Neues vom Relevanz-Ranking im HEIDI-Katalog der Universitätsbibliothek Heidelberg

Leonhard Maylein und Annette Langenstein

Das Relevanz-Ranking im Katalog der Universitätsbibliothek Heidelberg (HEIDI), bereits 2009 in einem Beitrag in dieser Zeitschrift beschrieben, wurde in den letzten Jahren durch neue Entwicklungen und Methoden stark verbessert. Der Aufsatz beschreibt die Realisierung der bisherigen Rankingmaßnahmen unter der neu eingesetzten Suchmaschinenplattform SOLR. Weiter werden verschiedene neue Möglichkeiten für Rankinganpassungen unter SOLR sowie deren Einsatz im HEIDI-Katalog dargestellt.

The relevance ranking in the catalogue of the Heidelberg University Library (HEIDI), which was described in an article in this magazine in 2009, has been significantly improved through new advancements and methods during the past few years. The essay describes the realization of the ranking measures taken so far under new newly-installed search engine platform SOLR. In addition, several new opportunities for adjustment of ranking under SOLAR are presented, as well as their use in the HEIDI catalogue.

Im Jahr 2009 haben die Autoren die im selbstentwickelten Katalog der Universitätsbibliothek Heidelberg (HEIDI-Katalog) getroffenen Maßnahmen zur Verbesserung des Relevanz-Rankings beschrieben (1). An der grundsätzlichen Situation, dass Detailinformationen zu Rankingverfahren in den verschiedenen Katalogen praktisch nicht publiziert werden, hat sich seither wenig geändert. Immer mehr Bibliotheken entschieden sich in den letzten Jahren für den Einsatz kommerzieller Suchindexe von Discovery-Systemen, um die Nachweissituation und die Recherchemöglichkeiten insbesondere für Aufsatzliteratur zu verbessern. Meist wird das zugehörige Recherchefrontend mit lizenziert und in der Regel wird dieses Frontend dann auch für die Recherche in den eigenen Katalogdaten genutzt. Die gängigen Systeme am Markt setzen ausgeklügelte Rankingverfahren ein, um der enormen Datenflut (in der Regel ist die Grenze zu über einer Milliarde Titelnachweisen längst überschritten) gerecht zu werden.

Leider behandeln die Anbieter die von ihnen getroffenen Rankingmaßnahmen noch immer als Betriebsgeheimnis. Nur selten erhält man als Kaufinteressent oder als Kunde auch nur Andeutungen zu den Verfahren, die sich im Verborgenen abspielen. So erfährt man von dem einen Anbieter, dass grundsätzlich auf die Einbeziehung der „inverse document frequency" (IDF)[1] verzichtet wird (ob gezwungenermaßen oder wie behauptet aus Überzeugung sei dahingestellt), oder von einem anderen Anbieter, dass in Abhängigkeit der gewählten Frontendsprache Titel in derselben Sprache höher gewichtet werden. Mehr, gar konkrete Gewichtungen, wird nicht preisgegeben.

Bibliotheken, die solche Discovery-Systeme nutzen, haben in der Regel wenig Möglichkeiten, das Ranking zu beeinflussen. Diese beschränken sich oft darin, die eigenen Katalogdaten im Vergleich zum ungleich größeren Pool mehr oder weniger qualitätvoller Titelaufnahmen zu bevorzugen.[2] Es sind Zweifel daran angebracht, ob sich Bibliotheken, die ihren Bestand und die Bedürfnisse ihrer Nutzer am besten kennen, blind auf die von den Herstellern der Discovery-Systeme entwickelten Rankingmethoden verlassen sollten (3, 2).

Auch die Universitätsbibliothek Heidelberg hat sich 2011 für die Nutzung eines Discovery Systems entschieden. Die Wahl fiel auf das Produkt Summon der Firma Serials Solutions. Allerdings setzt die Universitätsbibliothek Heidelberg weiterhin auf ihren bewährten HEIDI-Katalog. Die Daten des Summon-Index werden über eine Programmierschnittstelle (engl. application programming interface, API) genutzt und in einem zweiten Treffer-Reiter unter der Bezeichnung „Artikel & mehr..." präsentiert. Die lokalen Katalogdaten werden nicht in den Summon-Index importiert. Dies erlaubt es, für den ursprünglichen Katalogbestand weiterhin sehr spezifische Suchmöglichkeiten (Normdaten, spezifische Facetten[3] und vieles mehr) einzusetzen. Es erlaubt aber eben auch ein speziell an die lokalen Bedürfnisse (z.B. Fächerspektrum) angepasstes und jederzeit änderbares Ranking. Zugegebenermaßen erkauft man sich eine solche „Zwei-Reiter-Lösung" durch eine (zumindest scheinbar) schlechtere Nutzung der Daten des Discovery Systems. Ein Student, der nur mal schnell die Literaturliste für

1 Ein Wort, das nur in wenigen Datensätzen vorkommt, wird höher gewichtet als häufig vorkommende Wörter.

2 Im Falle von Primo ist das Ranking im lokalen Index individueller konfigurierbar (z.B. mit Hilfe von Fieldboosts). Im aggregierten Index (Primo Central) ist zumindest ein nutzerabhängiges Fachboosting möglich.

3 Zum Beispiel eine Facette zum geographischen Bezug (4).

seine Vorlesung im Katalog abarbeitet, ist eben häufig nicht an „Artikel & mehr…“ interessiert und klickt diesen Reiter erst gar nicht an. Dieser Nutzer stellt allerdings auch nicht unbedingt die Zielgruppe für die Aggregatordaten dar – vorausgesetzt, die in der Regel vorhandenen großen E-Book-Bestände sind bereits im „traditionellen“ Katalogbestand nachgewiesen und nicht nur über den Aggregator-Index verfügbar. Die geringere Nutzung der Nachweise vom Discovery System könnte auch so gedeutet werden, dass in vielen Fällen der Katalogbestand mit Print- und E-Book-Nachweisen ausreichend ist und die Nutzererwartung erfüllt. Nichtsdestotrotz ist die Entscheidung für eine „Zwei-Reiter-Lösung“ immer eine schwierige Abwägung, die sicherlich von Zeit zu Zeit neu vorgenommen werden muss.

Im Folgenden soll nun die Weiterentwicklung des Rankings in HEIDI in den letzten Jahren beschrieben werden. Seit der ersten Veröffentlichung der Rankingmaßnahmen (1) wurde der HEIDI-Katalog in vielen Bereichen weiterentwickelt. Eine auch für das Ranking wichtige Änderung war der Umstieg vom selbst programmierten RPC-Backend basierend auf dem Suchmaschinenframework Lucene auf die Software SOLR (7). Mit dem Open-Source-Produkt SOLR steht eine komplette und fertig einsetzbare Suchmaschinenplattform zur Verfügung, die ihrerseits auf Lucene basiert und eine Fülle von zusätzlichen Komponenten bietet. Mit deren Hilfe können praktisch alle modernen Suchmaschinenfeatures sehr rasch umgesetzt werden.[4] Einige dieser Komponenten, die für die Umsetzung von Rankingmaßnahmen geeignet sind, sollen hier vorgestellt werden.

