HUMBOLDT-UNIVERSITÄT ZU BERLIN

Improved scalable Near duplicate search in very large sequence archives

Exposé zur Diplomarbeit

Thomas Stoltmann*

11. November 2013

Betreuer: Astrid Rheinländer, Prof. Dr. Ulf Leser

1 Motivation und Zielstellung

Ein zentrales Streben der Molekularbiologie ist es, die Genotypen zu bestimmen, die mit Phänotypen korrelieren [Mar08]. Aus diesem Grund ist die DNA-Sequenzierung seit Jahrzenten ein Forschungsfeld vom besonderen Interesse. Seit der Einführung der *Didesoxynucleotiden Sequenzierung* von Sanger et al. [SNC77] durchlief die DNA-Sequenzierung einen stetigen Wandel.

Die in den letzten Jahren aufkommenden *Next Generation Sequencing* (NGS) Techniken stellen derzeit die Spitze dieser Entwicklung dar. Sie verbindet, dass sie im Vergleich zur Sanger-Sequenzierung deutlich kürzere Reads erzeugen, diese jedoch viel schneller zu einem Bruchteil der Kosten produziert werden.

Aufgrund der stetigen Weiterentwicklung bezüglich der Geschwindigkeit und der Kosten bei der DNA-Sequenzierung gestaltet sich die Weiterverarbeitung der sequenzierten Reads zunehmend immer mehr zu einer Herausforderung. Insbesondere das Alignieren von Reads gegen ein Referenzgenom ist aufwendig und verlangt bei dem weiter steigenden Durchsatz moderner NGS-Geräte immer größere Rechnerkapazitäten.

In der Studienarbeit [Sto13] wurde der HDN-Join Algorithmus zur Erkennung von nah-exakten Duplikaten beschrieben, deren Editabstand kleiner gleich 1 ist [vgl. Sto13]. Dieser Algorithmus wurde innerhalb von Stratosphere für eine effiziente Parallelisierung implementiert und evaluiert. In der präsentierten Version wurde HDN-Join für Short-Read Daten gleicher Länge entworfen. Zwar können Reads unterschiedlicher Länge problemlos mit HDN-Join verarbeitet werden, jedoch werden Längenunterschiede nicht hinreichend bei der Kandidatenerzeugung und Partitionierung beachtet, sodass ggf. unnötige Kandidatenpaare erzeugt werden oder unnötig große Partitionen. Diese Einschränkung soll durch Verbesserungen entfernt werden, sodass auch verschieden lange Reads optimaler verarbeitet werden können. Dazu sollen neue Partitionierungsmethoden gefunden und getestet werden. Des Weiteren sind Optimierungen am Kern von HDN-Join geplant, um eine effiziente Ausnutzung von

*stoltman@informatik.hu-berlin.de

Speicher- und Netzwerkressourcen zu ermöglichen. Dies soll vor allem durch effizientere Datenstrukturen für DNA-Reads erfolgen. Ein weiteres Ziel dieser Diplomarbeit ist der Vergleich von HDN-Join mit einem aktuellen Similarity-Join Algorithmus. Während der Evaluation in der Studienarbeit zeigte sich, dass der zum Vergleich herangezogene Algorithmus All-Pairs-Ed zeigte nicht geeignet für das Anwendungsszenario war. Aus diesem Grund soll nun PPJoin+ zum Vergleich herangezogen werden für den eine MapReduce-Implementierung von Vernica [Ver12] bereits existiert. Zusammenfassend sind folgende Punkte für die Studienarbeit geplant:

1. verbesserte Partitionierung mit Berücksichtigung längenverschiedener Reads
2. neue PACT-Datentypen
3. optimierter Datenfluss
4. Portierung von [Ver12] auf Stratosphere zum Vergleich mit HDN-Join
5. umfangreichere Evaluation sämtlicher Anpassungen und Vergleich mit [Ver12]

2 Herangehensweise

Im folgenden Kapitel werden die geplanten Arbeiten für die Diplomarbeit dargestellt.

2.1 Ist-Zustand

In diesem Abschnitt soll der derzeitige Ist-Zustand der PACT-HDN-Join Implementierung dargestellt werden [vgl. Sto13]. Die derzeitige PACT-Implementierung von HDN-Join erzeugt aus dem minimalen Präfix eines jeden Reads die nicht überlappenden Q-Gramme und verwendete deren Hashwerte zur Partitionierung. Das Vorgehen wird in Abbildung 1 dargestellt. Diese Partitionierung wurde vor allem für Reads der gleichen Länge ausgelegt. Treten zwischen Reads unterschiedliche Längen auf, insbesondere Längenunterschiede größer als τ, werden diese Unterschiede bei der Partitionierung nicht hinreichend beachtet. Folglich werden ggf. Partitionen erzeugt, die größer sind, als sie eigentlich sein müssten, da sie womöglich Strings mit Längenunterschieden größer als τ enthalten. Genau diese Schwachstelle soll in der Diplomarbeit u. a. adressiert werden, da es geplant ist, die neue HDN-Join Version auf Reads unterschiedlicher Längen zu evaluieren.

