Projekt: Semantic Web

Entwurf und Integration eines Item-Based Collaborative Filtering Tag Recommender Systems in das Bibsonomy Projekt

Jens Illig
illig@innofinity.de

18. April 2006

Zusammenfassung

Die Projektarbeit trägt den Titel *Tag Recommender* und beschäftigt sich mit der Erweiterung des Social Bookmarking Systems *Bibsonomy* um automatisch erstellte Tagvorschläge. Kern der Arbeit ist das Ermitteln von individuell passenden Tagging-Vorschlägen für URLs und BibTeX-Einträgen aus einem bisherigen Datenbestand von Taggings.

1 Einleitung

Ziel der in diesem Dokument beschriebenen Projektarbeit am Fachgebiet Wissensverarbeitung der Universität Kassel ist die Erweiterung des bestehenden *Bibsonomy*-Systems um die Funktionalität eines Tag-Recommenders. *Bibsonomy* als solches ermöglicht die Eingabe von URLs und deren Markierung mit Schlüsselworten (Tagging). Zusätzlich zur Möglichkeit URLs auszuzeichnen erlaubt es das System Gleiches mit BibTeX-Einträgen zu tun.
Ein solches Tagging wird stets nach Anmeldung des Benutzers unter einem Benutzeraccount vorgenommen, so dass neben der Zuordnung von URLs bzw. BibTeX-Einträgen zu Tags auch eine Verknüpfung zum Benutzer existiert.
Basierend auf den vom *Bibsonomy*-System angelegten bidirektionalen Verknüpfungen zwischen URLs bzw. BibTeX-Einträgen und Tags sowie zwischen Benutzern und Tags kann eine *Collaborative Filtering* Implementation Benutzern Tags vorschlagen, die den bisherigen Tags des Benutzers ähneln oder zu dem zu taggenden Eintrag (URL oder BibTeX) passen, sofern dieser dem System bereits zuvor bekannt war.
Durch Tag-Empfehlungen für neue Einträge eines Benutzers wird eine höhere Tag-Konsistenz und Frequenz in Form von Wiederverwendung bereits bekannter Tags und mehr Komfort beim Tagging-Vorgang ermöglicht.
In diesem Dokument werden verschiedene Ansätze zur Umsetzung eines Tag Recommenders diskutiert und schließlich wird ein kurzer Überblick über die Implementation und Integration eines ausgewählten Verfahrens in das Bibsonomy-Projekt gegeben.
Im Unterschied zu den üblichen Recommender-Systemen wie sie beispielsweise auf Webauftritten von Online Shops Verwendung finden, ist eine Selektion des Benutzers nicht ein 2-Tupel aus Benutzer und Inhalt (Item), sondern es existiert immer auch die Verbindung zu einem Tag (einem Wort, das der Benutzer mit dem Inhalt verbindet). Eine Selektion (Post) ist also ein Tripel aus Benutzer, Tag und Bookmark bzw BibTeX-Eintrag. Eine Beispielsituation, in der ein Tag vorgeschlagen werden soll, ist in Abbildung 1 in UML-Syntax dargestellt.

2 Collaborative Filtering

2.1 Item-Based

Übliche Recommender Systeme auf Basis von Collaborative Filtering wie in [2] beschrieben, berechnen die Ähnlichkeit der Benutzer durch Bestimmung einer Metrik zur Messung der Übereinstimmung von Selektionen (Bewertungen) der User. Der Ansatz des Collaborative Filtering Verfahrens ist auf

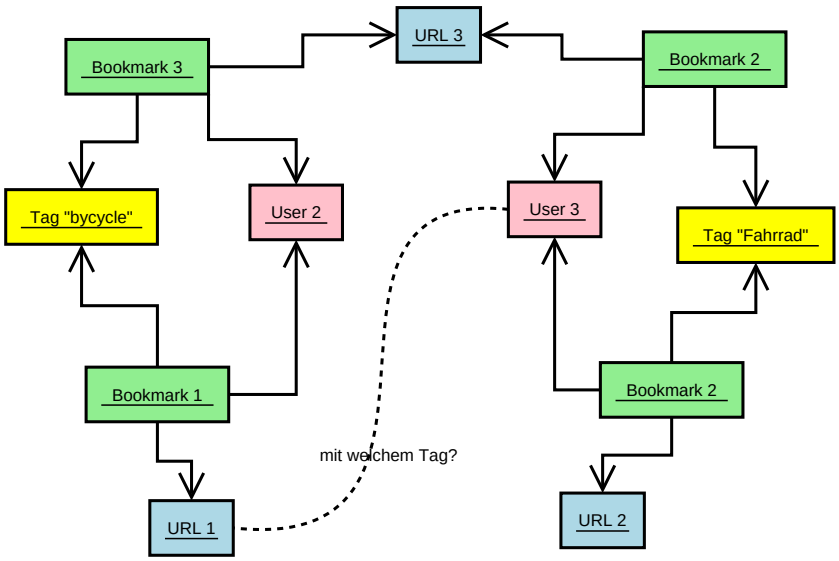

Abbildung 1: User 3 möchte URL1 mit einem Tag versehen

der Annahme aufgebaut, dass Benutzer, welche bisher gleiche Inhalte gleich bewertet haben, auch weiterhin in ihren Meinungen übereinstimmen werden. So wird versucht eine Wertung eines Benutzers für jedes beliebige Objekt vorherzusagen. Die Objekte, welche die besten geschätzten Wertungen erhalten haben, werden vorgeschlagen.
Eine Problematik solcher Systeme ist ihr sehr hoher (quadratisch mit der Anzahl der Benutzer steigender) Rechenaufwand bzw. Speicheraufwand beim Bereitstellen der Ähnlichkeiten (Nachbarschaften) zwischen Benutzern.

2.2 User-Based

Item-Based Collaborative Filtering Ansätze [3] versuchen den Rechenaufwand geringer zu halten indem nicht die Ähnlichkeit zwischen Benutzern, sondern die Ähnlichkeit zwischen Inhalten (Items) vorberechnet wird. Die Annahme, welche zur erhofften Steigerung der Performance führen soll ist die verhältnismäßig statisch bleibende Beziehung zwischen Inhalten. Bei der Berechnung von Abschätzungen wie gut ein Inhalt zu einem Benutzer passt (also größenordnungsmäßig wie wahrscheinlich es sein wird, dass der Inhalt vom Benutzer selektiert werden wird), summieren Item-Based Recommender die Ähnlichkeit des zu untersuchenden Inhalts zu allen vom Benutzer bereits selektierten Inhalten. Dies sind i.d.R. verhältnismäßig wenige. User-Based Recommender hingegen verwenden erst die Ähnlichkeit des Benutzers zu anderen Benutzern um dann deren bewertete Inhalte gewichtet aufzuaddieren und auf diesem Wege Gütewerte für mögliche Selektionen zu erhalten.

