Noname manuscript No.
(will be inserted by the editor)

Projektseminar „Similarity Search Algorithms“

Dustin Lange · Tobias Vogel · Uwe Draisbach · Felix Naumann

Received: date / Accepted: date

Zusammenfassung Mithilfe von Verfahren aus dem Bereich Ähnlichkeitssuche können zu einer Anfrage an einen Datenbestand nicht nur exakte, sondern auch ähnliche Objekte gefunden werden, z. B. Bilder mit ähnlichen Motiven wie auf dem Anfragebild. Mit aktuellen Forschungsansätzen aus diesem Bereich befasste sich das Seminar „Similarity Search Algorithms“, welches wir in diesem Bericht vorstellen.

Das Ziel des Seminars war ein breiter Vergleich bekannter Indexierungsalgorithmen mit Datensätzen aus verschiedenen Bereichen. Die Studenten befassten sich mit je zwei Ähnlichkeitsmaßen für Datensätze aus fünf verschiedenen Domänen und mit je einem von sechs verschiedenen Indexstrukturen zur Ähnlichkeitssuche in metrischen Räumen. In diesem Bericht evaluieren wir die Kombination der Ähnlichkeitsmaße mit den Indexstrukturen bzgl. Indexaufbau und knn-Anfragen. Außerdem beschreiben wir die Durchführung des Seminars und werfen einen Blick auf *lessons learned*.

Schlüsselwörter Ähnlichkeitssuche · Indexstrukturen · Lehre

1 Einleitung

Ähnlichkeitssuche beschreibt Verfahren, die über die in DBMS implementierte exakte Suche hinausgehen und Nutzern ermöglichen, zu einem Anfrageobjekt ähnliche Objekte zu finden. Ein Anwendungsgebiet hierfür ist die Suche nach ähnlichen Bildern zu einem gegebenen Anfragebild; das können etwa Bilder mit den gleichen Motiven oder Farben sein. Ein weiteres Beispiel stellt das Auffinden von ähnlichen Tupeln zu einem Anfragetupel in einer Tabelle dar. Zu einer Anfrage an eine Personendatenbank nach *(Hans Meyer, Berlinn)* sollen auch Tupel mit leichten Abweichungen wie *(Hans Meyer, Berlin)* oder *(Hans Mayer, Berlin)* gefunden werden. Das im Sommersemester 2010 im Rahmen des Masterstudiums am Hasso-Plattner-Institut durchgeführte Projektseminar „Similarity Search Algorithms“ behandelte aktuelle Forschungsansätze aus dem Bereich Ähnlichkeitssuche.

Für eine effektive und effiziente Ähnlichkeitssuche werden sowohl ein Ähnlichkeitsmaß als auch eine Indexstruktur für den Zugriff auf die Elemente im Datenbestand benötigt. Durch geschickte Ausnutzung der Eigenschaften der Ähnlichkeitsmaße werden Vergleiche in einer Indexstruktur vorberechnet und zur Anfragezeit entsprechend eingespart. Das Seminar behandelte Ansätze für den metrischen Raum, für den verschiedene Ideen zur Indexstruktur-Erstellung in der Literatur beschrieben sind [26, 29].

Im folgenden Abschnitt 2 werden die Datensätze mitsamt entwickelter Ähnlichkeitsmaße sowie die Indexstrukturen vorgestellt. In Abschnitt 3 beschreiben wir die Aufgabenstellung und den Ablauf des Seminars. Abschnitt 4 enthält eine Evaluierung der im Seminar erstellten Implementierungen. Wir beschreiben unsere Erfahrungen und gewonnenen Erkenntnisse in Abschnitt 5.