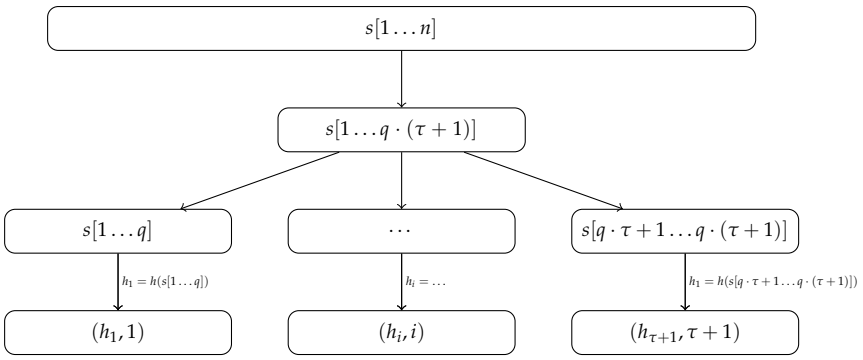

Abbildung 1: Ablauf der Partitionierung mit nicht überlappenden Q-Grammen.

2.2 Verbesserung der Partitionierung

Ein möglicher Weg, die bestehende Partitionierungen um diesen Aspekt zu erweitern, ist eine vorgestellte Längenpartitionierung. Jeder Read der Länge l wird dabei in die Partitionen $l-\tau,\ldots,0,\ldots,l+\tau$ eingeteilt. Da diese Partitionen – abhängig von der Längenverteilung – kleiner als die ursprünglichen Eingabedatenmengen sind und die Größe der von der Q-Gram-Partitionierung erzeugten Partitionen quadratisch mit Readanzahl steigt, kann dies zu kleineren Partitionen führen. Allerdings ist bei diesem Ansatz fraglich, ob nicht die Gesamtdatenmenge nach beiden Partitionen größer als ohne eine vorgeschaltete Längenpartitionierung ist. Die Vorgehensweise wird durch Abbildung 2 erläutert.

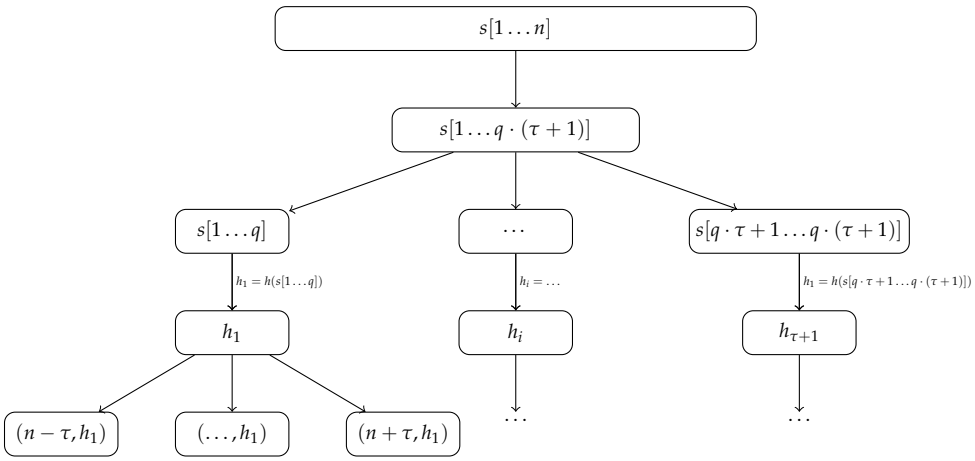

Abbildung 2: Ablauf der Partitionierung mit nicht überlappenden Q-Grammen.