3 Umsetzung

3.1 Datenbasis

Übertragen auf die Datensätze von Bibsonomy bedeutet der Einsatz von Model-Based Collaborative Filtering, dass eine der Dimensionen Benutzer, Tags und Inhalte zur Vorberechnung ausgewählt wird. Inhalte wie Bookmarks und BibTeX-Einträge gibt es in extrem großer Anzahl. Die Dimension der Inhalte scheidet also wegen hohem Aufwand beim Auffinden der ähnlichen Inhalte aus. Mag das in [3] genannte MovieLens-Beispiel mit 3500+ verschiedenen Filmen, noch verhältnismäßig gut inhaltsbasiert berechenbar sein, so ist trotz mittlerweile 4-6x schnelleren Rechnern wegen des quadratischen Wachstums des Aufwands eine Datenbank von über sieben Millionen Bookmarks noch immer sehr schwer handhabbar. Selbst wenn die nächsten k Bookmarks vorberechnet vorlägen sind damit schwerlich Tag-Recommendations umsetzbar, da viele Inhalte von nur einem Benutzer gepostet wurden und damit aus einem ähnlichen Inhalt nur sehr wenige Bookmarks (häufig ein oder zwei pro Post) für die

Auswertung gewonnen werden können.
Mehr Sinn macht eine Vorberechnung der k ähnlichsten Benutzer für jeden Benutzer. Hier liegen im Testdatensatz ca 75000 Benutzer vor, was weniger ist als die Anzahl an Tags mit 550000. Diese Arbeit beschäftigt sich jedoch mit der Anwendbarkeit von Item-Based Collaborative Filtering, genauergesagt mit der Erstellung von Recommendations auf Basis von Tag-Tag-Ähnlichkeiten, da Tags vorgeschlagen werden sollen. So entsteht beim Auffinden von Tags, die den vom Benutzer bereits zuvor verwendeten Tags ähnlich sind, kein Umweg über ähnliche Benutzer. Dieser Umweg würde neben mehr vorschlagbaren Tags zwangsläufig auch eine zusätzliche Indirektion und Unsicherheit in die Recommendations miteinfließen lassen. Zudem steigt die Anzahl der Tags schwächer als linear mit der Anzahl der Posts und ähnelt einer Wurzelfunktion, was praktikabel bleibenden Rechenaufwand bei wachsendem Datenbestand schließen lässt.

Da Bibsonomy gegenüber üblichen Recommender Szenarios eine Stelle mehr in den Relationen besitzt (Tags), ergibt sich für die Speicherung von Posts eine boolsche Matrix wie in Bild 2 dargestellt.
Um die dreidimensionale Matrix in eine auf die bekannten Recommender Algorithmen anwendbare Form zu bringen, kann auf mehrere Arten zu zweidimensionalen Matrizen aggregiert werden. Als naheliegende Aggregatfunktion bietet sich die Summation an.
Tags können als Vektoren wahlweise von Vorkommnissen an Inhalten, bei Benutzern oder einer Kombination von beidem aufgefasst werden. Mit der Wahl einer einzigen Sorte von Vorkommnissen als Vektorkomponenten entscheidet man sich für eine zweidimensionale Matrix, deren Komponenten Aggregate von Vektoren einer dreidimensinalen Matrix sind. In den Bildern 3 wird dieser Vorgang verdeutlicht. Es ergeben sich Matrizen an den Seitenwänden eines Quaders, das von der dreidimensionale Matrix beschrieben wird.

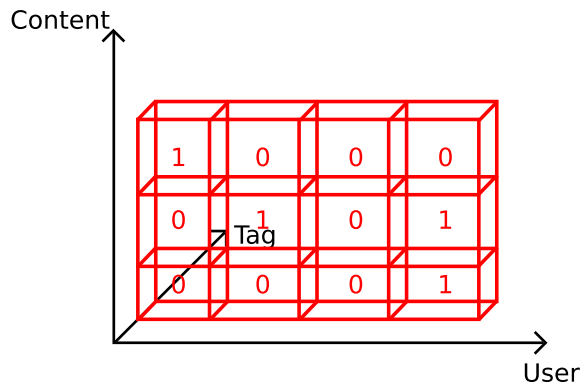

Abbildung 2: dreidimensionale Bitmatrix zur Markierung von Posts

Die Spaltenvektoren einer so gewonnenen zweidimensionalen Matrix ergeben die Repräsentanten eines Tags, welche Zwecks Aufbaus einer Liste der k ähnlichsten Tags verglichen werden können. Die Vektoren sind in einem Koordinatensystem darstellbar wie in Abbildung 4.
Zum Ermitteln der Ähnlichkeit der Vektoren können verschiedene Metriken wie Cosinus, die Pearson-Korrelation oder aber auch asymmetrische Maße verwendet werden, wie eine Implikationswahrscheinlichkeit der Form „wenn Benutzer X ein Tag verwendet hat, dann kann gefolgert werden, dass mit Wahrscheinlichkeit p auch Benutzer Y das gleiche Tag verwenden würde“. Jedes dieser Maße muss auf einem einzelnen oder einem zusammengesetzten Vektorraum aufbauen. Es ergeben sich also mehrere mögliche

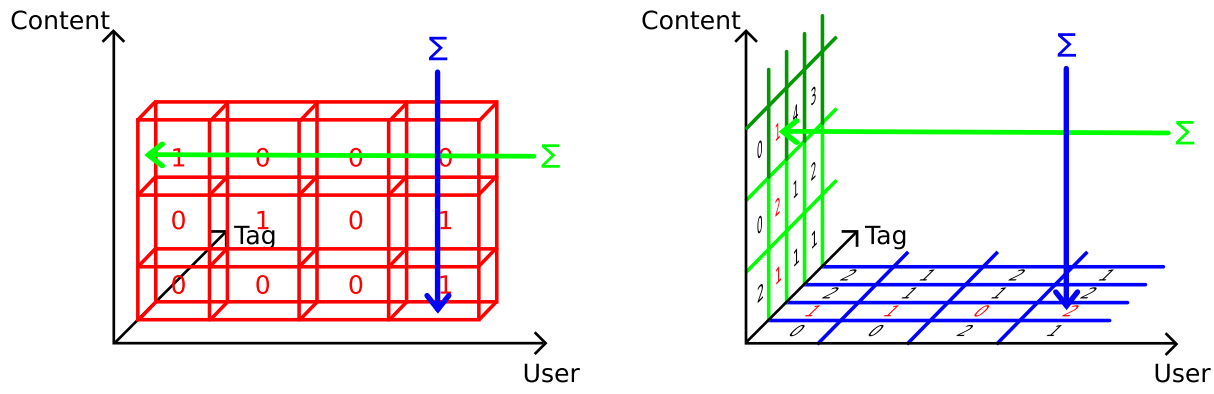

Abbildung 3: Aufbau zweidimensionaler Matrizen aus einer boolschen (posted / nicht posted) dreidimensionalen Matrix mittels Summation

