Spektrale Bloom-Filter für Peer-to-Peer Information Retrieval

Martin Eisenhardt, Wolfgang Müller, Andreas Henrich
LS AI 1, Universität Bayreuth
martin.eisenhardt@uni-bayreuth.de

Abstract: Spektrale Bloomfilter können als Zusammenfassung von Peer-Daten in verteiltem P2P-Retrieval verwendet werden. Dies ermöglicht eine messbar bessere Auswahl derjenigen Peers, an die eine Anfrage gestellt wird, und somit auch eine höhere Effizienz der Anfragebearbeitung. Experimentelle Daten belegen diese Verbesserung des Retrievals in P2P-Netzen.

1 Einführung

Bloomfilter [Bl70] sind eine komprimierte, verlustbehaftete Repräsentation von Mengen. Sie unterstützen effizient Abfragen, ob ein Element in einer Menge vorhanden ist. Sie sind verlustbehaftet: Anfragen können fälschlich positiv beschieden werden. Bloomfilter werden in einer großen Breite von Anwendungen eingesetzt, von verteiltem Caching [FCAB98] bis zu verteiltem IR in P2P-Netzen [CAN02].

Während klassische Bloomfilter nur die Zugehörigkeit oder Nicht-Zugehörigkeit eines Elements zu einer Menge speichern, sind Spektrale Bloomfilter (SBF) in der Lage, Multimengen zu repräsentieren. SBFs ermöglichen es, Anfragen nach der Häufigkeit von Elementen in der Multimenge zu bearbeiten. Hierbei ist garantiert, dass wenn ein SBF ausgibt, ein Element e sei n_e Male in S enthalten, e *höchstens* n_e Male in S vorkommt. Die von uns vorgeschlagene Modifikation der SBFs – die wir mit mSBF für *modifizierte SBF* bezeichnen wollen – unterscheiden sich hier in Details, die sie für ihre Anwendung in P2P-IR prädestinieren, und die in Abschnitt 3 beschrieben werden. Wenn auch innerhalb dieser Veröffentlichung nur von P2P-IR die Rede ist, sind die hier erarbeiteten Resultate auch im Hinblick auf Content Based Multimedia Retrieval nützlich. Für Näheres verweisen wir hier auf [MH03].

Das Papier ist wie folgt gegliedert: Zunächst beschreiben wir eine Methode zum P2P-IR auf Basis von Peer-Zusammenfassungen und deren Verteilung (Abschnit 2). Dort wird auch auf die Verwendung von SBFs eingegangen. In Abschnitt 3 erfolgt eine Beschreibung von Bloomfiltern, SBFs und mSBFs, Abschnitt 4 enthält Experimente zur Verwendung von mSBFs für P2P-IR.

2 P2P-IR mit Zusammenfassungen

P2P-IR bearbeitet (wie klassisches IR) Anfragen, die aus einer Menge von Anfragetermen bestehen. Das Resultat einer Anfrage ist dann eine nach Relevanz der Dokumente geordnete Liste von Dokumenten (Ranking). In P2P-IR muss zusätzlich vor dem Ranking der Dokumente bestimmt werden, welche Peers überhaupt die Anfrage bearbeiten müssen. Ohne diese Peer-Auswahl wird die Anfragebearbeitung sehr ineffizient, da jeder Peer per Zeiteinheit zu viele Anfragen erhält, und viele Peers Anfragen bearbeiten, zu denen sie nichts beizutragen haben.

Cuenca-Acuna und Nguyen [CAN02] schlagen vor, mit Hilfe von Peer-Zusammenfassungen auszuwählen, welche Peers eine Anfrage bearbeiten sollen. Als Peer-Zusammenfassung wählen sie Bloomfilter: Jeder Peer legt in einem Bloomfilter ab, welche Terme (Wörter) in mindestens einem der in ihm gespeicherten Dokumente enthalten sind. Diese Zusammenfassungen werden dann geeignet im P2P-Netz verteilt, so dass alle Peers Zusammenfassungen aller anderen Peers enthalten. Bei der Bearbeitung einer Anfrage wird ein Peer A_0 zunächst ein Ranking der anderen Peers A_i bilden, um zu erfahren, welche Peers $A_{k_1}, \ldots, A_{k_m}$ am wahrscheinlichsten etwas zum Ergebnis beizutragen haben. Danach wird er die Anfrage an diese Peers weitergeben, die Resultate sammeln und die besten Resultate dem Benutzer mitteilen.