Der zweite Partitionierungsansatz geht auf die Idee zurück, dass bei $\tau = 1$ zwei Zeichenketten nur ähnlich sein können, wenn sie zum großen Teil fast exakt gleich sind, d. h. es ist nur eine Editoperation erlaubt. Durch Ausnutzung dieser Tatsache kann ohne Präfixfilterung ein effizienter Ansatz zur Partitionierung gewonnen werden. Bei ihm werden Zeichenketten in eine gemeinsame Partition eingeordnet, falls sie über eine gemeinsame Teilzeichenkette verfügen. Dazu werden die zu partitionierenden Zeichenketten in $\tau+1$ Teile zerlegt. Damit zwei Zeichenketten innerhalb des Editabstandes τ liegen, muss mindestens ein Teil gleich sein, sofern jeder Teil größer τ ist. Da ein kreuzweises String-Matching sämtlicher Zeichenkettenteile zu aufwendig ist, wird analog zum Rabin-Karp-Algorithmus auf Hashing gesetzt. Dazu werden die Hashwerte der Teilzeichenketten berechnet und zur Einordnung in die Partition verwendet anstelle der Teilzeichenkette selbst. Um die Selektivität weiter zu erhöhen, wird zur Angabe der Partitionszugehörigkeit nicht nur der Hashwert verwendet, sondern auch die relative Position der Zeichenkette, aus der der Hashwert erzeugt wurde. Abbildung 3 stellt diesen Ablauf der Partitionierung schematisch dar. Bei diesem Verfahren ist es vom Vorteil, die nachfolgenden Schritte an diese Partitionierung anzupassen: Die Wahrscheinlichkeit für eine Nichtübereinstimmung zweier positionsgleicher Teilzeichenketten ist bei Ungleichheit ihrer Hashwerte höher als dies bei Gleichheit der Fall wäre. Eine Mismatch bei gleichen Hashwert kann unter dieser Konstellation nur durch eine Kollision zweier Hashwerte erzeugt werden. Folglich sollte HDN-Join zuerst auf den positionsgleichen Teilzeichenketten mit ungleichen Hashwert operieren.

2.3 Verbesserung der der PACT-Datentypen

Ein weiterer Aspekt, der verbessert werden soll, ist die Datenmenge, die innerhalb des Datenflussgraphen fließt. Dies soll einerseits durch eine effizientere Kodierung der Daten erfolgen und andererseits durch Reduzierung der notwendigen Datenmenge.

Dazu sollen neue PACT Datentypen mit reduziertem Alphabet von 8 oder 16 Zeichen realisiert

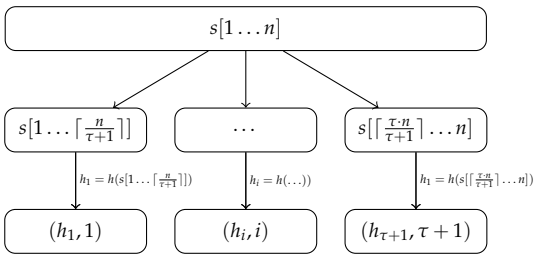

Abbildung 3: Ablauf der Partitionierung mit Teilzeichenketten.

werden, der die vorher verwendeten PactStrings ablösen. Ein solcher Datentyp könnte beim Serialisieren speichereffizient durch eine drei bzw. vier Bit Kodierung repräsentiert werden - zum Vergleich PactStrings würden hier mindestens 8 Bit benötigen (modifiziertes UTF-8). Innerhalb der JVM kann jedes Zeichen eines solchen Datentyps durch ein Byte dargestellt werden. PACTStrings benötigen dafür mindestens 2 Bytes pro Zeichen (UTF-16). Ferner wäre es auch überlegenswert, selbst innerhalb der JVM die serialisierte Repräsentation zu verwenden. Die Benutzung einer solchen gepackten Repräsentation zur Laufzeit kommt jedoch mit einem höheren Rechenaufwand für elementare Operationen einher (z. B. Zugriff auf einzelne Basen, etc.). Bedingt durch die neue Binärrepräsentation der Zeichen müssen jedoch mögliche Auswirkungen auf die Hashfunktionen untersucht wurden. Unter Umständen sind Anpassungen an diesen notwendig.

Weitere Einsparung können erreicht werden, indem nur die benötigten Teile der Zeichenketten weitergeleitet werden. So kann beispielsweise die Q-Gram-Partitionierung lediglich das später genutzte Präfix für HDN-Join weiterleiten oder die Substring-Partitionierung lediglich die genau anderen Teilzeichenketten. Sollten dann jedoch später für die Verifikation die vollständigen Zeichenketten benötigt werden, müssen diese wieder beschafft werden. Auch hier gilt es, die Einsparung durch den zusätzlichen Aufwand für das Nachschlagen der vollständigen Zeichenketten abzuwägen und zu untersuchen.

2.4 Portierung von Vernica [Ver12]

Für die Evaluation von HDN-Join soll im Gegensatz zur Studienarbeit auf einem parallelisierten Similarity-Join Algorithmus zurückgegriffen werden. Zu diesem Zweck bietet sich die PPJoin+ Map-Reduce-Implementierung von [Ver12] an. Dies ist insbesondere interessant, da PPJoin+ bisher nicht mit dem Editabstand evaluiert wurde. Jedoch muss die Implementierung von [Ver12] von Hadoop auf Stratosphere portiert werden, um Einflüsse durch die unterschiedlichen Systeme auf die Messwerte zu verhindern. Da die bestehende Implementierung als Set-Similarity-Join Algorithmus auf Dokumenten und Tokens mit der Jaccard-Simiarlity arbeitet, sind für den Vergleich weitere Anpassungen notwendig. Dazu wird die Set-Similarity-Implementierung in eine Similarity-Join-Implementierung mit Q-Grammen abgewandelt. Ferner wird der BK-Teil (Indexed Kernel) mit PPJoin zur Verwendung des Editabstandes modifiziert [vgl. Xia+11].