Hasso-Plattner-Institut an der Universität Potsdam
Prof.-Dr.-Helmert-Str. 2-3, 14482 Potsdam, Deutschland
E-Mail: vorname.nachname@hpi.uni-potsdam.de
Web: `http://www.hpi-web.de/naumann/`

2 Datensätze und Ähnlichkeitsstrukturen

Die im Seminar behandelten **Datensätze** stammten aus verschiedenen Domänen und unterschieden sich sowohl in ihrer Struktur als auch in ihrem Umfang. Dies ermöglichte eine Evaluierung dieser Unterschiede mit den verschiedenen Indexstrukturen. Die verwendeten Datensätze umfassten:

- *CDs*: ca. 1,7 Mio. CD-Tupel extrahiert von FreeDB [15] (Schema: Künstler, Titel, Genre, ...)
- *Adressen*: ca. 1 Mio. generierte Adresstupel (Schema: Name, Anrede, Adresse, ...)
- *Fließtext*: Kurzzusammenfassungen aller ca. 466.000 dt. Wikipedia-Artikel (DBpedia [2], Stand 2010)
- *Bilder*: ca. 10.000 Thumbnails aus dem Corel-Datensatz [16]
- *Prozessdiagramme*: ca. 600 Event-driven-Process-Chain-Diagramme aus dem SAP-Referenzmodell [7, 12] in EPML

Jedes studentische Team entwickelte (je nach Teamgröße) Ähnlichkeitsmaße für 1-2 Datensätze. Im Folgenden beschreiben wir kurz die somit entstandenen 10 **Ähnlichkeitsmaße**:

- *CD 1*: Es wird der gewichtete Durchschnitt der Ähnlichkeit der Attribute Künstler, Titel, Genre, Sprache, Anzahl Lieder, Veröffentlichungsdatum und Liedtitel 1-3 berechnet. Als Ähnlichkeitsmaße werden Levenshtein-Distanz [13], Jaro-Winkler-Ähnlichkeit [27], Smith-Waterman-Gotoh-Distanz [9], Soundex [23] und Monge-Elkan-Ähnlichkeit [18] verwendet.
- *CD 2*: Es wird der gewichtete Durchschnitt der Ähnlichkeit der Attribute Künstler, Titel und Liedtitel 1-3 mit den Ähnlichkeitsmaßen Levenshtein-Distanz und Double Metaphone [22] berechnet, nachdem Stoppwörter entfernt wurden. Zusätzlich werden weitere Attribute auf Gleichheit geprüft.
- *Adressen 1*: Die Ähnlichkeit zweier Adressen ist der gewichtete Durchschnitt der Attribut-Ähnlichkeit von Anrede, Titel, Vorname, Nachname, Strasse, Hausnummer, Postfach, Postleitzahl und Telefon. Als Ähnlichkeitsmaße werden Levenshtein-Distanz, die Jaccard-Ähnlichkeit [10] basierend auf 3-Grammen und die Kölner Phonetik [24] verwendet. Die Attributwerte wurden teilweise in ein einheitliches Format überführt und mit anderen Attrbibutwerten kombiniert.
- *Adressen 2*: Die Ähnlichkeit ist der gewichtete Durchschnitt der Ähnlichkeit der Attribute Name, Straße, Ort und Postleitzahl. Als Ähnlichkeitsmaße werden SoftTFIDF [6], die Jaro-Winkler-Ähnlichkeit und die Kölner Phonetik verwendet.
- *Fließtext 1*: Dieses Maß vergleicht 2 Texte, indem zunächst die Stoppwörter entfernt werden und die verbliebenen Wörter normalisiert werden (einfaches Stemming). Aus der Häufigkeitsverteilung der Wörter werden Vektoren gebildet, die mittels Cosinus-Ähnlichkeit verglichen werden.
- *Fließtext 2*: Nach dem Entfernen von Stoppwörtern werden Keywords extrahiert (alle Nomen). Von diesen Wörtern wird ein Hash-Wert mittels SimHash [25] berechnet. Die Anzahl der Übereinstimmungen in den Hash-Werten wird ins Verhältnis gesetzt zu deren Länge.
- *Bilder 1*: Dieses Ähnlichkeitsmaß berechnet für disjunkte quadratische Kacheln eines Bildes ein Histogramm und vergleicht dieses mit dem entsprechenden Histogramm eines zweiten Bildes mit Pearsons empirischem Korrelationskoeffizienten [21]. Dadurch wird die Verteilung der Farben auf der Bildfläche berücksichtigt.
- *Bilder 2*: Zwei Bilder werden mit dem Ansatz von Liu et al. [14] verglichen. Bilder werden in eine quadratische Form transformiert, ihre Histogramme ausgeglichen, in den YIQ-Farbraum konvertiert [11] und in einen $64 \cdot 64 \cdot 3$-dimensionalen Vektor transformiert. Dieser wird auf einen 100-dimensionalen Vektor reduziert [1] und dann mit dem euklidischen Abstand verglichen.
- *Prozessdiagramme 1*: Die Labels aller Knoten der beiden Diagramme werden tokenisiert und mit der Hausdorff-Distanz [20] verglichen.
- *Prozessdiagramme 2*: Die Labels aller Knoten der beiden Diagramme werden tokenisiert. Dann werden alle Elemente der beiden von Duplikaten befreiten Tokenmengen mit der Levenshtein-Distanz verglichen und mit dem Ungarischen Algorithmus [19] ein Mapping erstellt. Der Durchschnitt der dort ausgewählten Kantengewichte bildet die Prozessähnlichkeit.