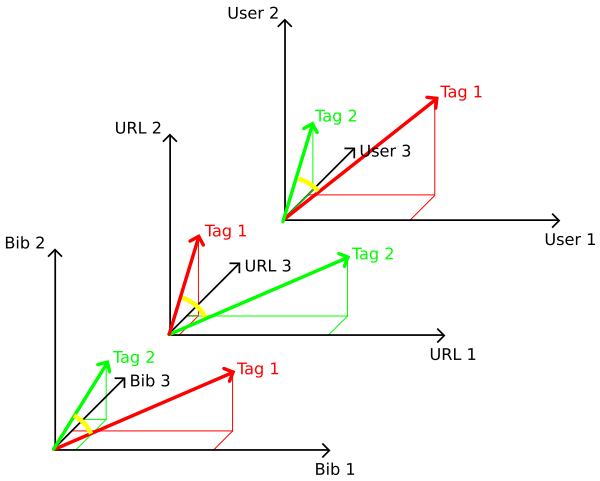

Abbildung 4: Verschiedene Vektorraumdarstellungen aus Bibsonomy-Daten

verschiedene Ähnlichkeiten bei gleichem Ähnlichkeitsmaß und zu vergleichendem Tag-Paar. Jedes der Maß wird aktualisiert werden müssen sobald der Vektor des Tags verändert wird. Dies wird beispielsweise dadurch verursacht, dass ein Benutzer ein Tag verwendet und die dem Nutzer/Inhalt entsprechende Vektorkomponente erhöht werden. Damit ändert sich auch die Ähnlichkeit zu allen oder je nach Ähnlichkeitsmaß zumindest sehr vielen Benutzern.

3.2 Ähnlichkeitsmaße

Ähnlichkeitsmaße bilden das Auswahlkriterium für die Ermittlung der nächsten Nachbarn. Zur Berechnung von der Recommendations wurde im Verlauf der Projektabwicklung auf verschiedene Matrizen und Ähnlichkeitsmaße zurückgegriffen, deren Auswirkungen am Ende des Dokuments anhand von Evaluationsdaten gezeigt werden. Verwendet wurden:

- Cosinus-Maß
  Das Cosinus-Maß ist eine häufig verwendete Standard-Metrik. Eine Beschreibung findet sich in [3].

- Implikationswahrscheinlichkeit
  Die meisten Ähnlichkeitsmaße sind Metriken. Ggf. kann es jedoch sinnvoll sein ein asymetrisches Maß zu verwenden. Z.B. könnten so Homonyme besser berücksichtigt werden. Man kann bei einer Resource, die mit dem Tag *Steuer* versehen wurde nicht folgern, ob *Auto* oder *Steuererklärung* ebenfalls angemessen wäre. Die Gegenrichtung wäre jedoch gut möglich. Ebenso könnten Konzepthierarchien besser dargestellt werden. Daher wurde ein neues Maß entwickelt, welches eine solche Asymmetrie aufweist. Die Idee dabei ist, für die Berechnung der Implikationswahrscheinlichkeit aus Tag A Tag B folgern zu können nicht mehr durch die Verwendungsanzahl des Tags B in Form der Vektorlänge des Repräsentantenvektors von Tag B zu dividieren. Als Formel kommt zum Einsatz:

$$p(T_A \Rightarrow T_B) = \frac{\vec{T_A} \cdot min(\vec{T_A}, \vec{T_B})}{\vec{T_A}^2} \tag{1}$$

  Hierbei ist min die komponentenweise agierende Minimumfunktion und $\cdot$ das Skalarprodukt. Die Minimumfunktion wird zur Normierung benötigt.

- Semisymmetrische Implikationswahrscheinlichkeit
  Die Implikationswahrscheinlichkeit neigt stark zu Implikationen zu häufig verwendeten Tags. Möchte man Äquivalenzen bevorzugen aber dennoch die Asymmetrie beibehalten, so muss die Rückimplikationswahrscheinlichkeit abgemildert in die Formel miteinfließen. Dies geschieht hier in Form einer Multiplikation mit der mittels Quadratwurzel gedämpften Rückimplikationswahrscheinlichkeit.

$$p(T_A \rightleftharpoons T_B) = p(T_A \Rightarrow T_B) \cdot \sqrt{p(T_B \Rightarrow T_A)} \tag{2}$$

4 Verwendete Recommender-Verfahren

Durch die Kombinierbarkeit von verschiedenen Ausgangsentitäten wie Benutzer, Inhalt oder Tag, verschiedenen Ähnlichkeitsmaßen und verschiedenen Indirektionsstufen und Lückenfüller-Algorithmen (zum Auffüllen zu kleiner Rückgabemengen) ergeben sich extrem viele Möglichkeiten ein Recommender System umzusetzen. In dieser Projektarbeit wurden daher nur einige wenige der möglichen Kombinationen implementiert und getestet. Diese Kombinationen und ihre Evaluation werden in den Folgenden Abschnitten vorgestellt.

4.1 Evaluation verschiedener Verfahren

Um die Ergebnisse der auf den genannten Ähnlichkeitsmaßen, sowie unterschiedlichen Miteinbeziehungen und Kombinationen von Vektoren aufbauenden Recommender Implementationen zu testen, wurde ein Testdatensatz aus Daten des Social-Bookmarking-Systems *del.icio.us* aufgebaut. Dabei wurden für die Berechnungsgrundlage des Recommenders nur Daten verwendet, welche vor Dezember 2004 in das System eingetragen wurden. Der auf diese Daten trainierte Recommender wurde eingesetzt um alle Posts des Monats Dezember vorherzusagen.
Trainingsdaten: Bookmarks vor Dezember 2004

- 1,35 mio Bookmarks
- 15605 Benutzer
- 117631 Tags

Testdaten: Bookmarks im Dezember 2004

- 265587 neue Bookmarks
- 91972 davon auf bereits bekannte Inhalte (ca. $\frac{1}{3}$)

Die Ergebnisse wurden in Form von Precision-Recall [4] Kurven in doppelter Recall-Auflösung gegenüber den Standard Recall Leveln gemessen. Dabei wurden nur Tags von Posts als relevant gewertet, welche im System bereits vor Beginn des Testzeitraums vorkommen und im Testzeitraum tatsächlich vom Benutzer für den Inhalt verwendet wurden. Tags, welche dem System zu Beginn des Testzeitraums noch nicht bekannt sind, können unabhängig vom verwendeten Collaborative Filtering Algorithmus nicht vorgeschlagen werden. Sie wurden daher als irrelevant eingestuft. Im Falle, dass ein vorgeschlagenes Tag nicht vom Benutzer verwendet wird, kann jedoch nicht gefolgert werden, dass es in keiner Form von Relevanz ist. Der Benutzer könnte es auch vergessen haben, da der Recommender, mit denen die Testdaten eingegeben wurden, nicht genau gleich funktioniert.