Umsetzung der bisherigen Rankingverbesserungen mit SOLR

Die in (1) beschriebenen Rankingmaßnahmen wurden beim Umstieg auf SOLR zunächst ohne inhaltliche Anpassungen übernommen. Sie betreffen die (Nicht-) Berücksichtigung der Feldlänge sowie die Dokumenten-, Feld- und Phrasen-Boosts.

Feldlänge

Durch Codeanpassungen wurde die Berechnung der *lengthNorm* auch in SOLR unterdrückt, so dass die Feldlänge bei der Ermittlung der *fieldNorm* keine Rolle mehr spielt und die Score-Berechnung so der spezifischen Beschaffenheit von Katalogdaten eher Rechnung trägt (1).

In SOLR ist es möglich, eine eigene Similarity-Klasse zu definieren und durch einen entsprechenden Konfigurationseintrag in der Schemabeschreibung des Index die Standard-Similarity-Berechnung global oder für bestimmte Feldtypen zu überschreiben. Für HEIDI wurde eine Similarity-Klasse *UBHDSimilarity/UBHDSimilarityFactory* als Kopie der Klasse *DefaultSimilarity/DefaultSimilarityFactory* von SOLR erstellt und angepasst.

```
<similarity class=„de.uni_heidelberg.ub.heidi.solr.UBHDSimilarityFactory“ />
```

Abbildung 1: Überschreiben der Standard-Similarity-Berechnung durch eine eigene Klasse in der Konfigurationsdatei schema.xml.

Für das Suchfeld der Kataloganreicherungen (exttext) wurde hingegen entschieden, die Feldlänge zu berücksichtigen. Beim Inhalt dieses Feldes handelt es sich meist um klassische, im Vergleich zu Katalogdaten weniger strukturierte und weniger umfangreiche Textdokumente. Für solche Textdokumente ist das Default-Ranking von SOLR und Lucene ausgelegt. In langen Texten kommen Suchbegriffe in der Regel häufiger vor. Die daraus resultierende höhere Gewichtung des Suchfelds wird durch die Einbeziehung der Feldlänge ausgeglichen.

Der Einfachheit halber wurde die entsprechende Fallunterscheidung fest in der Klasse *UBHDSimilarity* codiert (Abbildung 2). Die ab SOLR 4.0 vorgesehene Lösung, dies flexibel über die Feldtypendefinition in der Schemabeschreibung des Index konfigurieren zu können (8) wurde vorerst nicht realisiert, da sie deutlich aufwändiger ist. Notwendig wäre eine Anpassung der mitgelieferten SchemaSimilarityFactory. Allerdings fehlen in dieser einige Implementierungen der DefaultSimilarity (z.B. die Methoden *coord* und *queryNorm*) und müssten zuerst ergänzt werden.

```
public void computeNorm(FieldInvertState state, Norm norm) {
  boolean exttext=state.getName().equals(„exttext“);
  final int numTerms;
  if (discountOverlaps)
   numTerms = state.getLength() - state.getNumOverlap();
  else
   numTerms = state.getLength();
// Ermittlung der fieldNorm: Feldlänge nur bei Feld exttext berücksichtigen
  norm.setByte(encodeNormValue(
    state.getBoost()
    * ((float) (1.0 /
      (exttext?Math.sqrt(numTerms):1.0)
    ))
  ));
 }
```

Abbildung 2: Quellcodeanpassung SOLR 4.0.

4 Einen Überblick über die Features von SOLR bieten beispielsweise (5) und (6).

Beim Einsatz der SOLR-Version 4.0 ist zu beachten, dass ein Bug bei der Berechnung der *fieldNorm* für Felder, die durch die copyField-Anweisung aus mehreren anderen Feldern bestückt werden, zu Fehlern führt, die sämtliche Rankinganstrengungen zunichte machen.[5] Im HEIDI-Katalog wird beispielsweise das Freitext-Feld, das die Inhalte der Felder *Titel, Schlagwort, ISBN/ISSN, Autor, Körperschaft, Erscheinungsjahr, Verlag* und *Verbundidentnummer* umfasst, mit der copyField-Funktion zusammengesetzt.

Dokumenten-Boosts

Bei der Ermittlung der statischen Dokumenten-Boosts werden im HEIDI-Katalog nach wie vor

- Bandsätze zu Zeitungen und Zeitschriften niedriger,
- neuere Auflagen höher,
- neuere Erscheinungsjahre höher,
- Online-Ressourcen höher und
- Titel mit vielen besitzenden Bibliotheken auf dem Campus höher gewichtet.

Die Auswahl dieser Faktoren und auch die in (1) beschriebene Gewichtung hat sich in den vergangenen Jahren bewährt und wurde dementsprechend nicht verändert.

Angedacht ist, die aus der Bilddatenbank HeidICON der Universitätsbibliothek Heidelberg generierten Titelsätze zu frei verfügbaren Bildern grundsätzlich niedriger zu gewichten, da sie als Online-Ressourcen gegenüber einem Großteil des übrigen Bestands bevorzugt werden. Allerdings können Nutzer diese Titel, wenn sie unerwünscht sind, auch recht einfach über die Funktion „Treffer einschränken – ohne Bilder“ aus der Treffermenge entfernen.

Feld-Boosts und Phrasen-Boosts bei der Suche

Da es für den MultiFieldQueryParser von Lucene keine direkte Entsprechung in SOLR gibt, wurden bei der Migration auf SOLR die Feld-Boosts zunächst durch eine Vorverarbeitung der Suchanfrage im Katalog-Frontend realisiert. Die Suchanfrage wird dort geparsed und Suchterme, die keinem spezifischen Suchfeld zugeordnet sind, werden durch eine entsprechende Oder-Verknüpfung über die Suchfelder

- *freitext*[6] (Boost: 1.0),
- *exttext*[7] (Boost: 0.5) und
- *1w*[8] (Boost: 2.0) ersetzt.

Die Anzahl der Suchfelder, in denen parallel und mit unterschiedlichen Boosts gesucht wird, hat sich in der Zwischenzeit allerdings erhöht (siehe Abschnitt „Neue Rankingmaßnahmen“).

Die Bevorzugung von Titeln, welche die eingegebene Suche als exakte oder ähnliche Phrase beinhalten, wurde bei der Migration auf SOLR zunächst unverändert beibehalten und – wie vorher auch – über eine Ergänzung der ursprünglichen Suchanfrage im Frontend realisiert.

Die Realisierung dieser Feld- und Phrasen-Boosts ist in HEIDI inzwischen (zumindest teilweise) durch den Einsatz des EDisMAX Query Parser abgelöst. Dieser bietet darüber hinaus weitere interessante Möglichkeiten für Rankinganpassungen, die im Abschnitt „EDisMax Query Parser“ beschrieben werden.