2.5 Evaluation

Die geplante Evaluation von HDN-Join wird aus vier Teilen bestehen.

Im ersten und zweiten Teil werden die Verbesserungen von HDN-Join gegen den bisherigen Ist-Zustand jeweils unabhängig voneinander getestet. Hierzu werden im ersten Schritt die neuen Partitionierungsverfahren wie in der Studienarbeit untersucht. Anschließend sollen die Auswirkungen der neuen Partitionierungsverfahren auf das Gesamtsystem betrachtet werden. Dazu sollen Messungen zum Speed-Up und Scale-Up durchgeführt werden. Der zweite Teil wird dies analog für die neuen PACT-Datentypen durchführen.

Nachdem Abschluss der ersten beiden Teile wird anschließend feststehen, welches Partitionierungsverfahren, die besten Resultate liefert und welcher PACT-Datentyp besser ist. Diese „Gewinner" werden anschließend fest in das Gesamtsystem integriert. Im folgenden dritten Teil der Evaluation wird nun die Verbesserung des Speed-Ups und Scale-Ups im Vergleich zu der ursprünglichen Implementierung untersucht.

Im letzten Teil steht ein Vergleich zu einem aktuellen Similarity-Join-Algorithmus an. Dazu wird das auf Stratosphere portierte System von Vernica [Ver12] herangezogen. Untersucht werden soll hier vor allem wieder der Speed-Up und Scale-Up.

3 Verwandte Arbeiten

Im Rahmen dieses Kapitels werden bestehende Arbeiten betrachtet, die die algorithmischen Problemen behandeln, die der Studienarbeit zugrundeliegen.

3.1 Similarity Joins

Similarity-Join-Algorithmen lassen sich in zwei Kategorien einteilen, die exakten und die approximativen. Letztere lösen das Similarity-Join-Problem approximativ und sind damit für den Zweck der Studienarbeit ungeeignet. Alle hier vorgestellten Algorithmen gehören zur ersten Kategorie, sodass nachfolgend mit Similarity-Join immer ein exakter Similarity-Join bezeichnet wird. Erste Arbeiten zu diesem Thema stammen unter anderem von [Gra+01] und [CGK06], in denen grundlegende Techniken wie die Präfixfiltrierung entwickelt wurden.

Fast alle Similarity-Join-Algorithmen lassen sich in eine von drei Gruppen einordnen. Eine dieser Gruppen bilden die filter-and-refine-Algorithmen. Sie erzeugen Kandidatenpaare während der filter-Phase mithilfe von Signaturen, die anschließend in der refine-Phase verifiziert werden. Zu ihnen gehören unter anderem Part-Enum [vgl. AGK06], All-Pairs [vgl. BMS07] und ED-Join [XWL08]. Die nächste Gruppe besteht aus Algorithmen, deren Kernideen auf einer Partitionierung der zu verarbeitenden Strings basieren. Zu ihnen zählen Pass-Join [vgl. Li+11] und PartSS [vgl. LSZ11]. Eine weitere Gruppe wird von Algorithmen gebildet, die sich einer Baumdatenstruktur bedienen, wie PeARL [vgl. RL11] oder Trie-Join [vgl. WLF10; FWL12]. Sie sind tendenziell eher für kurze Strings geeignet [vgl. WLF12].

Eine ausführlichere Darstellung bisheriger Similarity-Join-Algorithmen sowie ihrer Evaluationen wird im Rahmen des *Related Work* Kapitels in der Diplomarbeit stattfinden.

3.2 MapReduce-basierte Similarity-Joins

Neben den klassischen Similarity-Join-Algorithmen existieren auch mit MapReduce parallelisierte Algorithmen, die in diesem Abschnitt beschrieben werden.

Der Set-Similarity-Join-Algorithmus von Vernica et al. [VCL10] bzw. Vernica [Ver12] baut auf einem dreistufigen Vorgehen auf und wendet zur Parallelisierung das MapReduce-Paradigma an. Für jede Stufe gibt es genau zwei unterschiedliche Implementierungen, die jeweils aus einer oder zwei MapReduce-Programmen bestehen. Als Eingabe erhält der Algorithmus eine oder zwei Listen von Dokumenten. Ein Dokument stellt dabei eine Liste von Token dar.