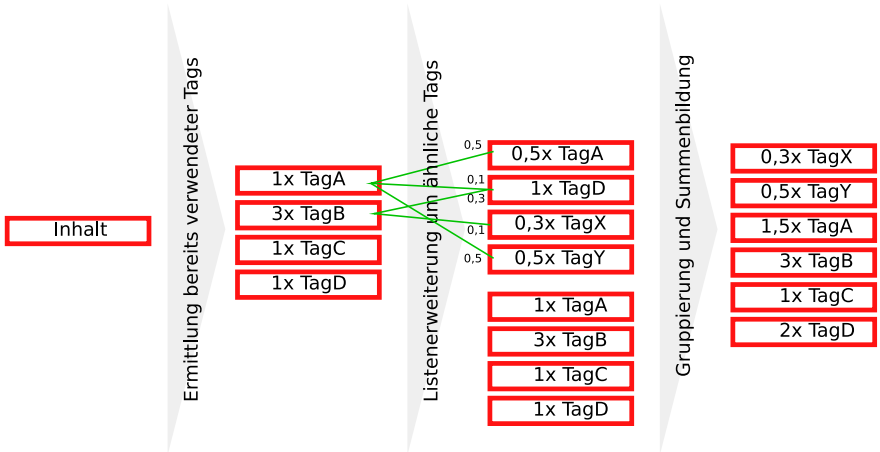

Abbildung 5: Inhaltsbasierte Tag-Expansion

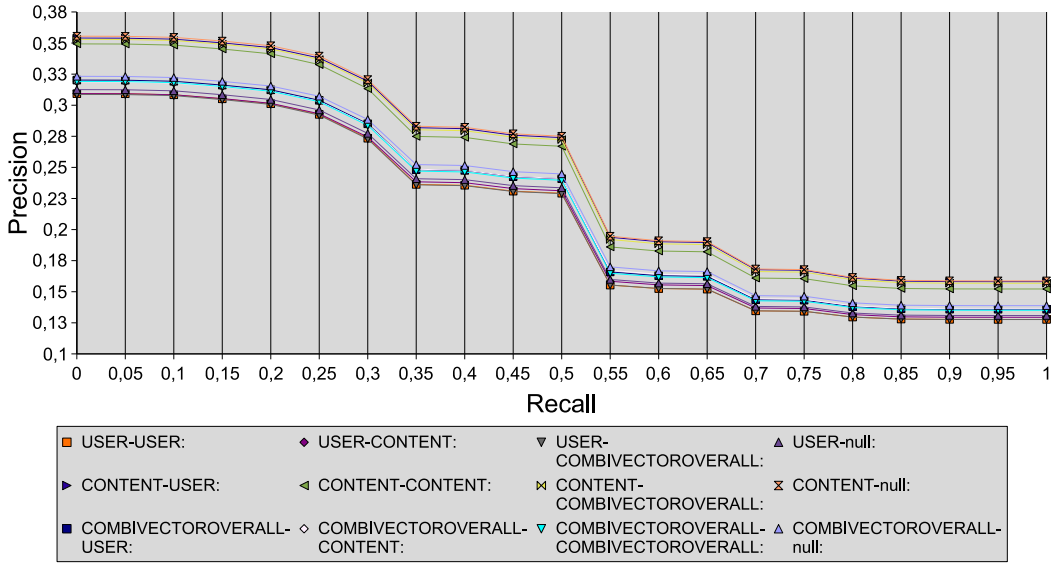

Abbildung 6: Kombination verschiedener Cosinus Ähnlichkeiten bei Tag-Expansionen von Inhalt und Benutzer

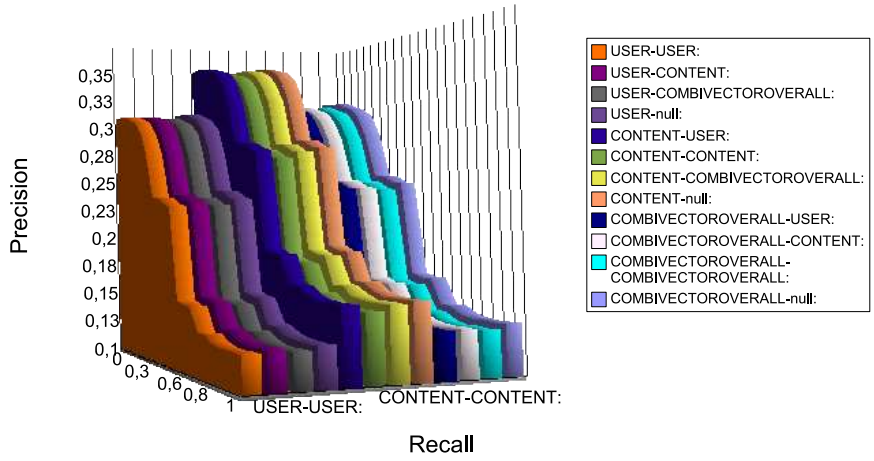

Abbildung 7: Kombination verschiedener Cosinus Ähnlichkeiten bei Tag-Expansionen von Inhalt und Benutzer

4.2 Zweiseitige Expansionen

Zunächst wurde versucht Tags, welche an bereits bekannten Inhalten stehen, und Tags, welche ein Benutzer bereits verwendet hat, in eine neue Menge an Tags zu überführen, deren enthaltene Tags möglichst viele der folgenden Bedingungen erfüllen:

- Das Tag wurde bereits zuvor für den zu taggenden Inhalt verwendet
- Das Tag wurde bereits zuvor vom gleichen Benutzer verwendet
- Das Tag befindet sich unter den k nächsten Nachbarn eines für den Inhalt bereits verwendeten Tags
- Das Tag befindet sich unter den k nächsten Nachbarn eines vom Benutzer bereits verwendeten Tags

Dieses Verfahren bildet zu jedem Tag eines Inhalts die Summe der Anzahl direkter Verwendungen am Inhalt und der mit der Tagähnlichkeit multiplizierten Vorkommen eines zweiten Tags in dessen k nächsten Nachbarn sich das erste Tag befindet. Analog wurde ausgehend vom Benutzer verfahren. Abbildung 5 visulisiert diesen Schritt ausgehend von einem dem System bereits bekannten Inhalt.

Die resultierenden Listen solcher vom Inhalt oder Benutzer ausgehenden Tag-Expansionen müssen zu einer geordneten Menge an Tag-Recommendations zusammengefasst werden. Da Benutzer Tags vieler Themenbereiche verwenden können, aber an einem Inhalt üblicherweise nur zum Themengebiet des Inhalts passende Tags verwendet werden, wertet das in diesem Abschnitt beschriebene Verfahren bei einem bekannten Inhalt lediglich die Expansion der inhalsbezogenen Tags auf, wenn ein bekannter Inhalt gewählt wird und bewertet Tags, welche aus der Expansion der benutzerbezogenen Tags garnicht hervorgehen, noch mit Faktor 0,5 anstatt sie zu ignorieren. Zur Aufwertung (wenn also ein Tag aus beide Expansionen hervorgeht) hat sich anhand von Testreihen auf dem beschriebenen Testdatensatz, die sich nur auf Post der ersten 50-100 Benutzer im Testzeitraum bezogen, die Wertung des Tags nach folgender Formel als sinnvoll erwiesen:

$$score(T_X) = (1 + \frac{16 \cdot userExpScore(T_X)}{|T_{UY}|}) \cdot contentExpScore(T_X) \quad (3)$$

Wobei $|T_{UY}|$ für die Anzahl der vom Benutzer Y verwendeten unterschiedlichen Tags steht und $userExpScore$ bzw. $contentExpScore$ für die aus der Tag-Expansion ausgehend vom Benutzer bzw. Inhalt hervorgehende Wertung des Tags X. Abbildung 6 zeigt die Precision-Recall-Kurven unter Verwendung von Cosinus-Ähnlichkeiten zwischen Vektoren bzw. Kombinationen von Vektoren aus unterschiedlichen möglichen Vektorräumen. Die Ähnlichkeiten sind:

- *USER*
  Die im Benutzer-Vektorraum gemessene Ähnlichkeit.
- *CONTENT*
  Die im Inhalt-Vektorraum gemessene Ähnlichkeit.
- *COMBIVECTOROVERALL*
  Die Tag-Repräsentatenvektoren aus dem Benutzervektorraum in einen Vektor mit höherer Dimension geschrieben und im so entstandenen Vektorraum verglichen.
- *null*
  Alle Ähnlichkieten sind 0 $\Rightarrow$ Keine Expansion. Nur direkt verwendete Tags werden berücksichtigt.