Als zu implementierende **Indexstrukturen** wählten wir bekannte und bewährte Verfahren aus den vergangenen Jahren:

- D-Index (Distance Index) [8],
- GNAT (Geometric Near-neighbor Access Tree) [4],
- LAESA (Linear Approximating Eliminating Search Algorithm) [17],
- M-Tree (Metric Tree) [5],
- VP-Tree (Vantage Point Tree) [28] und
- MVP-Tree (Multi Vantage Point Tree) [3].

3 Seminarablauf

Im Vorfeld des Seminars wurde von den Betreuern ein Framework in Java entwickelt, das die Kombination der verschiedenen Indexstrukturen mit den verschiedenen Datensätzen erlaubt. Das Framework war zunächst als grobe Vorgabe konzipiert und sollte von den Studenten im Laufe des Seminars weiter verfeinert werden. Zusätzlich wurde eine Webanwendung entwickelt, die die Ergebnisse einer Ähnlichkeitssuche veranschaulicht. Der gesamte Quellcode wurde den Studenten in einem Subversion-Repository zur Verfügung gestellt. Ein zusätzliches Trac-Wiki diente der Dokumentation.

Das Seminar belegten 10 Studenten, die in 6 Gruppen mit je 1-2 Personen aufgeteilt wurden. Jeder Gruppe wurden eine Indexstruktur und jeweils 1-2 Datensätze zugewiesen.

Zunächst bestand die Aufgabe in der Entwicklung geeigneter Ähnlichkeitsmaße für ihre Datensätze. Hierfür mussten entsprechende Literatur gesichtet und anschließend die Ähnlichkeitsmaße nach den Framework-Vorgaben implementiert werden. Auf diese Weise entstanden für jeden Datensatz zwei verschiedene Ähnlichkeitsmaße.

Im zweiten Schritt stellten die Studenten eines der ausgewählten Paper zur jeweiligen Indexstruktur vor und implementierten die darin beschriebenen Algorithmen. Neben dem eigentlichen Algorithmus zum Indexaufbau mussten auch Algorithmen für Bereichs- und k-nächste-Nachbarn-Anfragen (knn-Anfragen) implementiert werden. Hierbei mussten die Studenten Anpassungen des Frameworks untereinander diskutieren und eine gemeinsame Entscheidung über die Schnittstellen treffen. Neben der Nachimplementierung der beschriebenen Algorithmen zum Indexaufbau und zu Indexanfragen mussten sich die Studenten auch Gedanken über die Persistierung und das Laden existierender Indexstrukturen machen.

Im dritten und letzten Schritt evaluierten die Studenten ihre Indexstrukturen anhand ihrer Implementierungen. Dazu verwendeten sie ihre eigenen sowie mindestens ein fremdes Ähnlichkeitsmaß. Damit erfolgte gleichzeitig eine Evaluierung des Frameworks, denn nur durch eine korrekte Implementierung der Schnittstellen war eine Verarbeitung verschiedener Datensätze mit Hilfe verschiedener Indexstrukturen möglich.