Neue Rankingmaßnahmen

Stemming

Einer der großen Vorteile von SOLR gegenüber der vorher eingesetzten Lucene-Lösung ist die einfache Parametrierbarkeit. Weitere Funktionen, Felder oder Analyzer können durch einfaches Editieren der Konfigurationsdateien genutzt werden. Direkt beim Umstieg auf SOLR wurde ein Stemming für die wichtigsten Suchfelder *Titel (ti)* und *Schlagwort (sw)* eingerichtet. Dies wurde so realisiert, dass die originalen Suchfelder ohne Stemming erhalten bleiben und die Inhalte per copyField-Befehl zusätzlich in entsprechende Felder mit Stemming (*ti_stem* und *sw_stem*) indexiert werden. Für das Stemming wird die ‚SnowballPorterFilterFactory‘ mit der Spracheinstellung ‚German‘ verwendet.

Im Regelfall werden bei einer Freitextsuche die beiden Felder *ti_stem* und *sw_stem* mit durchsucht, allerdings mit einem relativ geringen Boost-Faktor. Die beschriebene Feldaufteilung der Freitextsuche ändert sich dadurch wie folgt:

- *freitext*: Boost 1.0
- exttext: Boost 0.5
- 1w: Boost 2.0
- *ti_stem*: Boost 0.2
- *sw_stem*: Boost 0.2

Eine Suche nur im Feld *Titel* (aus der Feldsuche heraus oder bei einer ebenfalls möglichen Feldangabe in der einfachen Suche) wird automatisch so erweitert, dass (mit niedrigerem Boost-Faktor und Oder-Verknüpfung) auch das Feld *ti_stem* durchsucht wird:

- *ti*: Boost 1.0
- *1w*: Boost 2.0
- *ti_stem*: Boost: 0.2

5 https://issues.apache.org/jira/browse/SOLR-3981 [17. Januar 2013]

6 Umfasst die Felder: *Titel, Schlagwort, ISBN/ISSN, Autor, Körperschaft, Erscheinungsjahr, Verlag und Verbundidentnummer*.

7 Beinhaltet die Daten aus den Kataloganreicherungen (z.B. Inhaltsverzeichnisse, Klappentexte etc.).

8 Suchfeld für Einworttitel.

Abbildung 3: Facette „nur exakte Begriffe“.

Entsprechendes gilt für die Suche im Schlagwortfeld (Felder *sw* und *sw_stem*).

Stemming-Verfahren erhöhen generell die Trefferменge und führen zwangsläufig zu falschen positiven Treffern. Durch das angegebene Verfahren ist jedoch sichergestellt, dass Titel, die nur über das Stemming gefunden werden, in der Trefferliste weit hinten landen. Katalognutzer, die dennoch lieber auf das Stemming verzichten, können die zusätzliche Suche in den Felder *ti_stem* und *sw_stem* über die Facette „nur exakte Begriffe“ abschalten. Dies ist insbesondere auch dann wichtig, wenn man statt des Rankings eine Sortierung nach E-Jahr oder Autor/Titel wählt. Hier wird der Nutzer per Suchtipp explizit auf diese Möglichkeit hingewiesen (siehe Abbildung 3).

Kompositazerlegung

Seit Mitte 2012 nutzt HEIDI die Software Lingo (9), um bei der Indexierung deutscher Titel eine Kompositazerlegung durchzuführen. Die Zerlegung beschränkt sich zurzeit noch auf die Titel- und die Schlagworteinträge. Die erzeugten Teilwörter werden zusätzlich zu den Komposita indexiert und der Einfachheit halber mit in den Stemming-Feldern *ti_stem* und *sw_stem* abgelegt. Sie erhalten damit bei einer Suche die gleichen Gewichte wie die Wortstämme. Die Einschränkung der Suche auf „exakte Begriffe“ ignoriert (wie gewünscht) die Suche in den Teilwörtern.

Die Vorgehensweise, Teilwörter als Synonym zum Kompositum zu indexieren, wurde verworfen. Grundsätzlich ist es bei Lucene und SOLR zwar möglich,

dass als Synonyme indexierte Begriffe eine unterschiedliche Wortanzahl aufweisen, jedoch wird dies bei Phrasensuchen nicht korrekt berücksichtigt. So ist es nicht möglich, anzugeben, dass mehrere Terme zusammen die Position eines einzelnen Terms besetzen (14), ein grundsätzliches Problem auch bei der Zerlegung von CamelCase-oder Bindestrich-Wörtern durch den WordDelimiterFilter (10). Zudem würde diese Vorgehensweise die Einschränkung der Suche auf „exakte Begriffe“ sowie unterschiedliche Gewichtungen erschweren.

Eine Kompositazerlegung auf Seiten der Suchanfrage ist bislang nicht realisiert. Dies ist ungleich schwerer umzusetzen, da Lingo häufig mehrere Zerlegungsmöglichkeiten anbietet. Hieraus eine halbwegs korrekte Suchanfrage zu konstruieren ist kaum möglich. Eine Unterstützung der Meinten-Sie-Funktion durch die Kompositazerlegung ist allerdings in Planung.

Grundsätzlich muss festgestellt werden, dass der Nutzen der Kompositazerlegung im Vergleich zum betriebenen Aufwand einigermaßen überschaubar ist. Um bestimmte Fachtermini, die sich in vielen Fächern aus Einzelwörtern der verschiedensten Sprachen zusammensetzen (man denke nur an die Fächer Chemie oder Medizin mit Wortzusammensetzungen aus lateinischen, griechischen und deutschen Begriffen), müssen die von Lingo mitgelieferten Wörterbücher stark angepasst werden. Zudem sind häufig die einzelnen Teilworte bereits in den Verweisungsformen der vergebenen Schlagwörter enthalten.

Es finden sich aber durchaus Fälle, wo die Kompositazerlegung zu den gewünschten Treffern führt, so findet eine Suchanfrage nach „Zoll“ auch die „Zollordnung“ oder die Suche nach „Bundesland“ findet auch den Titel mit dem Begriff „Flächenbundesland“.

Synonyme

Mit dem Umstieg auf SOLR wurde der HEIDI-Katalog auch um ein Wörterbuch erweitert, das die bei der Indexierung gefundenen Terme um Synonyme erweitert.[9]

Das Synonymwörterbuch des HEIDI-Katalogs umfasst insbesondere Wortvarianten in alter und neuer Rechtschreibung, aus der Gemeinsamen Normdatei (GND) generierte Transkriptionsvarianten von Personennamen,[10] Gegenüberstellung von chinesischen Schriftzeichen in der traditionellen, der vereinfachten und in der japanischen Form sowie einzelne Spezialfälle (z.B. §, paragraph, paragraf).

Für die Synonymverarbeitung wurde auf die Standardfunktionalität von SOLR zurückgegriffen. Synonyme werden daher in HEIDI gleichrangig zu den originalen Begriffen indexiert und bei der Suchanfrage gewichtet.