Im ersten Schritt findet eine Sortierung der Token vom Join-Attribut des Records nach ihrer Häufigkeit statt. Die sortierte Tokenliste spielt eine wichtige Rolle für die nachfolgenden Schritte. Die erste Implementierung, das *Basic Token Ordering* (BTO), benötigt zwei Map/Reduce-Programme. Wohingegen die zweite Implementierung, das *One Phase to Order Tokens* (OPTO), nur ein MapReduce-Programm benötigt. Sie muss jedoch dafür die Tokenliste komplett im Speicher halten.

Im zweiten Schritt werden die Paare aus Record-Ids (RIDs) bestimmt, die die gewählte Ähnlichkeitsschwelle überschreiten. Die Ergebnisse werden in Form von Tupeln aus RID des einen Dokumentes sowie des anderen Dokumentes und ihrer Ähnlichkeit zurückgegeben. Beide Implementierungen benötigen nur ein MapReduce-Programm. Das als *Basic Kernel* (BK) bezeichnete Verfahren verwendet

einen einfachen und selbstentwickelten Similarity-Join-Algorithmus. Das *Indexed Kernel* (PK) Verfahren benutzt für die gleiche Aufgabe den PPJoin+ Algorithmus von Xiao et al. [Xia+11].

Während des dritten Schrittes wird mithilfe der Paarliste aus dem zweiten Schritt, die als ähnlich angesehen werden, das finale Ergebnis aus den Ursprungsdaten extrahiert. Das als *Basic Record Join* (BRJ) bezeichnete Verfahren verwendet zwei MapReduce-Programme, um das Ergebnis zu berechnen. Hingegen erledigt das *One-Phase Record Join* (OPRJ) die gleiche Arbeit in einem Programm, indem es die insgesammt vier Funktionen zu zwei zusammenfasst.

In der Evaluation ergab sich, dass der beste Scale-Up für den Self-Join und den RxS-Join durch die BTO-PK-BRJ-Kombination erreicht wird. Der Speed-Up war bei allen getesteten Kombinationen gleich [vgl. Ver12, S. 49ff].

In [MF12] wird ein Framework für Similarity-Joins mit MapReduce beschrieben. Es verwendet eine zweiphasige Herangehensweise, die für Sets und Multisets geeignet ist, sich jedoch ggf. auf Vektoren anwenden lässt. Die Ähnlichkeitsmaße müssen die *Shuffling Invariant Property* erfüllen [vgl. CS02], wie es bei *Nominal Similarity-Messures* (z. B. Jaccard-Similarity) der Fall ist [vgl. MF12]. Aufgrund der fehlenden Möglichkeit, den Editabstand zu verwenden, ist das Framework für das angestrebte Anwendungsszenario nicht geeignet.

Weitere MapReduce Umsetzungen von Similiarity-Join-Algorithmen stammen von Elsayed et al. [ELO08] und Baraglia et al. [BML10].

4 Zeitplan

Die nachfolgende Tabelle zeigt alle anfallenden Arbeitspakete zur Durchführung der Diplomarbeit. Anzumerken ist, dass sich die einzelnen Arbeitspakete teilweise überlappen, sodass eine Gesamtdauer von 6 Monaten erreicht werden kann. Jedoch birgt insbesondere die Portierung der MapReduce-Implemetierung von [Ver12] Unwägbarkeiten, die vorher nicht vollständig ausgeräumt werden können.

Arbeitspaket	***Zeitrahmen***
Implementierung gesamt	2 Monate
- Implementierung der neuen Partitionierungsverfahren	1,5 Wochen
- Implementierung	1-2 Wochen
- Testen (Korrektheit)	0,5 Woche
- Implementierung der neuen PACT-Datentypen	1,5 Wochen
- Implementierung	1 Woche
- Testen (Korrektheit und Effizienz)	0,5 Woche
- Portierung der MapReduce PPJoin Implementierung von [Ver12]	1 Monat
- Portieren von Hadoop auf Stratosphere	2-4 Wochen
- Implementierung und Anpassungen für die Editabstand	1 Wochen
- Testen (Korrektheit)	1 Wochen
Evaluation	1,5 Monate
- Evaluation der Partitionierungsverfahren	1 Woche
- Evaluation der PACT-Datentypen	1 Woche
- Evaluation des Gesamtsystems (alt vs neu)	2 Woche
- Evaluation gegen Vernica	2 Wochen
Anfertigen der schriftlichen Ausarbeitung	2 Monate

Tabelle 1: Darstellung der Arbeitspakete und deren benötigter Zeitaufwand.

Literatur

Literaturverzeichnis

[Afr+12] Foto N Afrati, Anish Das Sarma, David Menestrina, Aditya Parameswaran und Jeffrey D Ullman. "Fuzzy joins using MapReduce". In: *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*. IEEE. 2012, S. 498–509.

[AGK06] Arvind Arasu, Venkatesh Ganti und Raghav Kaushik. "Efficient exact set-similarity joins". In: *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, September 12-15, 2006*. VLDB Endowment. 2006, S. 918–929.