Die Ergebnisse der Gesamtevaluierung werden in Abschnitt 4 beschrieben.

4 Evaluierung

Die Gestaltung des Frameworks für die Studenten war so konzipiert, dass alle Indexstrukturen mit allen Ähnlichkeitsmaßen kompatibel sein sollten. Dies ermöglichte es uns nach Abschluss des Seminars, eine Evaluierung der Indexstruturen und Ähnlichkeitsmaße durchzuführen. Hierfür haben wir auf einem dedizierten Rechner für jede Ähnlichkeitsmaß-Indexstruktur-Kombination einen Index erstellt und dann für jeden erstellen Index knn-Anfragen durchgeführt. Die Ergebnisse beschreiben wir im Folgenden. Am Ende des Abschnitts beschreiben wir außerdem kurz die webbasierte Anfrageschnittstelle, mit der alle implementierten Ähnlichkeitsmaße, Indexstrukturen und Anfragearten getestet werden können.

4.1 Indexaufbau

Zunächst haben wir die Komplexität des Indexaufbaus gemessen. Dafür haben wir die für den Indexaufbau benötigten Vergleiche (d. h. die Aufrufe des Ähnlichkeitsmaßes) gezählt. Die Ergebnisse für alle Kombinationen sind in Tabelle 1 aufgelistet. Die Tabelle enthält als Baseline-Index mit *Naive* eine einfache Liste aller Elemente im Datensatz, für deren Erstellung keinerlei Vergleiche nötig sind.

Die Ergebnistabelle zeigt, dass die Anzahl der benötigten Vergleiche von verschiedenen Faktoren abhängig ist. Zunächst fällt die große Diskrepanz zwischen den Zahlen der oberen Hälfte und der unteren Hälfte auf. Die Datensätze in der oberen Hälfte enthalten erheblich mehr Elemente, was natürlich zu einer erheblich größeren Anzahl an Vergleichen führt. Beispielsweise enthält der CD-Datensatz 3.000-mal so viele Elemente wie der Prozess-Datensatz, und es werden durchschnittlich 7.600-mal so viele Vergleiche benötigt. Es ist jedoch keine Schlussfolgerung über die Komplexität der Verfahren allein in Abhängigkeit von der Anzahl der Elemente möglich, da jeweils unterschiedliche Datensätze verglichen werden.

Durch die verschiedenen Ähnlichkeitsmaße pro Datensatz wird die Abhängigigkeit der Indexstruktur vom Ähnlichkeitsmaß ersichtlich. So unterscheiden sich beispielsweise die Ergebnisse der beiden Bilder-Ähnlichkeitsmaße für den Index M-Tree ganz erheblich. Ein Grund hierfür sind die unterschiedlichen Verteilungen der Ähnlichkeitswerte der implementierten Ähnlichkeitsmaße.

4.2 Anfrageausführung

Für jeden der erfolgreich erstellten Indizes haben wir in einem weiteren Experiment die Anfrageausführung evaluiert. Dazu wurden für jedes Ähnlichkeitsmaß 10 Objekte zufällig aus dem Datensatz ausgewählt und als

Tabelle 1 Benötigte Anzahl an Vergleichen für die Indexerstellung für jede Ähnlichkeitsmaß-Indexstruktur-Kombination. „Exception“ beschreibt während der Erstellung aufgetretene (nicht triviale) Fehler. Mit „Time-out“ werden Ausführungen gekennzeichnet, die wir nach einem Tag Laufzeit abgebrochen haben.