Cuenca-Acuna und Nguyen verwenden hierbei klassische IR-Ähnlichkeitsmaße für Dokumente. Sie definieren die Ähnlichkeit zwischen einem Dokument D und einer Anfragetermmenge Q durch $Sim(Q, D) = \frac{1}{|D|} \cdot \sum_{t \in Q} w_{D,t} \cdot \mathit{IDF}_t$. Hierbei ist $|D|$ die Anzahl der Terme in Dokument D, $w_{D,t} = 1 + \log f_{D,t}$, $f_{D,t}$ die Häufigkeit des Terms t in D, f_t die Zahl der Dokumente, die t enthalten, $\mathit{IDF}_t = \log(1 + \frac{N_D}{f_t})$ die *inverse document frequency*, ein Maß für die Trennschärfe von t, und N_D die Anzahl der Dokumente.

Leider ist in dem gegebenen Szenario f_t nicht exakt zu berechnen: die einzelnen Peers können nur wissen, *ob* in anderen Peers ein Term enthalten ist. Sie können damit analog der *IDF* die *inverse peer frequency IPF* berechnen: Ist N_P die Zahl der Peers, und $N_{P,t}$ die Zahl der Peers, die t enthalten, so definieren sie $\mathit{IPF}_t = \log(1 + \frac{N_P}{N_{P,t}})$, ihr Distanzmaß wird also zu $Sim(Q, D) = \frac{1}{|D|} \cdot \sum_{t \in Q} w_{D,t} \cdot \mathit{IPF}_t$.

Cuenca-Acuna verwenden die Summe der IPF_t der Anfrageterme, um die Peer-Relevanz für einen Peer A bezüglich einer Anfrage Q zu bestimmen:

$$R_A(Q) \quad = \quad \sum_{t \in Q \wedge t \in BF(A)} \mathit{IPF}_t \tag{1}$$

Als Notation gilt hier, dass $t \in BF(A)$, wenn der Bloomfilter $BF(A)$ der Menge A (bzw. des Peers A) die Membership-Anfrage $t \in A$? positiv beantwortet.

Während in [CAN02] beeindruckende Resultate bei Tests mit langen Anfragen angegeben wurden, wird aus Eq. 1 klar, dass die Termfrequenz, also die *Häufigkeit eines Terms in den im Peer enthaltenen Dokumenten*, nicht berücksichtigt wird. Offensichtlich ist dies bei langen Anfragen durchaus zu verschmerzen, jedoch stellt diese Vereinfachung für die Bearbeitung von kurzen Anfragen, wie sie gerade von “normalen” Benutzern ohne IR-

Erfahrung abgesetzt werden, ein Problem dar [SHMM98]. Hier werden viele Peers gleiche Teile der Anfrage Q mindestens einmal enthalten. Die Peer-Auswahl nach Eq. 1 kann zwischen diesen nicht sinnvoll eine Reihenfolge festlegen.

SBFs für verbesserte Peer-Selektion: SBFs als Peer-Repräsentation erlauben die Weitergabe von Häufigkeitsinformation: anstatt in einem "normalen" Bloomfilter $BF(A)$ zu speichern, ob ein Term t in A vorhanden ist, verwenden wir einen SBF *SBF*(A), um zu speichern, *in wie vielen Dokumenten des Peers* A der Term t enthalten ist. Aus Eq. 1 wird

$$R_A(Q) \quad = \quad \sum_{\substack{t \in Q \\ \wedge \mathit{SBF}(A,\,t) > 0}} (1 + \log \mathit{SBF}(A,t)) \cdot \log\left(1 + \frac{N}{\sum_{\mathit{all}\ A_i} \mathit{SBF}(A_i,t)}\right) \qquad (2)$$

SBF(A,t) bezeichnet hierbei die Häufigkeit des Terms t gemäß dem Spektralen Bloomfilter von A und N die Gesamtzahl von Dokumenten.

Bevor nun in Abschnitt 4 Messungen und Ergebnisse präsentiert werden, die den Nutzen der Verwendung von SBFs in dieser Anwendung demonstrieren, wird im folgenden Abschnitt die Wirkungsweise von Bloomfiltern und ihre Erweiterung zu mSBFs beschrieben.

3 Von Bloomfiltern zu mSBFs

Ein Bloomfilter, der Teilmengen einer Menge $S = \{s_1, \ldots, s_n\}$ repräsentieren soll, ist ein Bitvektor $V = (v_0, \ldots, v_m)$ aus m Bits, die zunächst mit dem Wert 0 initialisiert sind. Ein Bloomfilter verwendet k unabhängige Hashfunktionen $h_1, \ldots, h_k$ mit Wertebereich $\{0, \ldots, m-1\}$. Wird nun ein Wert s in den Bloomfilter eingefügt, so werden in V für alle Hashfunktionen $h_i \in \{h_1, \ldots, h_k\}$ die Bits $v_{h_i(s)}$ auf 1 gesetzt. Bei einer Anfrage, ob ein $x \in S$ in V gespeichert ist, wird x als zu V zugehörig angesehen, wenn für alle h_i gilt $v_{h_i(x)} = 1$. Dies ist jedoch mit einem Fehler behaftet: es kann vorkommen, dass $v_{h_i(x)} = 1$ für alle h_i, obwohl $x \notin V$. Wir haben dann eine *Kollision (false positive)*. Wenn jedoch ein h_i mit $v_{h_i(x)} = 0$ exisitiert, ist zwingend $x \notin V$.