[Ale+10] Alexander Alexandrov, Max Heimel, Volker Markl, Dominic Battré, Fabian Hueske, Erik Nijkamp, Stephan Ewen, Odej Kao und Daniel Warneke. "Massively parallel data analysis with PACTs on nephele". In: *Proceedings of the VLDB Endowment* 3.1-2 (2010), S. 1625–1628.

[ATY13] Maha Ahmed Alabduljalil, Xun Tang und Tao Yang. "Optimizing parallel algorithms for all pairs similarity search". In: *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM. 2013, S. 203–212.

[Bal+13] Susanne Balzer, Ketil Malde, Markus A Grohme und Inge Jonassen. "Filtering duplicate reads from 454 pyrosequencing data". In: *Bioinformatics* 7 (29 2013), S. 830–836.

[Bat+10] D. Battré, S. Ewen, F. Hueske, O. Kao, V. Markl und D. Warneke. "Nephele/PACTs: a programming model and execution framework for web-scale analytical processing". In: *Proceedings of the 1st ACM symposium on Cloud computing*. ACM. 2010, S. 119–130.

[Bla+10] S. Blanas, J.M. Patel, V. Ercegovac, J. Rao, E.J. Shekita und Y. Tian. "A comparison of join algorithms for log processing in mapreduce". In: *Proceedings of the 2010 international conference on Management of data*. ACM. 2010, S. 975–986.

[BML10] Ranieri Baraglia, Gianmarco De Francisci Morales und Claudio Lucchese. "Document Similarity Self-Join with MapReduce". In: *ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14-17 December 2010*. 2010, S. 731–736.

[BMS07] Roberto J. Bayardo, Yiming Ma und Ramakrishnan Srikant. "Scaling up all pairs similarity search". In: *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*. 2007, S. 131–140.

[BN02] Ricardo Baeza-Yates und Gonzalo Navarro. "New and faster filters for multiple approximate string matching". In: *Random Structures & Algorithms* 20.1 (2002), S. 23–49.

[BN97] Ricardo Baeza-Yates und Gonzalo Navarro. "Multiple approximate string matching". In: *Algorithms and Data Structures*. 1997, S. 174–184.

[Boc+07] Thomas Bocek, Ela Hunt, Burkhard Stiller und Fabio Hecht. *Fast similarity search in large dictionaries*. Techn. Ber. Department of Informatics, 2007. URL: `http://fastss.csg.uzh.ch/ifi-2007.02.pdf`.

[Boy11] Leonid Boytsov. "Indexing methods for approximate dictionary searching: Comparative analysis". In: *Journal of Experimental Algorithmics (JEA)* 16 (2011), S. 1–1.

[C+03] William W Cohen, Pradeep D Ravikumar, Stephen E Fienberg et al. "A Comparison of String Distance Metrics for Name-Matching Tasks." In: *IIWeb*. Bd. 2003. 2003, S. 73–78.

[CGK06] Surajit Chaudhuri, Venkatesh Ganti und Raghav Kaushik. "A primitive operator for similarity joins in data cleaning". In: *Proceedings of the 22nd International Conference on Data Engineering, ICDE 2006, 3-8 April 2006, Atlanta, GA, USA*. 2006, S. 5–5.

[Cha10] Jairam Chandar. "Join Algorithms using Map/Reduce". Magisterarb. University of Edinburgh, 2010. URL: `http://www.inf.ed.ac.uk/publications/thesis/online/IM100859.pdf`.

[CS02] S.H. Cha und S.N. Srihari. "On measuring the distance between histograms". In: *Pattern Recognition* 35.6 (2002), S. 1355–1370.

[DG08] J. Dean und S. Ghemawat. "MapReduce: Simplified data processing on large clusters". In: *Communications of the ACM* 51.1 (2008), S. 107–113.

[DG92] David DeWitt und Jim Gray. "Parallel database systems: the future of high performance database systems". In: *Communications of the ACM* 35.6 (1992), S. 85–98.

[ELO08] Tamer Elsayed, Jimmy Lin und Douglas W. Oard. "Pairwise document similarity in large collections with MapReduce". In: *ACL 2008, Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, June 15-20, 2008, Columbus, Ohio, USA, Short Papers*. The Association for Computer Linguistics. 2008, S. 265–268.

[FN04] Kimmo Fredriksson und Gonzalo Navarro. "Average-optimal single and multiple approximate string matching". In: *Journal of Experimental Algorithmics (JEA)* 9 (2004), S. 1–4.

[FWL12] Jianhua Feng, Jiannan Wang und Guoliang Li. "Trie-join: a trie-based method for efficient string similarity joins". In: *VLDB J.* 21.4 (2012), S. 437–461.