Datensatz	# Elem.	Naive	D-Index	GNAT	LAESA	M-Tree	MVP-Tree	VP-Tree
Addressen 1	1.039.776	0	219.780.028	Time-out	10.397.705	Time-out	Time-out	18.711.434
Addressen 2	1.039.776	0	146.375.091	Time-out	10.397.705	Time-out	Time-out	161.797.563
CD 1	1.847.233	0	Exception	169.679.699	18.472.275	424.991.371	36.391.258	36.148.740
CD 2	1.847.233	0	669.160.354	173.703.721	18.472.275	404.026.753	36.391.258	35.201.495
Fließtext 1	466.441	0	175.179.543	53.158.001	4.664.355	88.247.702	8.257.596	7.938.374
Fließtext 2	466.441	0	77.066.116	Time-out	4.664.355	Time-out	Time-out	8.031.446
Bilder 1	9.908	0	Exception	630.188	Exception	800.612	116.742	Exception
Bilder 2	9.908	0	386.435	3.596.710	99.025	2.798.212	116.742	112.445
Prozessdiagr. 1	603	0	26.716	26.384	5.985	68.423	4.702	4.466
Prozessdiagr. 2	603	0	32.314	24.496	5.985	76.464	4.702	4.466

Tabelle 2 Benötigte Anzahl an Vergleichen für knn-Anfragen ($k = 5$) für jede Ähnlichkeitsmaß-Indexstruktur-Kombination. „Exception“ beschreibt während der Erstellung aufgetretene (nicht triviale) Fehler.

Datensatz	# Elemente	Naive	D-Index	M-Tree	MVP-Tree	VP-Tree
CD 1	1.847.233	1.847.233	Exception	940.079	1.145.036	1.316.482
CD 2	1.847.233	1.847.233	1.814.506	1.578.158	1.653.843	1.687.491
Fließtext 1	466.441	Exception	467.391	463.845	462.920	Exception
Bilder 1	9.908	9.908	Exception	870	Exception	Exception
Bilder 2	9.908	9.908	11.187	9.945	9.908	Exception
Prozessdiagramme 1	603	603	702	567	496	500
Prozessdiagramme 2	603	603	800	568	562	541

Anfrageobjekte für knn-Anfragen ($k = 5$) verwendet. In Tabelle 2 sind die für eine Anfrage durchschnittlich benötigten Ähnlichkeitsmaß-Aufrufe dargestellt. Da für die Anfragen kein Goldstandard vorlag, erfolgte keine Analyse der Korrektheit der Ergebnisse. Allerdings konnten fehlerhafte Ergebnisse durch Majority Voting festgestellt werden, da alle Indexstrukturen für eine Anfrage mit demselben Ähnlichkeitsmaß dieselben Antworten liefern müssen. Indexstrukturen und Ähnlichkeitsmaße, die ausschließlich fehlerhafte Ergebnisse lieferten oder andere Fehler zur Laufzeit verursachten, sind in der Tabelle nicht aufgeführt.

Auch bei den Anfragen sind Unterschiede zwischen großen Datensätzen (wie CDs und DBpedia) und den kleineren Datensätzen erkennbar. Natürlich müssen bei erheblich umfangreicheren Datensätzen auch mehr Vergleiche durchgeführt werden.

Beim Vergleich der Ähnlichkeitsmaß-Aufrufe mit der Anzahl der Elemente fällt jedoch auf, dass die Indexstrukturen nur selten Vorteile gegenüber dem Durchlaufen des kompletten Datensatzes (*Naive*-Spalte) bieten. Deutliche Vorteile zeigt z. B. der M-Tree bei CDs (CD 1) und Bildern (Bilder 1). Viele der anderen Indexstrukturen bringen nur geringe Einsparungen. Eine Ursache hierfür sind die Implementierungen der knn-Suche. Da in den zugrundeliegenden Papern in einigen Fällen (GNAT, LAESA, MVP-Tree und VP-Tree) die knn-Anfrageausführung nicht beschrieben ist, war es Aufgabe der Studenten, eigene Implementierungen zu entwickeln. Hierbei ergab sich die Schwierigkeit, dass einige Indexstrukturen generell besser für Bereichsanfragen geeignet sind und knn-Anfragen nur mithilfe von Bereichsanfragen nachgebaut werden können. Um vollständige Ergebnislisten zu erzeugen, müssen Bereichsanfragen mehrfach und mit unterschiedlichen Parametern durchgeführt werden.