Für die inhalts- und die benutzerbezogenen Expansionen können unterschiedliche Ähnlichkeiten eingesetzt werden. Es wurden alle Kombinationen evaluiert. Dabei wurde die Schreibweise *Ähnlichkeitstyp für inhalsbezogene Expansion - Ähnlichkeitstyp für benutzerbezogene Expansion* verwendet. *CONTENT-USER* steht also für eine Kombination von inhaltsbasierter Expansion mit *CONTENT*-Ähnlichkeit und benutzerbasierter Expansion mit *USER* Ähnlichkeit.

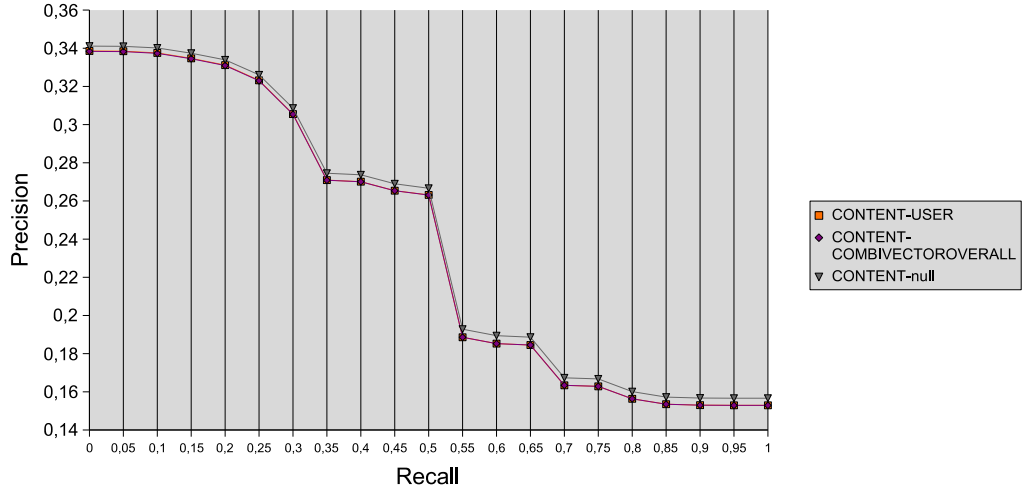

Abbildung 8: Kombination verschiedener vollständig asymetrischer Ähnlichkeiten in unterschiedlichen Vektorräumen bei Tag-Expansionen von Inhalt und Benutzer

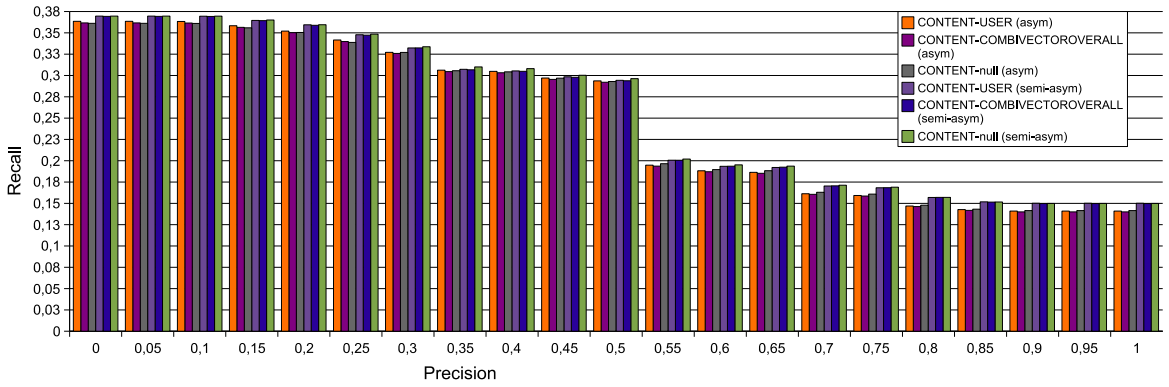

Abbildung 9: Entwicklungsevaluationen der asymetischen Maße auf dem gleichen Testdatensatz unter ausschließlicher Berücksichtigung der alphabetisch ersten 50 Benutzer

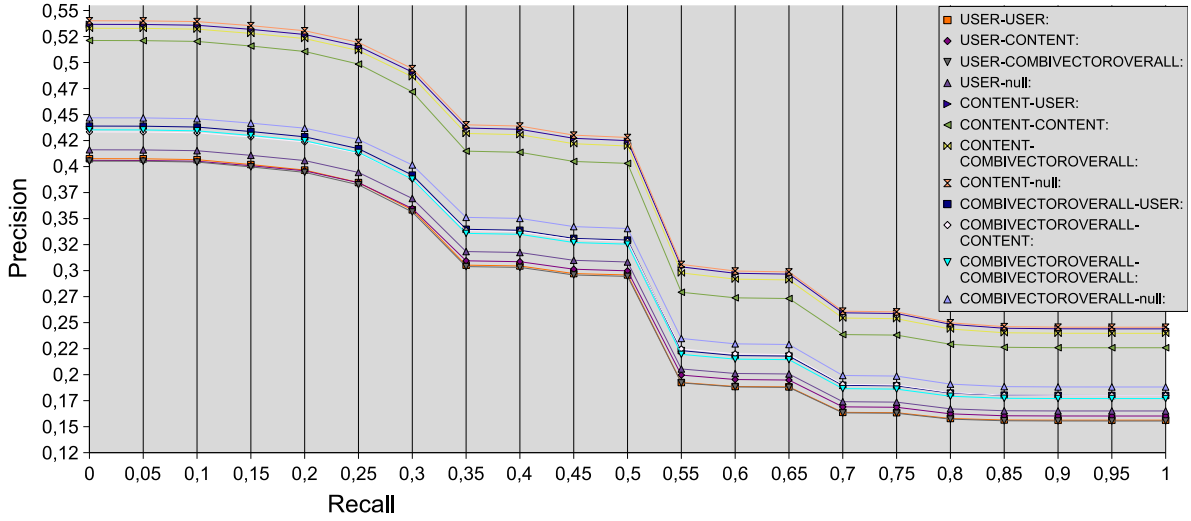

Abbildung 10: Kombination verschiedener Cosinus Ähnlichkeiten bei Tag-Expansionen von Inhalt und Benutzer bei Testdaten mit bekannten Inhalten

