swop.bsz-bw.de/volltexte/2014/1168/pdf/09_relevance_ranking_bsz_kolloq_2014.pdf

Annette Langenstein, Leonhard Maylein: „Neues vom Relevanz-Ranking im HEIDI-Katalog der Universität Heidelberg“. bit online 16, 2013, Nr.3. http://www.b-i-t-online.de/heft/2013-03-fachbeitrag-maylein.pdf

Dirk Lewandowski, Klaus Tochtermann: „LibRank: Neue Formen der Relevanz-Sortierung in bibliothekarischen Informationssystemen; Neuantrag auf Gewährung einer Sachbeihilfe“. Unveröff. Typoskript, 2013. http://gepris.dfg.de/gepris/projekt/246011126

Hajo Seng: „Relevance Ranking und querytype Detection“ Vortragsfolien; http://www.tub.tuhh.de/vufind-anwendertreffen/files/2013/08/Relevance-Ranking.pdf

Hajo Seng: „Relevance-Ranking bei Beluga“. Staats- und Universitätsbibliothek Hamburg, 2013. http://beluga-blog.sub.uni-hamburg.de/blog/2013/12/02/relevance-ranking-bei-beluga/

Gerald Steilen: GBV-Zentral Vortragsfolien; http://www.tub.tu-harburg.de/vufind-anwendertreffen/files/2013/09/GBV_Zentral.pdf

GBV-Zentral, Gerald Steilen, VZG Aktuell 2013 Ausgabe 3, S.20-21. https://www.gbv.de/Verbundzentrale/Publikationen/broschueren/vzg-aktuell/VZG_Aktuell_2013_03.pdf/at_download/file

Otto Oberhauser: „Relevance Ranking in den Online-Katalogen der nächsten Generation“, in: Mitteilungen der Vereinigung Österreichischer Bibliothekarinnen & Bibliothekare, 63, 2010, S. 25-37.

Ursula Schulz: „Kriterien für Suchmaschinen“, 2010. http://www.bui.fh-hamburg.de/pers/ursula.schulz/webusability/suchma.html

Lewandowski, Dirk: „Ranking library materials“, in Library Hi Tech, 27, 2009, S. 584-593

Scott Huffman: „Search evaluation at Google“, 2008. http://googleblog.blogspot.de/2008/09/search-evaluation-at-google.html

Tony Boston, Alison Dellit: „Relevance ranking of results from MARC-based catalogues: from guidelines to implementation exploiting structured metadata“, 2007. http://ll01.nla.gov.au/InfoOnline2007Paper.html

Dirk Lewandowski: „Web Information Retrieval. Technologien zur Informationssuche im Internet“. Reihe Informationswissenschaft der DGI, Hrsg.: Marlies Ockenfeld, Band 7. Deutsche Gesellschaft für Informationswissenschaft und Informationspraxis e.V.. DGI-Schrift, Frankfurt, Main, 2005

Gerard Salton: „Automatic text processing: the transformation, analysis, and retrieval of information by computer“. Addison-Wesley Publ., Boston, 1989

Datenschema-Dokumentationen

findex-Konfiguration der VZG: https://github.com/gbv/findex-config/tree/master/SolrCloud/solr_config

VuFind Index-Schema: https://vufind.org/wiki/index_schema

Zum solr-marc Metadaten-Schema: https://code.google.com/p/solrmarc/wiki/IndexProperties

Deutsche Nationalbibliothek: „Feldbeschreibung der Titeldaten der Deutschen Nationalbibliothek und der Zeitschriftendatenbank im Format MARC 21“, 2014. nbn-resolving.de/urn:nbn:de:101-2014040221

Deutsche Nationalbibliothek: „Konkordanz Pica - MARC21 für die bibliographischen Titeldaten der Deutschen Nationalbibliothek und der Zeitschriftendatenbank“, 2013. http://d-nb.info/103173869X/34

Rafał Kuć, Marek Rogoziński: „A tool for finding out why the documents are where they are in the results list“. http://explain.solr.pl/

Software-Dokumentationen

Markus Klose & Daniel Wrigley: „Einführung in Apache Solr“. O'Reilly, 2014.

Solr 4.6 Referenz: http://km.snippetinfo.net/sysdata/doc/7/7dd4f708e8a242e1/pdf.pdf

Analyser und Tokenizer: https://cwiki.apache.org/confluence/display/solr/Understanding+Analyzers%2C+Tokenizers%2C+and+Filters

tf-idf-Implementierung in solr: lucene.apache.org: „org.apache.lucene.search: Similarity“. http://lucene.apache.org/core/3_5_0/api/ core/org/apache/lucene/search/Similarity.html, 2011.

Snowball-Porter-stemming-Algorithmus: http://snowball.tartarus.org/algorithms/english/stemmer.html