In manchen Fällen (etwa beim D-Index) werden in einer einzigen Anfrage sogar viele Elemente mehrfach herangezogen, sodass mehr Vergleiche benötigt werden, als Elemente im Datensatz vorhanden sind. Dies kann durch eine ungeeignete, redundante Aufteilung der Elemente im Baum verursacht werden. Wie ein studentisches Team ermittelt hat, kann eine bessere Einstellung der Index-Parameter in Abhängigkeit von den Distanzverteilungen diesen Effekt verringern.

4.3 Webanwendung

Die im Vorfeld des Seminars entwickelte Webanwendung[1] ermöglicht das Testen der verschiedenen Ähnlichkeitsmaße und Indexstrukturen. Um schnelle Anfragen zu gewährleisten, wurde für die Indexierung der

[1] `http://www.hpi-web.de/naumann/sites/SimSearch2010/`

meisten Datensätze nur eine Teilmenge ausgewählt. Abbildung 1 zeigt eine Beispielanfrage mit Ergebnissen für den Bilder-Datensatz.

In der Anwendung muss zunächst eines der implementierten Ähnlichkeitsmaße ausgewählt werden. Dann werden ein paar zufällig gewählte Elemente angezeigt, aus denen der Nutzer eines als Anfrageobjekt auswählen kann. Alternativ ist es in den meisten Fällen möglich, ein neues Objekt (das nicht notwendigerweise im Datensatz enthalten ist) zu spezifizieren. Nun müssen noch einige Anfrageeinstellungen angegeben werden (Anfrageart und Bereichsgröße oder Anzahl der Nachbarn), bevor die Anfrage ausgeführt werden kann. Die Ergebnisse der Anfrage (die ähnlichsten Elemente aus dem Datensatz) werden in einer Tabelle angezeigt, die nach der Distanz zum Anfrageobjekt sortiert ist. Es werden zudem die Anfragedauer und die Anzahl der benötigten Vergleiche ausgewiesen.

5 Lessons Learned

Zurückblickend haben wir einige Erkenntnisse gewonnen, die wir in diesem Abschnitt beschreiben und in Folgeveranstaltungen mit Programmieranteil beachten werden.

Ähnlichkeitssuche ist ein spannendes und leicht zu motivierendes Thema, das sich mit Masterstudenten gut bearbeiten lässt. Unser Seminar erforderte einen hohen Abstimmungsaufwand zwischen den Gruppen. Die gemeinsame Codebasis hatte verschiedene Effekte: Durch die gegenseitigen Abhängigkeiten konnten die Studenten untereinander Fehler schnell entdecken und gemeinsam beheben. Zusätzlich haben sich die Gruppen gegenseitig unterstützt, z. B. durch gemeinsam genutzte Bibliotheken und Best Practices (u. a. log4j, Caching, Properties). Auch die Notwendigkeit, sich mit den Implementierungen der anderen Gruppen zu befassen, um fremde Ähnlichkeitsmaße für die Evaluierung zu nutzen, wirkte sich kommunikationsfördend aus.

Notwendig für ein Seminar mit Implementierungsanteil ist neben geeigneten Tools auch Wissen um den angemessenen Einsatz dieser Tools. In der Versionsverwaltung waren die Datensätze, das Framework und sämtliche studentischen Implementierungen für alle leicht zugreifbar gespeichert. Dies erlaubte ein einfaches Interagieren mit dem Code, insbesondere in Bezug auf das Erweitern des Frameworks nach den Bedürfnissen der Studenten. Branching in Subversion und insbesondere das Zurückmergen waren so aufwändig, dass die Gruppen lange Zeit in ihren eigenen Branches arbeiteten und Fortschritte und Fehlerbehebungen den jeweils anderen Gruppen oft nicht zur Verfügung standen.