EDisMAX Query Parser

Feld- und Phrasenboosting

Mit dem Extended DisMAX (EDisMAX) Query Parser (11) bietet SOLR ein (mit der Version 4.0 halbwegs ausgereiftes) Verfahren, das oben beschriebene Feld- und Phrasenboosting elegant und ohne großen Aufwand abzubilden.

Wie bereits aus dem Namen ersichtlich, stellt EDisMAX eine Erweiterung des DisMAX Query Parsers dar. Vereinfacht gesprochen, erlaubt es der DisMAX Query Parser, Nutzeranfragen auf verschieden gewichtete, über den Parameter qf (= Query Fields) definierte Suchfelder zu verteilen. Pro definiertem Suchfeld und Suchterm wird eine Teilabfrage gebildet. Diese Teilabfragen werden per Oder-Operator verknüpft. Beim Ranking wird im einfachsten Fall jeweils der höchste Score einer dieser Teilabfrage als Score für das gesamte Trefferdokument herangezogen (daher auch das „MAX“ in der Bezeichnung). Über den Parameter tie (= Tie breaker) ist es möglich, auch die in den übrigen Suchfeldern erzielten Scores in die Berechnung einzubeziehen.

Die Berechnung richtet sich dabei nach folgender Formel (12):

```
(score of matching clause with the highest score) +
((tie paramenter) * (scores of any other matching clauses))
```

Abbildung 4: Scoreberechnung im EDisMAX Query-Parser.

DisMax erlaubt es zudem über die Parameter pf (= Phrase Fields) und ps (Phrase Slop), innerhalb der Menge der Treffer diejenigen zu bevorzugen, bei denen die Suchanfrage als Phrase (ggf. unter Berücksichtigung einer über den Slop-Faktor definierten Ungenauigkeit) vorkommt.

Im Gegensatz zu dem in (1) verwendeten MultiField-QueryParser von Lucene ist der DisMAX Query Parser jedoch nicht in der Lage, explizite Einschränkungen von einzelnen Suchtermen auf bestimmte Suchfelder sowie bool‘sche Operatoren und Klammerungen in der Benutzeranfrage zu erkennen und entsprechend zu behandeln. Dies war ein Grund für die Entwicklung des Extended DisMAX Query Parsers. EDisMAX sorgt

9 Die Entscheidung für die Synonymerweiterung bei der Indexierung fiel aufgrund der Tatsache, dass zum damaligen Zeitpunkt das Trefferhighlighting von SOLR bei der Synoymerweiterung auf Seiten der Suchanfrage fehlerhaft war. Die Erweiterung der Liste (ein eher seltenes Ereignis) erfordert daher einen kompletten Indexneuaufbau.

10 Da diese auch in den Normdaten berücksichtigt sind, spielen diese Synonyme nur bei Treffern in den Titelstichwörtern eine Rolle.

u.a. dafür, dass die „Query Fields“ für bereits in der Anfrage einem speziellen Suchfeld zugewiesenen Terme nicht ausgewertet werden.

EDisMAX bietet zudem über die Parameter pf2 (= Phrase bigram fields), ps2 (= Phrase bigram slop), pf3 (= Phrase trigram fields) und ps3 (= Phrase trigram slop) eine Verfeinerung des Phrasen-Boostings.

Im HEIDI-Katalog wird der EDisMAX Query Parser erst seit der SOLR-Version 4.0 genutzt, da dieser in den Vorversionen noch mit reichlich Fehlern behaftet war. Bezüglich des Feld- und Phrasenboostings werden derzeit folgende Werte verwendet:[11]

- **qf:** freitext exttext^0.5 1w^2 ti_stem^0.2 sw_stem^0.2
- **pf:** freitext^6 exttext^3 ti_stem^1.2 sw_stem^1.2
- **pf2:** freitext^2 exttext
- **pf3:** freitext^4 exttext^1.5
- **tie:** 0.1

Bei der exakten Suche entfallen bei qf und pf die Stemming-Felder.

Boostfunktionen (z.B. Fachboosting)

Die Möglichkeiten von EDisMAX gehen jedoch weit über das Feld- und Phrasenboosting hinaus.

Der Parameter bq (= Boost Query) erlaubt es, eine Teilanfrage zu definieren, deren Zweck es ist, den zur Suchanfrage gefundenen Dokumenten zu einem höheren Score zu verhelfen.

Über die Parameter bf (= Boost Function, additive) und boost (= Boost Function, multiplicative) eröffnen sich eine Fülle von Möglichkeiten, das Ranking von SOLR zu beeinflussen. Mit Hilfe einer der verschiedenen Funktionen können unter anderem Feldinhalte in die Berechnung des Dokumenten-Scores einbezogen werden. Denkbar wäre hier beispielsweise, die Anzahl der Entleihungen eines Titels in einem separaten Indexfeld abzulegen und mit Hilfe der Boost-Funktionen vielgenutzten Titel beim Ranking zu bevorzugen.

In HEIDI wird die Boost-Funktion seit kurzem für ein fachspezifisches Boosting eingesetzt. Etwa 75% des Bestandes im HEIDI-Katalog ist über ein grobes Mapping der verschiedenen Systematiken und Klassifikationen einheitlich fachlich indexiert und erschlossen. Nutzer können über ein persönliches Profil ihr Fachinteresse hinterlegen. Bei der Suche werden dann Titel mit passenden Fachangaben höher gerankt als andere. Tests haben ergeben, dass sich der Parameter boost hierfür deutlich besser eignet als bq oder bf. Letztere haben den Nachteil einer additiven Scorebeeinflussung. Der Nutzen ist somit unterschiedlich

11 Die Parameterwerte müssen URI-codiert an SOLR übergeben werden und sind hier zur besseren Lesbarkeit uncodiert dargestellt.

groß, je nachdem, welcher Score bereits durch die ursprüngliche Anfrage erzielt wird.

Für den aktuell laufenden öffentlichen Beta-Test zum Fachboosting wurde daher folgende Parametrisierung gewählt:[12]

```
boost=if(query({!v=„fach:<FACHNUMMER>„}),2,1)
```

Dieser Parameter bewirkt folgendes: Nach der Ermittlung der Treffermenge wird für jedes Dokument festgestellt, ob die Suchanfrage fach:<FACHNUMMER> auf dieses Dokument ebenfalls zutrifft. Im positiven Fall wird der Dokumenten-Score mit zwei multipliziert, ansonsten mit eins (bleibt also wie er ist).

für die aktuelle Sitzung deaktivieren oder wieder aktivieren kann (Abbildung 7).