Für die Implikationswahrscheinlichkeitsmaße wurde nur noch mit *CONTENT*-Ähnlichkeit bei der inhaltsbasierten Expansion und den Ähnlichkeiten *USER*, *CONTENT* und *COMBIVECTOROVERALL* bei den benutzerbasierten Expansionen evaluiert, da sie sich ihren Konkurrenten stets als überlegen erwiesen hatten. Die Ergebnisse für den kompletten Testdatensatz sind in Abbildung 8 zu sehen. Anhand eines verringerten Testdatensatzes, welcher nur die Posts der ersten 50 Benutzer im Testzeitraum enthält, wurde bei der Entwicklung der semiasymetrischen Maße gerabeitet. Dabei konnte nur eine sehr geringe Verbesserung gegenüber der vollständig asymetrischen Implikationswahrscheinlichkeit festgestellt werden. Verbesserungen liefen in Richtung Symmetrie, weshalb im weiteren Verlauf des Projektes auf die CosinusÄhnlichkeit zurückgegriffen wurde. Die Ergebnisse der letzten Evaluation auf dem reduzierten Testdatensatz sind in Abbildung 9 zu sehen.
Das fast ausschließlich schlechtere Abschneiden der *CONTENT-USER* Recommendations gegenüber *CONTENT-null* Empfehlungen wirft Fragen auf, wie gut die Benutzerseitige Expansion Tags hervorbringt, welche ein Benutzer im Testzeitraum zum ersten Mal verwendet. Denn genau dies ist ihr Aufgabenbereich. Um dies zu testen wurde die Expansion auf jeden Benutzer, der im Testzeitraum neue Posts abgibt, angewandt und genau die Menge der von ihm zum ersten Mal verwendeten Tags als relevant eingestuft. Die Precision-Recall Kurve ist in Abbildung 11 zu sehen. Der Trivialansatz immer die von allen Benutzern insgesamt meistverwendeten Tags vorzuschlagen lieferte hier bereits bessere Ergebnisse.

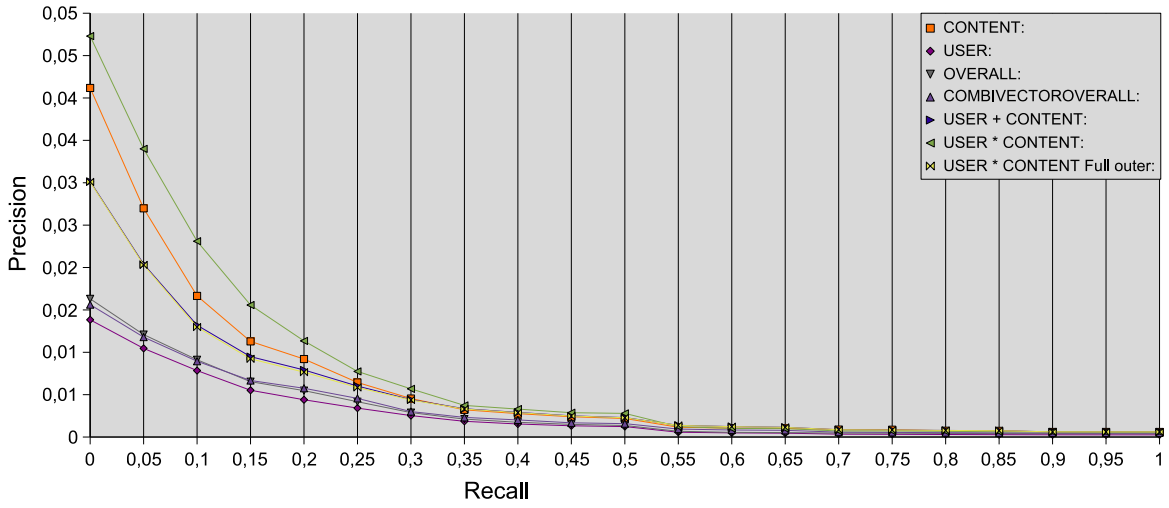

Abbildung 11: Evaluation benutzerseitiger Tag-Expansionen auf die Fähigkeit Tag-Erstverwendungen eines Benutzers vorherzusagen

4.3 Tagmapping

Das schlechte Abschneiden der benutzerbasierten Tag-Expansion hat zu einer einfacheren Darstellung geführt, die lediglich versucht Tags, welche an einem bereits bekannten Inhalt verwendet wurden, auf die schon verwendeten Tags des Benutzers, der den Inhalt neu taggen möchte, abzubilden. Man kann sich dies als ein einschichtiges neuronales Netz vorstellen, wobei jedes schon zuvor verwendete Tag des Benutzers als ein Neuron mit den k (gewählt wurde 30 nach [3]) ausgeprägtesten Synapsen zu anderen Tags repräsentiert wird. Die Neuronen mit der stärksten Aktivität setzen ihre Tags in die Recommendation-Liste. Werden weniger als die gewünschte Anzahl Tags vorgeschlagen, weil z.B. der Inhalt dem System noch unbekannt war, so wird die Liste mit den meistverwendeten Tags des Benutzers aufgefüllt.
Da der Fall, dass der zu taggende Inhalt dem System noch unbekannt ist, häufig vorkommt und auch die Menge der Tags eines Benutzers anfangs noch leer oder sehr klein ist, wird als Erweiterung des Verfahrens der Titel des Inhalts gelesen um die darin enthaltenen häufig bereits deskriptiven Worte wie Tags an einem bekannten Inhalt zu behandeln. Dabei werden Stopwörter entfernt und Worte sowohl in Groß- als auch Kleinschreibung verwendet. Die so extrahierten Wörter werden jedoch nur als halbes Tagvorkommen gewertet um wirkliche benutzergewählte Tags zu bevorzugen. Die extrahierten Tags werden zudem versucht in der Datenbank wiederzufinden um sie auf Tags des Nutzers abbilden zu können. Ein Nachteil des Verfahrens ist jedoch, dass die Auswahl von Tags eines Inhalts, die der Benutzer noch nie verwendet hat, benutzergruppenunspezifisch bleibt und sich lediglich an der Meistverwendung am Inhalt orientiert.
Das Verfahren ist effizient umsetzbar, denn es entsteht keine große Anzahl an Zwischenergebnissen entsteht, da beide Tags einer Ähnlichkeit von vornherein bekannt sind und einer Datenbank als zweidimensionaler Schlüssel geliefert werden können. Weiterhin brauchen nicht alle Tags eines Inhalts als Neuroneninputs verwendet zu werden, da die x meistverwendetsten Tags den Inhalt gut beschreiben und die meisten der relevanten Benutzertags ein solches Tag unter ihren ähnlichsten (stärksten Synapsen) aufweisen werden. Für die Expansion wurden daher nur die acht am Inhalt häufigst verwendeten Tags (ggf. um dem System bekannte Titelworte erweitert) benutzt. Die Ergebnisse sind in Abbildung 12 unter Verwendung von Cosinusähnlichkeit im Inhaltsvektorraum und gedrehten k Ähnlichstenlisten zu sehen. Bei gedrehten Ähnlichstenlisten haben die Neuronen nicht mehr grundsätzlich k stärkste Inputs, sondern soviele, wie oft ihr Tag unter den in Ähnlichstenlisten anderer Tags erscheint. Abbildung 13 zeigt den Vergleich zum gleichen Verfahren ohne gedrehte Ähnlichkeitslisten.
Die Implikationswahrscheinlichkeitsmaße kamen für dieses Verfahren wegen ihrer Neigung zu häufig verwendeten Tags auf der implizierten Seite nicht in Betracht, da daraus folgen würde, dass die mitunter seltenen benutzerspezifischen Tags eines Benutzers nicht mehr vorgeschlagen werden könnten.