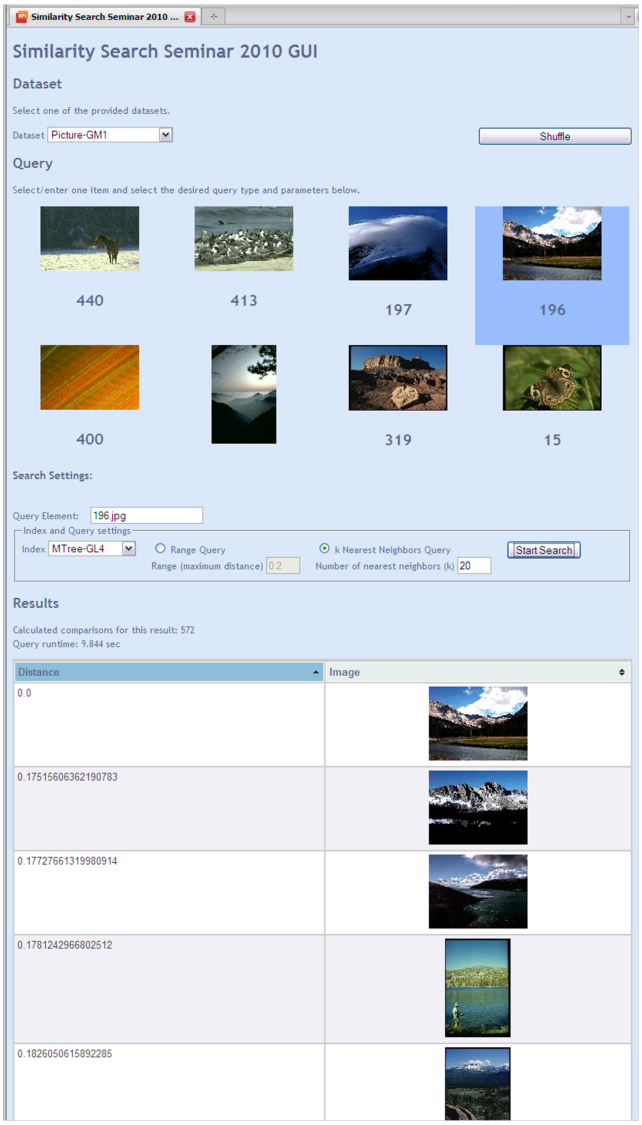

Abb. 1 Screenshot der für das Seminar entwickelten webbasierten Anfrageanwendung. Von oben nach unten sind dargestellt: Datensatzauswahl, Anfrageobjekt-Auswahl, Anfrageeinstellungen und Ergebnisliste

Ausarbeitung, Evaluierungsergebnisse und (grobe) Implementierungsdokumentation wurden in einem gemeinsamen Wiki verfasst. Vorteile des Wikis sind die Möglichkeit, kleinere Artikel statt eines monolithischen Textes zu verfassen, Links auch gruppenübergreifend zwischen den Artikeln zu setzen und die Verfügbarkeit als integrierte Dokumentation für interessierte Leser. Leider wurden die genannten Wiki-Funktionen (Links, Splitten von Artikeln) von den Studenten nur wenig genutzt. Zudem war es schwierig, den Umfang der Dokumentation vorzugeben, da die präzise Angabe von Artikel-/Seitenzahl nicht möglich ist. Problematisch gestaltete sich auch die Bewertung der Ausarbeitungen

durch die Aufspaltung und Formatierung der erstellten Wiki-Artikel.

Projektseminare verursachen einen vielfach höheren Aufwand als klassische Paper-Seminare. Ein gut geplantes Projektseminar ermöglicht Studenten die theoretische *und* praktische Auseinandersetzung mit dem Thema und belohnt die Betreuer durch ein gesteigertes Engagement der Studenten. Die entwickelte Webanwendung bietet als Seminarergebnis eine Demonstration der Leistungsfähigkeit verschiedener Indexierungsalgorithmen. Die Anwendung steht zur Verfügung unter:

`http://www.hpi-web.de/naumann/sites/SimSearch2010/`

Danksagung Wir möchten uns bei allen Studenten bedanken, die erfolgreich und engagiert an unserem Seminar teilgenommen haben.