Will man das Erscheinungsjahr als „Freshness"-Faktor (3) ins Ranking einbeziehen, so bietet es sich an, den Feldinhalt direkt über eine entsprechende Funktion zu verarbeiten (15). Voraussetzung ist, dass dieses Indexfeld tatsächlich nur Zahlen enthält, was sich aber für eine performante Sortierung anhand dieses Felds sowieso empfiehlt. Im HEIDI-Katalog wird das Erscheinungsjahr aktuell allerdings noch über den statischen Dokumentenboost berücksichtigt. Dies hat gegenüber Boostingfunktionen den Vorteil, dass die Scoreberechnung performanter ist, weil die dadurch beeinflusste *fieldNorm* bereits

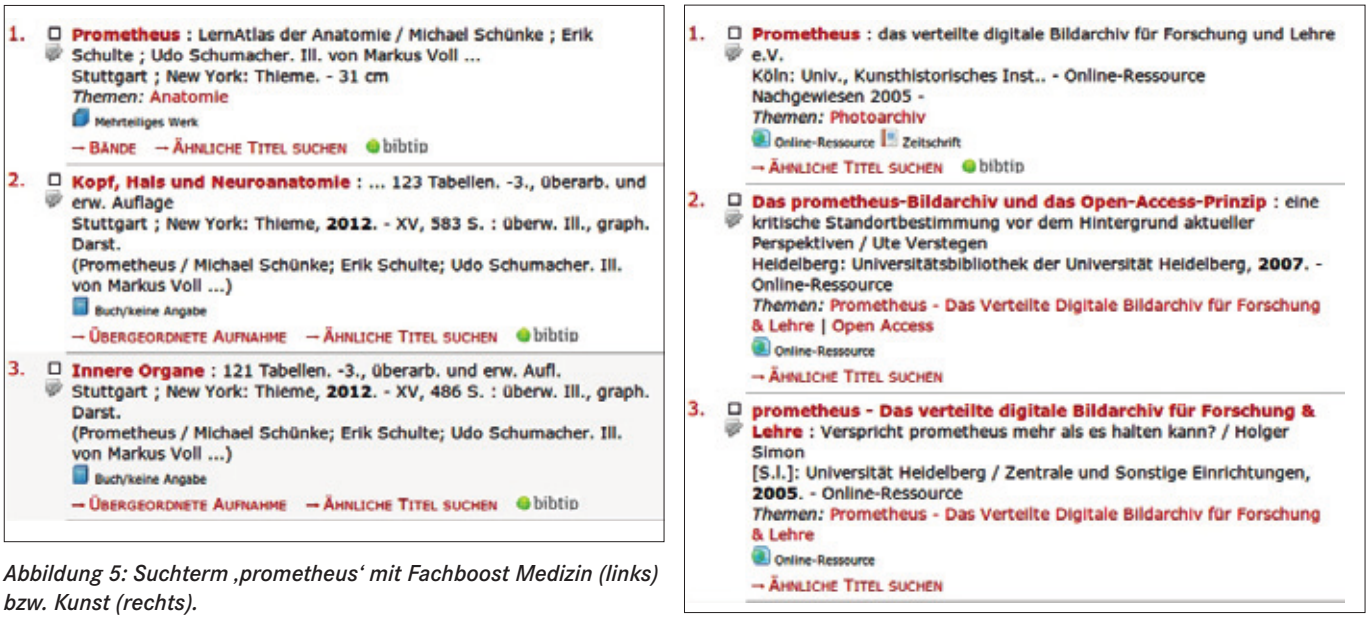

Abbildung 5: Suchterm ‚prometheus' mit Fachboost Medizin (links) bzw. Kunst (rechts).

Ob diese Gewichtung genau so erhalten bleibt, muss der weitere Echtbetrieb zeigen. Eine Überlegung ist, den Nutzer zukünftig selbst darüber entscheiden zu lassen, wie stark das Ranking beeinflusst werden soll (durch Vorgabe mehrerer Stufen).

Dass ein fachspezifisches Boosting auch in einem im Vergleich zu den großen Discovery-Systemen überschaubaren Katalog einer einzelnen Einrichtung sinnvoll sein kann, zeigen die beiden folgenden Beispiele (beide entnommen aus der Liste der 50 häufigsten Suchanfragen in HEIDI): Die Ergebnisliste zur Suche ‚prometheus' mit Fachboost Medizin und daneben das Ergebnis derselben Suche mit Fachboost Kunst (Abbildung 5), sowie das unterschiedliche Ranking zum Suchterm ‚acp' mit den Fachboosts Wirtschaft bzw. Medizin (Abbildung 6).

In der Trefferanzeige wird der Nutzer auf das von ihm gewählte Fachboosting hingewiesen, das er jederzeit

anfrageunabhängig im Index gespeichert ist. Eine nutzergesteuerte Gewichtung dieses Rankingfaktors (nicht für jede Disziplin ist der Bedarf an topaktueller Literatur gleich hoch (3)) ist beim statischen Verfahren jedoch nicht möglich.

Weitere Beispiele für die vielfältigen Einsatzmöglichkeiten der genannten Boost-Parameter finden sich in (5) und (6).

Suchterm-Auslassungen

Sowohl DisMAX als auch EDisMAX stellen den Parameter mm (= Minimum Should Match) bereit, der gegenüber den Standardverknüpfungen weitere Varianten ermöglicht. Grundsätzlich kann in SOLR festgelegt werden, ob Suchterme, die nicht explizit als „mandatory" (+) oder „prohibited" (-) deklariert sind, mit AND oder OR verknüpft werden sollen. Mit dem Parameter mm sind nun auch Vorgaben möglich wie:

- „80 Prozent der Suchterme müssen im Trefferdokument vorkommen" oder

12 Auch diese Werte müssen URI-codiert übergeben werden.

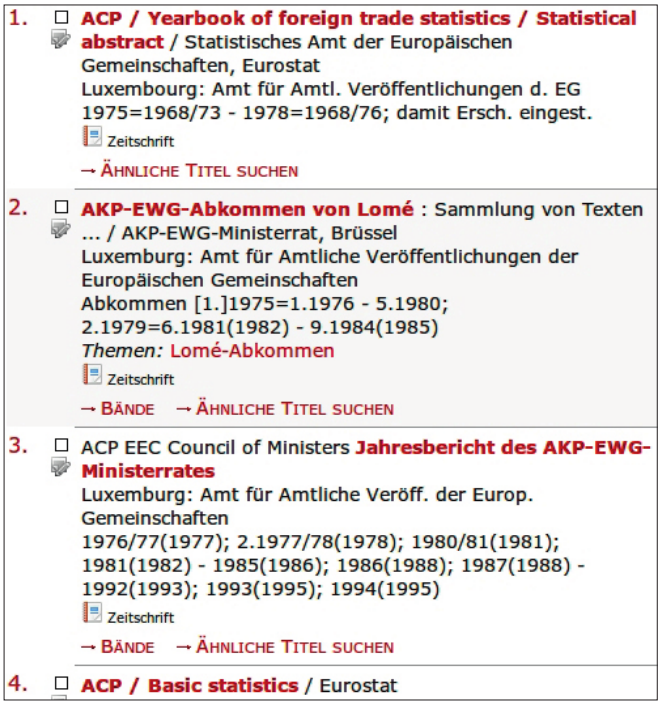

Abbildung 6: Suchterm ‚acp' mit Fachboost Wirtschaft (links) bzw. Medizin (rechts).