[Gra+01] L. Gravano, P.G. Ipeirotis, H.V. Jagadish, N. Koudas, S. Muthukrishnan und D. Srivastava. "Approximate string joins in a database (almost) for free". In: *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, September 11-14, 2001, Roma, Italy*. 2001, S. 491–500.

[GTS09] Vicente Gomez-Alvarez, Tracy K Teal und Thomas M Schmidt. "Systematic artifacts in metagenomes from complex microbial communities". In: *The ISME journal* 3.11 (2009), S. 1314–1317.

[HFN05] Heikki Hyyrö, Kimmo Fredriksson und Gonzalo Navarro. "Increased bit-parallelism for approximate and multiple string matching". In: *Journal of Experimental Algorithmics (JEA)* 10 (2005), S. 2–6.

[HS11] Marios Hadjieleftheriou und Divesh Srivastava. *Approximate String Processing*. Bd. 2. 4. Now Publishers, 2011, S. 267–402. DOI: 10.1561/1900000010.

[Hue+12] Fabian Hueske, Mathias Peters, Matthias J Sax, Astrid Rheinländer, Rico Bergmann, Aljoscha Krettek und Kostas Tzoumas. "Opening the black boxes in data flow optimization". In: *Proceedings of the VLDB Endowment* 5.11 (2012), S. 1256–1267.

[Hyy03] Heikki Hyyrö. "Practical methods for approximate string matching". Diss. 2003.

[Jia+13] Yu Jiang, Dong Deng, Jiannan Wang, Guoliang Li und Jianhua Feng. "Efficient parallel partition-based algorithms for similarity search and join with edit distance constraints". In: *Proceedings of the Joint EDBT/ICDT 2013 Workshops*. ACM. 2013, S. 341–348.

[Lev66] Vladimir I. Levenshtein. "Binary codes capable of correcting deletions, insertions, and reversals". In: *Soviet physics doklady*. Bd. 10. 8. 1966, S. 707–710.

[Li+11] Guoliang Li, Dong Deng, Jiannan Wang und Jianhua Feng. "Pass-join: A partition-based method for similarity joins". In: *Proceedings of the VLDB Endowment* 5.3 (2011), S. 253–264.

[LSZ11] Zhixu Li, Laurianne Sitbon und Xiaofang Zhou. "PartSS: An Efficient Partition-based Filtering for Edit Distance Constraints". In: *Australasian Database Conference (ADC 2011)*. Bd. 115. 2011, S. 103–112. URL: http://crpit.com/confpapers/CRPITV115Li.pdf.

[Mar08] E.R. Mardis. "The impact of next-generation sequencing technology on genetics". In: *Trends in genetics* 24.3 (2008), S. 133–141.

[MF12] Ahmed Metwally und Christos Faloutsos. "V-SMART-join: a scalable mapreduce framework for all-pair similarity joins of multisets and vectors". In: *Proceedings of the VLDB Endowment* 5.8 (2012), S. 704–715.

[MF82] Moshe Mor und Aviezri S. Fraenkel. "A hash code method for detecting and correcting spelling errors". In: *Communications of the ACM* 25.12 (1982), S. 935–938.

[Mis+13] Shashwat Mishra, Tejas Gandhi, Akhil Arora und Arnab Bhattacharya. "Efficient edit distance based string similarity search using deletion neighborhoods". In: *Proceedings of the Joint EDBT/ICDT 2013 Workshops*. ACM. 2013, S. 375–383.

[MM96] R. Muth und U. Manber. "Approximate multiple string search". In: *Combinatorial Pattern Matching*. Springer. 1996, S. 75–86.

[Mye86] E.W. Myers. "An O (ND) difference algorithm and its variations". In: *Algorithmica* 1.1 (1986), S. 251–266.

[Mye94] E.W. Myers. "A sublinear algorithm for approximate keyword searching". In: *Algorithmica* 12.4 (1994), S. 345–374.

[Mye99] Gene Myers. "A fast bit-vector algorithm for approximate string matching based on dynamic programming". In: *Journal of the ACM (JACM)* 46.3 (1999), S. 395–415.

[Nav01] G. Navarro. "A guided tour to approximate string matching". In: *ACM computing surveys (CSUR)* 33.1 (2001), S. 31–88.

[Nav97] Gonzalo Navarro. "Multiple approximate string matching by counting". In: (1997).

[RL11] A. Rheinländer und U. Leser. "Scalable Sequence Similarity Search and Join in Main Memory on Multi-Cores". In: *2nd International Workshop on High Performance Bioinformatics and Biomedicine (HiBB), Bordeaux, France* (2011).

[RUN98] M. Ronaghi, M. Uhlén und P. Nyrén. "A sequencing method based on real-time pyrophosphate". In: *Science* 281.5375 (1998), S. 363–365.