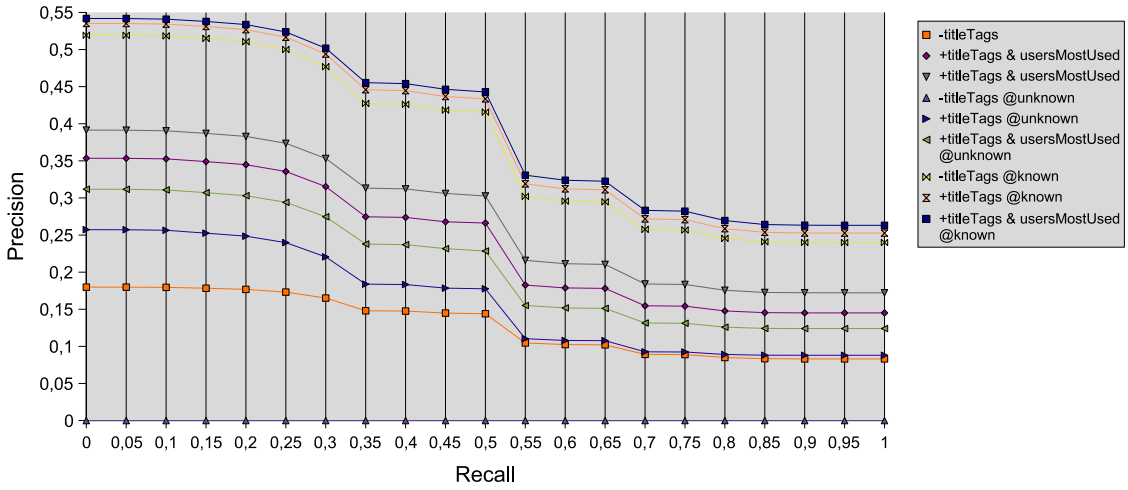

Abbildung 12: Evaluation des Tagmapping bei gedrehten Ähnlichstenlisten

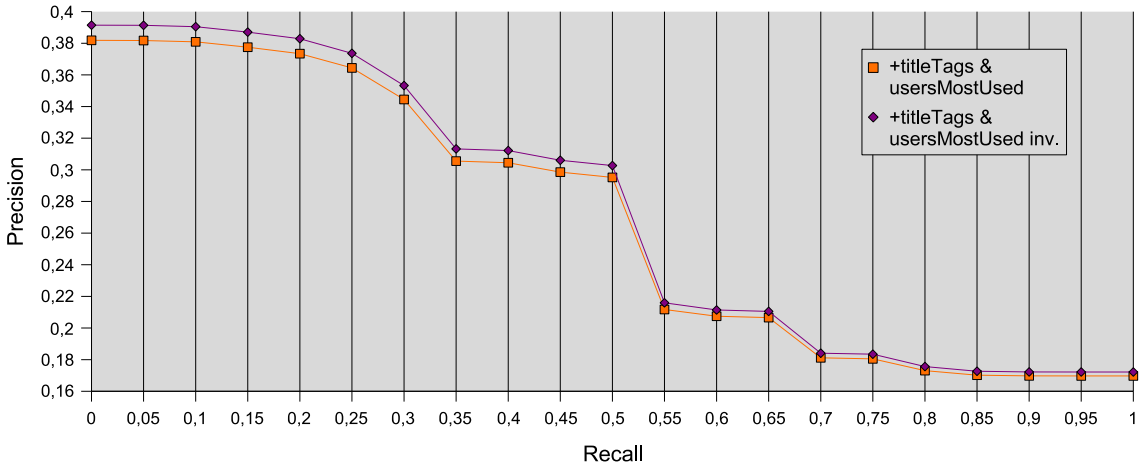

Abbildung 13: Vergleich zwischen Tagmapping bei gedrehten und ungedrehten Ähnlichstenlisten

5 Implementation

Die Implementation wurde im wesentlichen mit der Programmiersprache Java durchgeführt. Verwendung fanden neben den üblichsten Standard-Java5-APIs auch Servlets, JSPs, JSTL, Log4J, Tomcat, commons-dbcp und JUnit. Einige sehr umfangreiche Gruppierungen bei den Umwandlungen des Datenbankmodells wurden zudem über Perl-Skripte gelöst. Datenbankanfragen wurden in mySQL 5 kompatiblem SQL formuliert. Die Benutzeroberfläche verwendet HTML und Javascript.
Als Recommender-Verfahren wurde das Tagmapping mit invertierten Ähnlichstenlisten basierend auf Cosinusähnlichkeiten im Inhaltsvektorraum verwendet.

5.1 Aufbau der k nächsten Nachbarlisten

Bei der Implementation wurden die zweidimensionalen Matrizen verwendet, welche die Tag-Dimension enthalten. Mittels der Perl-Skripte tagContent.pl wurde die Datenbank Tabelle TagContent aufgebaut, welche zu jedem Tag alle Vektorkomponenten zur Verwendungshäufigkeit des Tags an Bookmarks und an BibTeX-Einträgen enthält, die nicht 0 sind. Gleiches wurde mit tagUser.pl und Tabelle TagUser für die Tag-Vektoren mit Userverwendungsanzahl-Komponenten getan. Aus diesen beiden Tabellen wurden in mehreren per Multithreading parallel laufenden Durchläufen zu jedem Tag die 30 nächsten Nachbarn berechnet. Dabei wird pro Iteration für eine durch den Hauptspeicher begrentzte Menge an Tagvektoren in einen invertierten Index gesetzt und über die komplette Tabelle aller Tagvektoren iteriert um das (bei Implikationswahrscheinlichkeiten modifizierte) Skalarprodukt und darauf aufbauend die Ähnlichkeit zwischen den iterierten Tags und den Tags im Index zu bilden. Beim durchlauf werden die Ähnlichstenlisten der Tags im Index aktualisiert. Diese Aufgabe übernimmt die Klasse

SimpleRecalculateMostSimTags. Sie ist als Subklasse von DatabaseAction implementiert.