- „Bei weniger als drei Suchtermen müssen alle Terme im Trefferdokument vorkommen; zwischen drei und fünf Suchtermen darf einer der Suchterme fehlen; Über fünf Suchterme reicht es, wenn 80 Prozent der Suchterme im Trefferdokument vorkommen."

Für den HEIDI Katalog wurde diese letztgenannte Variante gewählt. Im ersten Schritt wird zunächst immer mit „mm=100%" gesucht. Nur wenn das Suchergebnis weniger als vier Treffer umfasst, wird die Suche nochmal mit „mm=2<-1 5<80%" durchgeführt.[13]

Da in HEIDI der Verlagsort nicht im Freitextfeld indexiert ist (ansonsten würde eine Suche nach bestimmten Ortsnamen wie „Heidelberg" eine zu große Trefferzahl liefern), ist diese Funktion recht nützlich, wenn Nutzer ganze Literaturzitate inklusive des Verlagsorts per Copy-and-Paste in den Suchschlitz übernehmen. Aber auch in anderen Fällen kommt es bei übernommenen Literaturzitaten vor, dass bestimmte Suchterme nicht katalogisiert sind.

Die folgende Suchanfrage liefert in HEIDI das gewünschte Ergebnis, obwohl der Verlagsort Heidelberg

13 Auch diese Werte müssen URI-codiert übergeben werden.

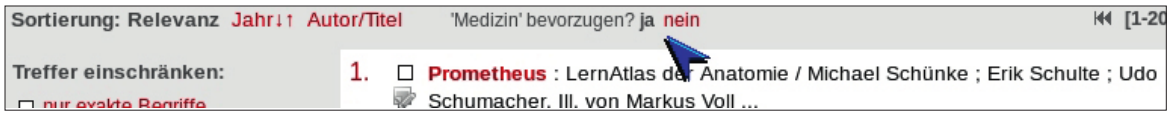

Abbildung 7: (De-)Aktivieren des Fachboostings in der Trefferliste.

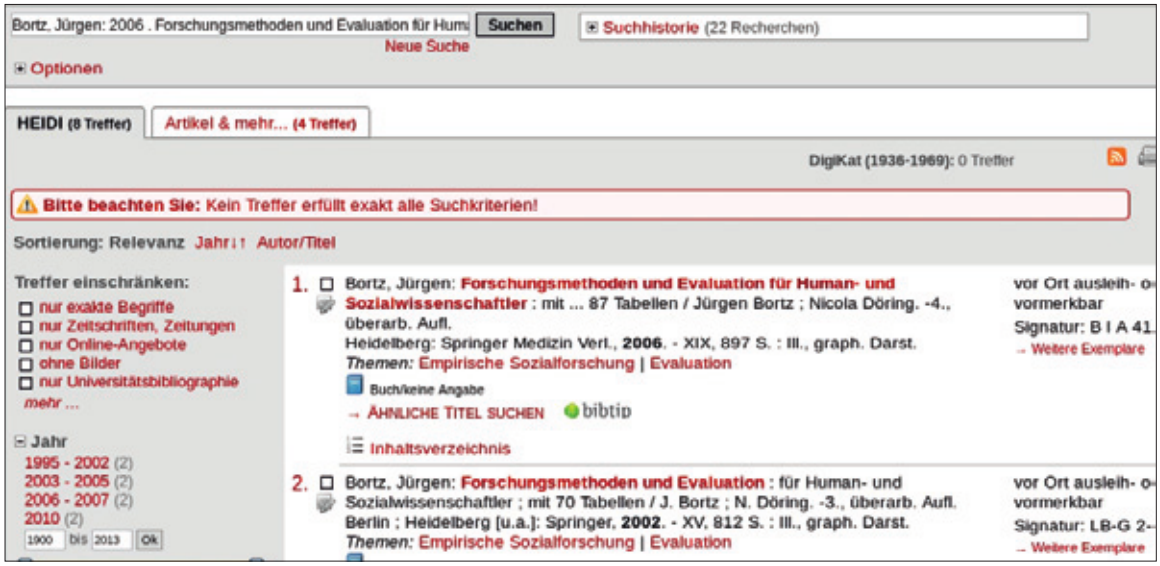

Abbildung 8: Auslassung von Suchtermen.

nicht im Freitextfeld indexiert ist. Der Nutzer wird mit einem entsprechenden Hinweis, wie in Abbildung 8 dargestellt, auf diesen Sachverhalt aufmerksam gemacht.

Suchanfrage: „Bortz, Jürgen: 2006. Forschungsmethoden und Evaluation für Human- und Sozialwissenschaftler. Heidelberg, Springer."

Einschränkungen von EDisMAX

Leider hat der Einsatz von EDisMAX auch seine Grenzen. So greift der Parameter mm (auch die Einstellung mm=100%, d.h., alle Suchterme müssen auch in den Treffern vorkommen) nur auf die sogenannten „top-level query" und nicht auf „sub-queries". Dies bedeutet, dass eine Klammerung von Anfrageteilen, wie sie z.B. beim Zusammensetzen einer Anfrage aus einer gefelderten Suche zwingend notwendig ist, dazu führt, dass alle Suchterme innerhalb der Klammerung als optional angesehen werden, unabhängig vom gewählten mm-Parameter. Dadurch erhält man sehr große Treffermengen und zahlreiche Treffer, die wenig mit den gewählten Suchbegriffen zu tun haben. Das ist spätestens dann störend, wenn man statt des Rankings eine Treffersortierung wählt.

Rang in Top-50-Suchanfragen	Suchterm (Freitextsuche)	Position in Trefferliste (ohne QueryElevation)	Position durch Query Elevation
1	jstor	1	
2	examen online	1[15]	
3	juris	10	1
4	öffnungszeiten	-	
7	palandt	1[16]	
11	jz	1[17]	
16	nature	1	
17	duden	120	1
19	njw	13	1[18]
25	prometheus	?	
34	mortimer chemie	1	
38	mla	1	
43	tipler	1	

Abbildung 9: Beispiele einiger Rankingergebnisse aus den Top-50-Suchanfragen.

Aus diesem Grund wird im HEIDI-Katalog sowohl bei der Feldsuche als auch bei Suchanfragen mit Klammerungen in der einfachen Suche auf EDisMAX verzichtet. Das Feld- und Phrasenboosting erfolgt in diesem Fall dann mit der beschriebenen Erweiterung der Suchanfrage im Katalog-Frontend.

Einige der oben genannten Boostfunktionen lassen sich auch ohne EDisMAX realisieren. So kann beispielsweise das oben dargestellte Fachboosting auch beim Standard Query Parser mit folgenden Parametern erreicht werden:[14]

- &q={!boost b=$fachboost v=$qq}
- &qq=<URSPRÜNGLICHE SUCHANFRAGE>
- &fachboost=if(query($v),2,1)
- &v=fach:<FACHNUMMER>

Query Elevation

Die Nutzung der QueryElevationComponent (13) von SOLR stellt die extremste Form der Beeinflussung des Rankings dar. In der Dokumentation wird sie auch mit „editorial boosting" oder „best bets" bezeichnet. Zu vorgegebenen Nutzeranfragen können die Dokumente/Titel konfiguriert werden, welche die Trefferliste anführen sollen.