In spektralen Bloomfiltern enthält nun jede Komponente von V b Bits ($b \geq 0$). Beim Einfügen von s wird jetzt, die Stelle $v_{h_i(s)}$ (anstatt auf 1 gesetzt zu werden) inkrementiert. Es ist also möglich zu zählen, *wie häufig* s eingefügt wurde. Wenn man nun anfragen will, wie oft ein x in V eingefügt wurde, so genügt es, *SBF*$(V,x) = \min_{i=1}^{k} v_{h_i(x)}$ zu berechnen. *SBF*(V,x) ist dann *mindestens* die Anzahl der Male, die x in V eingefügt wurde. Die Maximale Häufigkeit, die mittels SBF dargestellt werden kann, ist natürlich $2^b - 1$. Im Gegensatz zu klassischen Bloomfiltern kann man aus SBFs auch löschen: wird s aus V entfernt, so dekrementiert man die $v_{h_i(s)}$.

Vorausgesetzt, dass keine Löschungen vorgenommen werden, kann das Verhalten von SBFs bei Kollisionen verbessert werden. Angenommen, es gäbe eine Kollision zwischen $x_0 \in S$ und $\{x_1, \ldots x_c\} \subset S$. Das hieße also $\cup_{i=0}^{k}\{h_i(x_0)\} \subset \left(\cup_{i=0}^{k}\{h_i(x_1), \ldots h_i(x_1)\}\right)$. Dann enthält $v_{h_i(x_0)}$ jeweils die Anzahl der Einfügungen von x_0 *zuzüglich* der Anzahl der

Einfügungen von mindestens einem $x \in \{x_1, \ldots x_c\}$. Eine solche Kollision tritt zwar nur dann auf, wenn auch eine Kollision in einem klassischen Bloomfilter auftreten würde; dies aber kann die Anzahl stark verfälschen. Bezeichnen wir mit $\#x$ die Häufigkeit mit der x wirklich eingefügt wurde, so ist bei einer Kollision $SBF(V, x_0) \geq \#x_0 + \min_{i=1}^{c} \#x_i$.

Um SBFs nun in unserem IR-Kontext sinnvoll einsetzen zu können, modifizieren wir sie so, dass sie das *einmalige Einfügen eines mit einem Element $s \in S$ assoziierten Integerwertes*, sowie Häufigkeitsabfragen bezüglich s unterstützen. Anstatt beim n-maligen Einfügen von x die $v_{h_i(x)}$ n-mal zu inkrementieren, fügen wir ein x mit der Häufigkeit n nun in den Bloomfilter ein, indem wir die $v_{h_i(x)}$ gemäß $v_{h_i(x)} \leftarrow \max(v_{h_i(x)}, n)$ modifizieren. Ist also der Wert von $v_{h_i(x)}$ bereits gleich oder größer n, so bleibt er unverändert. Anderenfalls wird er zu n gesetzt. $v_{h_i(x)}$ gibt somit Auskunft über die maximale Häufigkeit, mit der ein Term y mit $v_{h_i(y)} = 1$ in V vorkommt. Diese Vorgehensweise reduziert den Fehler bei Kollisionen im Vergleich zu SBFs. Die entsprechenden Bloomfilter nennen wir mSBFs. mSBFs eignen sich exzellent zur Darstellung von Termhäufigkeiten in Peer-Zusammenfassungen. Ihr einziger Nachteil ist, dass Löschungen nun nicht mehr zu einer Reduktion der $v_{h_i(x)}$ genutzt werden können.

4 Experimente

In unseren Experimenten untersuchten wir die PlanetP-artige Anfragebearbeitung in einem Netz von 1000 Peers. Jeder Peer enthielt 100 zufällig zugeteilte Dokumente der Reuters-Collection. Kein Dokument war im Netz zweimal vorhanden. Die Peer-Zusammenfassungen waren Bloomfilter und mSBFs mit je 22000 Einträgen. Im Gegensatz zu den Experimenten in [CAN02] verwendeten wir kurze, aus 2 oder 3 Termen bestehende Anfragen. Dies ist konsistent mit Daten aus Web-Suchmaschinen [SHMM98]. Von unseren 40 Testanfragen bestanden 30 aus drei Suchtermen, 10 aus zwei Suchermen.