5.2 Datenbankzugriffsschicht

Die Zugriffe auf die Datenbank wurden in einer Datenbankschicht abstrahiert. Dies hat den Vorteil, dass der DBCP Connection Pool und die Iteration über Ergebnismengen, sowie deren Freigabe nicht mehr innerhalb der Logikfunktionen erscheinen müssen. Kern der Abstraktion ist die Klasse Database, welche in ihrem Konstruktor bereits eine von ihrer eingebetteten Klasse DatabaseOperation abgeleitete Datenbankaktion ausführt. Anfragen und Kommandos sind als Subklassen von DatabaseOperation implementiert und kapseln teils dynamisch erzeugte SQL-Statements und deren Parameter- und Ergebnismengenverwaltung.

5.3 Updateprozess für Ähnlichstenlisten

Um die Ähnlichstenlisten von Tags aktuell zu halten müssen alle Ähnlichkeiten aller seit der letzten Aktualisierung verwendeten Tags neu berechnet werden. Zwar wurde ins Auge gefasst eine Online-Aktualisierung umzusetzen, jedoch können nicht alle Skalarprodukte (bis zu $0,5 \cdot 550000^2$) effizient gespeichert werden um daraus und aus den leicht speicherbaren Zweinormen der Tagvektoren bei Tagverwendung sofort eine neue Ähnlichkeit errechnen zu können. Werden zu jedem Tag nur die k stärksten Ähnlichkeiten gespeichert, kann zwar sehr schnell errechnet werden, wie stark sich eine Ähnlichkeit verändern wird, jedoch nicht auf welchen neuen absoluten Wert die Ähnlichkeit steigen wird und ob sie in die Liste der ähnlichsten Tags eintreten wird. Verfahren auf Basis einer Dreiecksungleichung wie zu Beginn von [1] beschrieben wurden ebenfalls in Betracht gezogen, jedoch eignen sie sich nicht für k nächste Nachbarn, da aufgrund des teils hohen Abstands zum Ausgangstag die von der Ähnlichstenliste beschriebene Umgebung des Ausgangstags zu groß wird um durch das Verfahren noch einen Vorteil erreichen zu können.
Somit wurde das Updatesystem in das Neuberechnungssystem integriert. Unterschiede bestehen bei Verwendung als Updatesystem darin, dass nicht die Ähnlichkeiten aller Tags zu allen anderen, sondern nur die Ähnlichkeit der veränderten Tags zu allen anderen berechnet werden und dabei die aufgrund der Cosinus-Metrik symmetrischen Ähnlichkeiten auch in den Listen, welche das Tag als nächten Nachbarn referenzieren mit aktualisiert werden. Durch Entfernen von Tags von Inhalten oder häufiges disjunktes Verwenden von zuvor stark korrelierten Tags können Ähnlichkeiten zu veränderten Tags so weit sinken, dass sie aus den Nachbarlisten herausfallen und andere unveränderte Tags an ihre Stelle treten müssten. Daher wird zu jedem Tag neben den nächsten Nachbarn auch die Ähnlichkeit zu dem ähnlichsten Tag außerhalb der nächsten Nachbarn zum Zeitpunkt der letzten Komplettaktualisierung des Tags, zu dem die Ähnlichstenliste gehört, gespeichert. So kann, wenn mehrere Ähnlichkeiten einer Ähnlichstenliste durch "Fremdaktualisierung"unter diesen Wert fallen, auch eine Komplettaktualisierung dieser Ähnlichstenliste ausgelöst werden. Wichtig bei der Aktualisierung ist auch, dass während der länger dauernden Rechenvorgänge die Datenbank nicht mit zu vielen Festplatteseek-intensiven Anfragen belastet wird. Durch die Komplettaktualisierung mit Anfragen auf die komplette Tagvektor-Tabelle und Einzelanfragen der Ähnlichstenlisten über clustered Primary Keys wird Seeking minimiert.

5.4 Integration

Das Recommender Subsystem wurde soweit möglich getrennt vom restlichen Bibsonomy System in einem separaten Paket *recommender* mit Unterpaketen implementiert. An der Funktionalität des bestehenden Systems wurden die wesentlichen Änderungen in den Klassen *BibtexShowHandler* und *BookmarkShowHandler* zum Aufruf der Recommendation-Erstellung und dem Füllen der neuen Member-Liste *RecommendedTag* am *BibtexBean* bzw. *BookmarkBean* sowie in den dazugehörigen Klassen *BibtexHandler* und *BookmarkHandler* zur Pflege der Liste veränderter Tagvektoren vorgenommen. Daneben wurde die Klasse *DBTagManager* leicht modifiziert, so dass die Methode *deleteTags* die Liste der vorherigen am Inhalt stehenden Tags zurückliefert um unnötige doppelte Datenbankanfragen zum Löschen und Wiedereinfügen zu vermeiden. Die JSP-Seiten *edit_bibtex* sowie *edit_bookmark* wurden verändert um die Recommendations anzuzeigen und die aus dem Projekt-CVS-HEAD übernommenen javascript Funktionen zu verwenden.

6 Fazit und Ausblick

Die verschiedensten Möglichkeiten das Item-Based Collaborative Filtering auf Bibsonomy-Tag-Recommendations zu übertragen ließen im Verlauf des Projektes kaum Geradlinigkeit aufkommen. Es lassen sich weit mehr Ansätze zur Lösung des Recommendings finden als im Rahmen dieser Projektarbeit erprobt werden konnten. Daher kann man sicher von keiner ultimativen Lösung sprechen, was das Ergebnis natürlich an manchen Stellen etwas unzufriedenstellend macht. So zum Beispiel bei der aufwändigen Aktualisierung vorberechneter Daten oder der nicht in allen Fällen voll ausgenutzten Möglichkeiten des Vorschlagens von Tags ähnlicher Benutzer. Letzteres könnte durch Verwendung einer Kombination von Item- und User-Based Collaborative Filtering evtl. noch integriert werden. Nichtsdestotrotz bietet die vorgestellte Implementation eine recht gute Precision und in der Regel sehr zutreffende Taggingvorschläge. Auch bei sehr großen Datenbanken wie den zur Evaluation verwendeten Delicious-Daten bleibt das System ausreichend schnell um praktisch eingesetzt werden zu können. Das Kaltstartproblem des Collaborative Filtering Ansatzes wird durch den aus der Titel-Tagextraktion gebildeten Content-Boost angenehm vermindert.

7 Danksagung

Die Projektarbeit wurde von Herrn Dr. Andreas Hotho betreut. Ihm und allen beteiligten Mitarbeitern am Fachgebiet Wissensverarbeitung der Universität Kassel sei an dieser Stelle für Freundlichkeit und schnelle, ausgiebige Reaktionen auf Fragen, sowie schneller Bereitstellung der benötigten Rechnerressourcen gedankt.

Literatur

[1] V. Ramasubramanian and Kuldip K. Paliwal. An efficient approximation-elimination algorithm for fast nearest-neighbour search based on a spherical distance coordinate formulation. *Pattern Recognition Letters*, 13(7):471–480, 1992.

[2] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *CSCW*, pages 175–186, 1994.

[3] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, pages 285–295, 2001.

[4] C.J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 1979.