Der notwendige Konfigurationsaufwand lohnt sich sicher nur für oft gestellte Suchanfragen, bei denen die übrigen Rankingmaßnahmen versagen und bei denen man den gesuchten Titel eindeutig identifizieren kann. Darüber hinaus ist es sinnvoll, bei den häufigsten Suchanfragen – wo möglich – das Suchergebnis zu optimieren, selbst dort wo das Ranking schon recht gute Ergebnisse zeigt.

Im HEIDI-Katalog wurden die 50 meistgestellten Suchanfragen (TOP-50) daraufhin untersucht, ob die mutmaßlich gesuchten Titel weit oben in der Trefferliste auftauchen. Dabei hat sich herausgestellt, dass das Ranking in der Regel sehr gute Ergebnisse liefert, wie in den Beispielen in Abbildung 9 ersichtlich. In der Trefferliste zu ‚nature' wird beispielsweise die Online-Ausgabe an Position 1, die Print-Ausgabe an Position 2 geführt.

In wenigen Einzelfällen war es sinnvoll, die Position der gesuchten Titel über Query Elevation zu verbessern: Bei ‚duden' führte die Vielzahl sehr ähnlicher Titelsätze und der weit verbreitete Besitz in fast allen Institutsbibliotheken dazu, dass der Score der ersten 124 Treffer nur minimal voneinander abwich und andere (Print-)Ausgaben vor die Online-Ausgabe positioniert wurden. Mittels QueryElevation wurde die Online-Ausgabe des ‚Duden' nun von Position 120 auf 1 gesetzt. Auch für ‚juris' und ‚njw' wurde das Suchergebnis auf diese Weise optimiert.

Kann die Suchanfrage dagegen – wie im Bsp. ‚prometheus' – nur im fachlichen Kontext interpretiert werden, so ist die QueryElevation ungeeignet und das Fach-Boosting vorzuziehen (siehe oben).

Durch einen gemeinsamen Suchschlitz auf der Home-

14 Auch hier wurde aus Darstellungsgründen auf eine URI-Codierung verzichtet.

15 An Position 1 wird 'Examen Online Klinik', an Position 2 'Examen Online Vorklinik' geführt.

16 Die neuesten Ausgaben des 'Bürgerlichen Gesetzbuches' werden an Position 1 und 2 geführt.

17 An Position 1 wird 'JZ / Schriftenreihe' gelistet, an Position 2 die 'Juristenzeitung' in der Online-Ausgabe.

18 Die Online-Ausgabe wurde an Position 1, die Print-Ausgabe an Position 2 gesetzt.

page zum wahlweise Durchsuchen von HEIDI oder dem übrigen Webangebot der Universitätsbibliothek kommen relativ häufig „Irrläufer-Suchen“ wie zum Beispiel Kataloganfragen nach „Öffnungszeiten“ oder „Fernleihe“ vor. Diese werden in HEIDI im Frontend abgefangen. Der Nutzer bekommt zusätzlich zu etwaigen Katalogtreffern auch entsprechende Links zum Webangebot der Universitätsbibliothek präsentiert.

Bei allen Bemühungen, das Ranking zu verbessern, wird es immer Nutzer geben, die lieber mit einer (chronologisch oder alphabetisch) sortierten Trefferliste arbeiten. Meist sind das Suchprofis (nicht selten Bibliothekare), die sehr differenziert suchen, kleine Treffermengen erzielen und deshalb ohne Ranking besser zurechtkommen. Um diesen Nutzern gerecht zu werden, bietet der HEIDI-Katalog diese Sortierfunktionen an (es gibt inzwischen auch Anbieter von Suchlösungen, die keine alphabetische Sortierung mehr vorsehen) und im persönlichen Profil können Nutzer eine Standardsortierung für ihre Anfragen auswählen.

```
ParsedQuery:
 BoostedQuery(boost(+(exttext:nature^0.5 | ti_stem:natur^0.2 |
 sw_stem:natur^0.2 | 1w:nature^2.0 | freitext:nature)~0.1
 () () () (),if(query(fach:33,def=0.0),const(2),const(1))))
```

***Abbildung 10:** Beispiel ParsedQuery in der Debug-Funktion von SOLR.*

Explain-Funktionen

Zur Überprüfung der gewählten Rankingmaßnahmen bietet SOLR sehr schöne und für genauere Analysen unerlässliche Debugging- und Explain-Funktionen. Durch Anhängen des Parameters *debugQuery=true* erhält man nicht nur Informationen zum Parsing und der Analyse der Suchanfrage, auch die Berechnung

```
16.690388 = (MATCH) boost(+(exttext:nature^0.5 | ti_stem:natur^0.2 | sw_stem:natur^0.2 | 1w:nature^2.0 | freitext:nature)~0.1 () () ()
(),if(query(fac_fach:33,def=0.0),const(2),const(1))), product of:
 16.690388 = (MATCH) sum of:
  16.690388 = (MATCH) max plus 0.1 times others of:
   1.01156 = (MATCH) weight(ti_stem:natur^0.2 in 2846591) [UBHDSimilarity], result of:
    1.01156 = score(doc=2846591,freq=4.0 = termFreq=4.0 ), product of:
     0.047317535 = queryWeight, product of:
      0.2 = boost
      6.108034 = idf(docFreq=28558, maxDocs=4722088)
      0.038733847 = queryNorm
     21.37812 = fieldWeight in 2846591, product of:
      2.0 = tf(freq=4.0), with freq of:
       4.0 = termFreq=4.0
      6.108034 = idf(docFreq=28558, maxDocs=4722088)
      1.75 = fieldNorm(doc=2846591)
   16.11931 = (MATCH) weight(1w:nature^2.0 in 2846591) [UBHDSimilarity], result of:
    16.11931 = score(doc=2846591,freq=1.0 = termFreq=1.0 ), product of:
     0.9994902 = queryWeight, product of:
      2.0 = boost
      12.902026 = idf(docFreq=31, maxDocs=4722088)
      0.038733847 = queryNorm
     16.127533 = fieldWeight in 2846591, product of:
      1.0 = tf(freq=1.0), with freq of:
       1.0 = termFreq=1.0
      12.902026 = idf(docFreq=31, maxDocs=4722088)
      1.25 = fieldNorm(doc=2846591)
   4.699216 = (MATCH) weight(freitext:nature in 2846591) [UBHDSimilarity], result of:
    4.699216 = score(doc=2846591,freq=4.0 = termFreq=4.0 ), product of:
     0.26982862 = queryWeight, product of:
      6.966223 = idf(docFreq=12106, maxDocs=4722088)
      0.038733847 = queryNorm
     17.415558 = fieldWeight in 2846591, product of:
      2.0 = tf(freq=4.0), with freq of:
       4.0 = termFreq=4.0
      6.966223 = idf(docFreq=12106, maxDocs=4722088)
      1.25 = fieldNorm(doc=2846591)
 1.0 = if(query(fach:33,def=0.0)=0.0,const(2),const(1))
```

***Abbildung 11:** Beispiel Explain-Funktion von SOLR.*

des Scores der einzelnen Treffer wird detailliert dargestellt.