Da durch die Konstruktion der mSBF klar ist, dass diese in keinem Fall kleinere Werte und damit schlechtere Abschätzungen der Anzahl der Dokumente mit einem bestimmten Term auf einem bestimmten Peer liefern, als SBFs, haben wir uns in den Messungen zunächst darauf konzentriert, das grundsätzliche Potential des Ansatzes bei kurzen Anfragen zu überprüfen. Wir vergleichen unseren Ansatz also hier mit einer zufälligen Peer-Auswahl.

Jede Anfrage wurde zunächst einer zentralen, nicht verteilten Suchmaschine übergeben; Resultat waren IDs der 20 Dokumente mit dem höchsten Rang (in der Folge: M_{20}). Danach wurden die Peers gerankt (Eq. 2). Nun wurden die Peers gemäß ihres Peer-Rangs nacheinander kontaktiert, und für jedes Dokument der M_{20} wurde notiert, im wievielten Peer es gefunden wurde. Der Median der Peer-Ränge der M_{20} wurde dann aufgezeichnet.

Entsprechende Experimente wurden für verschiedene Bittiefen der mSBF ($1, 2, 4, 6, 8$ Bits) durchgeführt. Die Ergebnisse sind in Fig. 1 zu sehen. Wichtig zur Einordnung der Resultate ist die Feststellung, dass man bei rein zufälligem Peer-Ranking im Durchschnitt die Hälfte der Peers (normierter Rang = 0.5) betrachten müsste um wenigstens die Hälfte der M_{20} zu finden. Das heißt, schon ein Peerranking auf Basis von 1-Bit Bloomfiltern — dies entspricht dem Ansatz aus [CAN02] — senkt die Kosten der Anfragebearbeitung

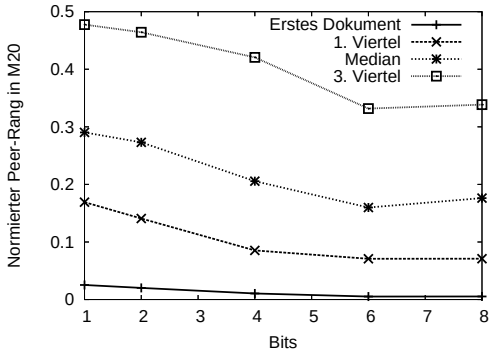

Abbildung 1: Vergleich der Kosten von PlanetP-Anfragen mit herkömmlichen Bloomfiltern (1 Bit) und mSBFs mit Bitanzahlen von 2-8.

bezüglich einem rein zufälligen Peer-Ranking um 42%. 6-Bit mSBFs bewirken eine weitere Verbesserung. Gegenüber rein zufälligem Peer-Ranking werden gar 68% eingespart, gegenüber herkömmlichen Bloomfiltern 44%.

Zusammenfassung und Ausblick: In dieser Publikation wurde eine Variante der Spektralen Bloomfilter vorgestellt, die mSBFs. Es wurde experimentell gezeigt, dass im P2P-IR durch Verwendung von mSBFs eine zielführende Peer-Auswahl möglich ist. Die bisher erzielten Resultate sind allerdings nicht ganz so überzeugend wie die Resultate in [CAN02] und unsere Experimente über P2P-Content Based Image Retrieval [MH03] erwarten lassen. Wir führen dies unter anderem auf die wesentlich längeren Anfragen in [CAN02] zurück. Durch die Kombination der in [CAN02] und [MH03] verwandten Techniken hoffen wir, die Leistungsfähigkeit des Netzwerkes für kurze Anfragen weiter zu verbessern.

Literatur

[Bl70] Bloom, B.: Space/time tradeoffs in hash coding with allowable errors. *CACM*. 13(7):422–426. 1970.

[CAN02] Cuenca-Acuna, F. M. und Nguyen, T. D.: Text-Based Content Search and Retrieval in ad hoc P2P Communities. Technical Report DCS-TR-483. Department of Computer Science, Rutgers University. 2002.

[FCAB98] Fan, L., Cau, P., Almeida, J., und Broder, A.: Summary cache: A scalable wide-area Web cache sharing protocol. In: *SIGCOMM '98*. Vancouver, BC, Canada. 1998.

[MH03] Müller, W. und Henrich, A.: Fast retrieval of high-dimensional feature vectors in p2p networks using compact peer data summaries. In: *ACM MIR'03 Workshop*. Berkeley, CA,USA. 2003.

[SHMM98] Silverstein, C., Henzinger, M., Marais, H., und Moricz, M.: Analysis of a very large altavista query log. Technical Report 1998-014. Digital SRC. 1998.