Literatur

1. E. Bingham and H. Mannila. Random projection in dimensionality reduction: applications to image and text data. In *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250, New York, NY, USA, 2001. ACM.
2. C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. DBpedia - A crystallization point for the Web of Data. *Web Semant.*, 7:154–165, September 2009.
3. T. Bozkaya and M. Ozsoyoglu. Distance-based indexing for high-dimensional metric spaces. In *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, SIGMOD '97, pages 357–368, New York, NY, USA, 1997. ACM.
4. S. Brin. Near Neighbor Search in Large Metric Spaces. In *The VLDB Journal*, pages 574–584, 1995.
5. P. Ciaccia, M. Patella, and P. Zezula. M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, VLDB '97, pages 426–435, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.
6. W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A Comparison of String Distance Metrics for Name-Matching Tasks. In *Proceedings of IJCAI-03 Workshop on Information Integration*, pages 73–78, 2003.
7. T. Curran, G. Keller, and A. Ladd. *SAP R/3 Business Blueprint: Understanding the Business Process Reference Model.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1998.
8. V. Dohnal, C. Gennaro, P. Savino, and P. Zezula. D-Index: Distance Searching Index for Metric Data Sets. *Multimedia Tools Appl.*, 21(1):9–33, 2003.
9. O. Gotoh. An improved algorithm for matching biological sequences. *Journal of molecular biology*, 162(3):705–708, 1982.
10. P. Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 37:547–579, 1901.
11. C. E. Jacobs, A. Finkelstein, and D. Salesin. Fast multiresolution image querying. In *SIGGRAPH*, pages 277–286, 1995.
12. G. Keller and T. Teufel. *SAP R/3 Process Oriented Implementation.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1998.
13. V. I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707–710, 1966.
14. T. Liu, C. Rosenberg, and H. Rowley. Clustering billions of images with large scale nearest neighbor search. In *Proceedings of the Eighth IEEE Workshop on Applications of Computer Vision.* IEEE Computer Society, 2007.
15. MAGIX AG. freedb.org. `http://www.freedb.org`, January 2011.
16. D. Martin, C. Fowlkes, D. Tal, and J. Malik. A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.
17. M. L. Micó, J. Oncina, and E. Vidal. A new version of the nearest-neighbour approximating and eliminating search algorithm (aesa) with linear preprocessing time and memory requirements. *Pattern Recogn. Lett.*, 15(1):9–17, 1994.
18. A. Monge and C. Elkan. The field matching problem: Algorithms and applications. In *In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 267–270, 1996.
19. J. Munkres. Algorithms for the Assignment and Transportation Problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957.
20. C. Olson. A Probabilistic Formulation for Hausdorff Matching. In *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 150–156, 1998.
21. K. Pearson. Mathematical contributions to the theory of evolution. iii. regression, heredity, and panmixia. In *Philosophical Transactions of the Royal Society of London*, volume 187, page 253, 1896. 0962-8436.
22. L. Philips. The double metaphone search algorithm. *C/C++ Users Journal*, 18:38–43, 2000.
23. W. Phillips, Jr., A. K. Bahn, and M. Miyasaki. Person-matching by electronic methods. *Commun. ACM*, 5:404–407, 1962.
24. H.-J. Postel. Die Kölner Phonetik – Ein Verfahren zur Identifizierung von Personennamen auf der Grundlage der Gestaltanalyse. *IBM-Nachrichten*, 19:925–931, 1969.
25. C. Sadowski and G. Levin. SimHash: Hash-based Similarity Detection. `http://simhash.googlecode.com/svn/trunk/paper/SimHashWithBib.pdf`, December 2007.
26. H. Samet. *Foundations of Multidimensional and Metric Data Structures.* Morgan Kaufmann, 2006.
27. W. E. Winkler. Methods for evaluating and creating data quality. *Information Systems*, 29:531–550, 2003.
28. P. N. Yianilos. Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces. In *SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*, 1993.
29. P. Zezula, G. Amato, V. Dohnal, and M. Batko. *Similarity Search - The Metric Space Approach.* Springer, 2006.