[Sel74] P.H. Sellers. "On the theory and computation of evolutionary distances". In: *SIAM Journal on Applied Mathematics* (1974), S. 787–793.

[Sel80] P.H. Sellers. "The theory and computation of evolutionary distances: pattern recognition". In: *J. algorithms* 1.4 (1980), S. 359–373.

[SK04] Sunita Sarawagi und Alok Kirpal. "Efficient set joins on similarity predicates". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004*. 2004, S. 743–754.

[SNC77] F. Sanger, S. Nicklen und A.R. Coulson. "DNA sequencing with chain-terminating inhibitors". In: *Proceedings of the National Academy of Sciences* 74.12 (1977), S. 5463.

[ST95] Erkki Sutinen und Jorma Tarhio. "On using q-gram locations in approximate string matching". In: *Algorithms—ESA'95* (1995), S. 327–340.

[ST96] Erkki Sutinen und Jorma Tarhio. "Filtration with q-samples in approximate string matching". In: *Combinatorial Pattern Matching, 7th Annual Symposium, CPM 96, Laguna Beach, California, USA, June 10-12, 1996, Proceedings*. Bd. 1075. Springer. 1996, S. 50–63.

[Sto13] Thomas Stoltmann. *Towards scalable near duplicate search in very large sequence archives. A deletion neighbood and hashing based similarity join algorithm.* Techn. Ber. Humboldt-Universität zu Berlin, 2013.

[Tim+09] Bernd Timmermann, Marcus W. Albrecht, Vyacheslav Amstislavskiy, Tatiana Borodina, Andreas Dahl, Ralf Herwig, Wilfried Nietfeld, Dmitri Parkhomchuk, Alexej Soldatov, Ralf Sudbrak und Hans Lehrach. "Systematische Analyse der genetischen Variabilität des Menschen". In: *Laborwelt* 3 (2009), S. 8–10.

[Ukk85a] Esko Ukkonen. "Algorithms for approximate string matching". In: *Information and control* 64.1-3 (1985), S. 100–118.

[Ukk85b] Esko Ukkonen. "Finding approximate patterns in strings". In: *Journal of algorithms* 6.1 (1985), S. 132–137.

[Ukk93] Esko Ukkonen. "Approximate string-matching over suffix trees". In: *Combinatorial Pattern Matching*. Springer. 1993, S. 228–242.

[VCL10] Rares Vernica, Michael J. Carey und Chen Li. "Efficient parallel set-similarity joins using MapReduce". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June 6-10, 2010*. ACM. 2010, S. 495–506.

[Ver12] Rares Vernica. "Efficient Processing of Set-Similarity Joins on Large Clusters". Diss. UNIVERSITY OF CALIFORNIA, IRVINE, 2012.

[VL09] Rares Vernica und Chen Li. "Efficient top-k algorithms for fuzzy search in string collections". In: *Proceedings of the First International Workshop on Keyword Search on Structured Data*. ACM. 2009, S. 9–14.

[Wan+09] Wei Wang, Chuan Xiao, Xuemin Lin und Chengqi Zhang. "Efficient approximate entity extraction with edit distance constraints". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2009, Providence, Rhode Island, USA, June 29 - July 2, 2009*. 2009, S. 759–770.

[WF74] Robert A Wagner und Michael J Fischer. "The string-to-string correction problem". In: *Journal of the ACM (JACM)* 21.1 (1974), S. 168–173.

[WK09] Daniel Warneke und Odej Kao. "Nephele: efficient parallel data processing in the cloud". In: *Proceedings of the 2nd workshop on many-task computing on grids and supercomputers*. ACM. 2009, S. 8.

[WLF10] Jiannan Wang, Guoliang Li und Jianhua Feng. "Trie-join: Efficient trie-based string similarity joins with edit-distance constraints". In: *Proceedings of the VLDB Endowment* 3.1-2 (2010), S. 1219–1230.

[WLF12] Jiannan Wang, Guoliang Li und Jianhua Feng. "Can we beat the prefix filtering?: an adaptive framework for similarity join and search". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2012, Scottsdale, AZ, USA, May 20-24, 2012*. 2012, S. 85–96.

[WM92] Su Wu und Udi Manber. "Fast text searching: allowing errors". In: *Communications of the ACM* 35.10 (1992), S. 83–91.

[Xia+11] Chuan Xiao, Wei Wang, Xuemin Lin, Jeffrey Xu Yu und Guoren Wang. "Efficient similarity joins for near-duplicate detection". In: *ACM Transactions on Database Systems (TODS)* 36.3 (2011), S. 15.

[XWL08] Chuan Xiao, Wei Wang und Xuemin Lin. "Ed-Join: an efficient algorithm for similarity joins with edit distance constraints". In: *Proceedings of the VLDB Endowment* 1.1 (2008), S. 933–944.