Mit Hilfe des Parameters *explainOther* ist es zudem möglich, für einzelne Dokumente festzustellen, warum sie die Suchanfrage nicht erfüllt haben oder warum sie nicht hoch genug gerankt wurden (5, S. 122). Die Webseite *explain.solr.pl*[19] bietet die Möglichkeit das explain-Ergebnis in eine grafische Darstellung (als Tortendiagramm) umzusetzen. In der aktuellen Version scheitert diese Umsetzung leider noch, wenn im explain-Ergebnis anstatt der DefaultSimilarity eine eigene Similarity-Klasse (hier [UBHDSimilarity]) aufgeführt ist. Dieser „Schönheitsfehler" sollte aber hoffentlich bald korrigiert sein.

Ausblick

Mit all diesen Maßnahmen, die bereits zu sehr guten Rankingergebnissen führen, sind die Möglichkeiten noch nicht ausgeschöpft.

Mit Ausnahme der Online-Medien wurde bislang im HEIDI-Katalog die Verfügbarkeit eines Mediums, d.h. Faktoren der „Locality" (3) noch nicht in das Ranking einbezogen.

Auch bei sogenannten „Popularity-Faktoren" (3) wurde bislang nur die Anzahl der besitzenden Bibliotheken (als indirekter Parameter) berücksichtigt, nicht jedoch die Nutzungshäufigkeit der Titel. Ein Bevorzugen von stark genutzten oder aktuell verfügbaren Titeln ist nicht unproblematisch (1) und wurde deshalb bislang noch nicht realisiert.

Ebenso wäre manchem Nutzer sicherlich daran gelegen, eine präferierte Teilbibliothek/Zweigstelle festzulegen und die Medien dieser Zweigstelle damit höher gewichten zu können (ebenfalls ein „Locality"-Faktor).

Im Rahmen der Arbeiten an persönlichen Nutzerprofilen sollen in Kürze die entsprechenden Optionen im HEIDI-Katalog angeboten werden. Voraussetzung ist die Einbeziehung der Ausleihstatistiken sowie des aktuellen Verfügbarkeitsstatus in den SOLR-Index. Gerade bei Letzterem ist selbstverständlich eine hohe Aktualität wichtig. Über entsprechende Mechanismen, welche die Datenbank des Bibliotheksinformationssystems zur Verfügung stellt, können hierzu laufend Informationen zu den geänderten Datensätzen ermittelt und der SOLR-Index in kurzen Intervallen geändert werden.

Der Nutzer soll zudem über einen einfachen Mausklick diese, in seinem Profil definierten Rankinganpassungen in der Trefferliste an- oder abschalten können. Über die Boost-Funktionen (s.o.) ist dies in SOLR recht einfach realisierbar. Um mehrere Einflussfaktoren abzubilden, können die Parameter für die Boost-Funktionen auch mehrfach in einer Anfrage verwendet werden.

Es bleibt abzuwarten, wie gut die Personalisierungsfunktionen insgesamt und speziell die persönlichen Rankingoptionen von den Nutzern angenommen werden.[20] Dies bestimmt, wie schnell und weit sich der HEIDI-Katalog in Richtung eines individuell einstellbaren Katalogs weiterentwickeln wird.

19 http://explain.solr.pl/ [31. Januar 2013].

Literaturhinweise

1. Langenstein, Annette/ Maylein, Leonhard: „Relevanz-Ranking im OPAC der Universitätsbibliothek Heidelberg", in: b.i.t.online, 12 (2009), S. 408-413.
2. Oberhauser, Otto: „Relevance Ranking in den Online-Katalogen der nächsten Generation", in: Mitteilungen der Vereinigung Österreichischer Bibliothekarinnen & Bibliothekare, 63 (2010), S. 25-37.
3. Lewandowski, Dirk: „Ranking library materials", in *Library Hi Tech*, 27 (2009), S. 584-593.
4. Wiesenmüller, Heidrun/ Maylein, Leonhard/ Pfeffer, Magnus: „Mehr aus der Schlagwortnormdatei herausholen : Implementierung einer geographischen Facette in den Online-Katalogen der UB Heidelberg und der UB Mannheim", in: b.i.t.online, 14 (2011), S. 245-252.
5. Smiley, David/ Pugh, Eric: Apache Solr 3 enterprise search server : Enhance your search with faceted navigation, result highlighting, relevancy ranked sorting, and more, Birmingham, UK [u.a.]: Packt Publishing 2011.
6. Kuć, Rafał: Apache Solr 3.1 cookbook : over 100 recipes to discover new ways to work with Apache's Enterprise Search Server, Birmingham [u.a.]: Packt Publishing 2011. [Eine Neuauflage zu Solr 4 ist für März 2013 angekündigt]
7. SOLR: http://lucene.apache.org/solr/ [31. Januar 2013].
8. Similarity-Konfiguration in SOLR: http://wiki.apache.org/solr/SchemaXml#Similarity [31. Januar 2013].
9. Lingo: http://lex-lingo.blogspot.de/ [31. Januar 2013].
10. SOLR, WordDelimiterFilter: http://wiki.apache.org/solr/AnalyzersTokenizersTokenFilters#solr.WordDelimiterFilterFactory [31. Januar 2013].
11. SOLR, Extended DisMAX Query Parser: http://wiki.apache.org/solr/ExtendedDisMax [31. Januar 2013].
12. SOLR, DisMAX Query Parser: http://wiki.apache.org/solr/DisMaxQParserPlugin [31. Januar 2013].
13. SOLR, Query Elevation: http://wiki.apache.org/solr/QueryElevationComponent [31. Januar 2013].
14. SOLR, Synonym-Filter: http://wiki.apache.org/solr/AnalyzersTokenizersTokenFilters#solr.SynonymFilterFactory [31. Januar 2013].
15. SOLR, Date-Boosting: http://wiki.apache.org/solr/FunctionQuery#Date_Boosting [31. Januar 2013].

Leonhard Maylein
Annette Langenstein
Abt. Informationstechnologie und EDV-Versorgung
Universitätsbibliothek
Plöck 107-109
69117 Heidelberg
maylein@ub.uni-heidelberg.de
langenstein@ub.uni-heidelberg.de

20 Immerhin muss sich ein Nutzer zunächst einloggen, damit sein persönliches Profil wirksam wird – eine Hürde, die jedoch mit zunehmender Verbreitung von Single-Sign-on-Verfahren (Shibboleth) für universitäre Dienste ihren Schrecken verlieren dürfte.