Expose für die Bachelorarbeit: Optimierung der Laufzeit von PREON

Betreuer: Prof. Dr. Ulf Leser, Arik Ermshaus

Insititution: Humbold-Universität zu Berlin

Henry Beiker

April 2022

1 Einleitung

Die Tumorforschung ist seit Beginn des Humangenomprojekts 1990 weit gekommen. Ärzten heutzutage ist es möglich, viele Tumor-Typen mit bewährten Therapien zu behandeln [9]. Jedoch helfen diese aktuellen, auf großen Studien basierenden Therapien gerade bei Patienten mit seltenen Tumoren oftmals nicht. Hierfür werden Techniken der Präzisionsonkologie genutzt, bei denen Therapien aufgrund der genetischen Eigenschaften eines Patienten vorgeschlagen werden [5]. Diese Therapien werden von Tumorboards erstellt. Ein molekulares Tumorboard besteht aus einer Vielzahl von Experten, welche sich unter Hinzunahme von verschiedenen Informationsquellen zur Interaktion zwischen genetischen Eigenschaften, dem Tumor-Typ und den Medikamenten beraten [1]. Um diese Informationen bereitstellen zu können, werden mehrere Datenbanken integriert, um möglichst viele Erkenntnisse für einen Patienten zur Hand zu haben. Bei einer solchen Datenintegration ist es wichtig, die Daten zu normalisieren, sodass heterogene Repräsentationen korrekt aufeinander abgebildet werden [1]. Da die zur Verfügung stehenden Datenbanken in der Regel sehr viele Einträge besitzen, werden schnelle und akkurate String-Matching-Algorithmen benötigt, um alle Einträge aus mehreren Quellen akkurat und schnell zu normalisieren und zu integrieren.
Für diesen Zweck wurde PREON [1] entwickelt. PREON nutzt eine Folge von Matching-Algorithmen, welche eine steigende Komplexität besitzen. Da insbesondere der letzte Schritt von PREON sehr rechenintensiv ist, ist es das Ziel dieser Arbeit, PREON in diesem Schritt zu beschleunigen und eine Laufzeitverbesserung zu erreichen. Dies werden wir dadurch versuchen zu erreichen, indem wir ein Indizierungsverfahren verwenden, welches die Berechnung der Edit-Distance - im letzten Schritt von PREON - beschleunigt. Zusätzlich werden wir eine Reihe von Filtermethoden implementieren, welche bei dem aktuell implementierten

Scanning möglichst viele Kandidaten aussortieren, damit für diese die teuren Operationen gar nicht erst durchgeführt werden müssen. Außerdem werden wir prüfen, ob sich diese Filter auch während dem Indizierungsverfahren einsetzen lassen, um dieses weiter zu beschleunigen.

2 Grundlagen und verwandte Arbeiten

In diesem Abschnitt werden wir auf die Grundlagen eingehen. Dazu gehört eine Problembeschreibung der String-Ähnlichkeitssuche, eine Definition der Edit-Distanz sowie Definitionen der von uns verfolgten Ansätze, um eine Beschleunigung der Ähnlichkeitssuche zu bewirken. Anschließend werden wir auf PREON eingehen, welches das Problem mit einer naiven Implementierung löst.

2.1 Grundlagen

2.1.1 String-Ähnlichkeitssuche

Definition 1 *Von einer String-Ähnlichkeitssuche sprechen wir genau dann, wenn wir aus einer Menge von Strings S und einem gegebenen Eingabestring q mithilfe einer Ähnlichkeitsfunktion sim sowie einer Schwelle t alle Strings aus S finden, sodass das Ergebnis der Ähnlichkeitsfunktion von q und einem String aus S kleiner ist als die Schwelle t [7].*

2.1.2 Edit-Distanz

Die Ähnlichkeitsfunktion *sim* aus Definition 1 ist bei uns die Edit-Distanz.

Definition 2 *Die Edit-Distanz von zwei Strings $d(s_1, s_2)$ entspricht der minimal benötigten Anzahl von eingefügten, gelöschten oder ersetzten Zeichen, um s_1 in s_2 zu transformieren. Zwei Strings sind unterhalb einer Edit-Distanz-Schwelle t, wenn $d(s_1, s_2) \leq t$ gilt [7].*

2.1.3 Trie-Index

Eine fundamentale Änderung, welche wir erproben wollen, ist die Nutzung eines Indexing Verfahrens namens minIL [8]. Dieses Verfahren erlaubt es uns, mithilfe von Sketches welche zufällig aus einem String gebildet werden, einen Suchbaum zu erstellen. Dadurch können wir Kandidaten welche zu unähnlich zu einem Sketch sind bereits vor der eigentlichen Edit-Distanz Berechnung ausschließen. Da mehrere Kandidaten denselben Sketch haben können, können wir so effizient viele Kandidaten auf einmal verwerfen.

2.1.4 Partitions-Filter

Der Partitionsfilter ermöglicht es uns, mögliche Kandidaten zu verwerfen, ohne die Edit-Distanz zu berechnen. Dazu partitionieren wir einen Kandidaten-String in $t + 1$ Partitionen. Naiv legen wir fest, dass alle Partitionen möglichst gleich

groß sind. Wenn nun keine der $t+1$ Partiotionen einem Teilstring des Eingabestrings q entspricht, kann der Kandidat verworfen werden, da die Schwelle von t bei der Edit-Distanz aufgrund des Schubladenprinzips übertroffen wird [3].

2.1.5 Längen-Filter

Eine weitere simple Methode ist es, Strings auszuschließen, welche substanziell kürzer oder länger sind als der Eingabestring. Substanziell heißt hierbei, dass ein String um mindestens t Zeichen länger oder kürzer ist, da somit die Schwelle t ohnehin nicht eingehalten werden kann [4].

2.1.6 Häufigkeits-Filter

Beim Häufigkeits-Filter zählen wir die Anzahl der Vorkommen eines Buchstaben und speichern diese in einem Vektor. Für zwei Häufigkeitsvektoren zählen wir nun jeweils die Anzahl aller $+1$ und -1 Operationen, um den einen Vektor in den anderen zu transformieren. Hierbei erhalten wir eine untere Schranke für die Edit-Distanz [4].

2.1.7 Datenspezifischer-n-Gramm Filter

Die generelle Idee von dem n-Gramm Filter ist es, n-Gramme beziehungsweise Substrings zu suchen, welche auffällig oft in einem Datensatz vorkommen. Anschließend können wir überprüfen, ob wenn ein Eingabestring q eines dieser n-Gramme enthält, dieser auch in einem Kandidaten vorzufinden ist. Ist dies nicht der Fall, können wir den Kandidaten aussortieren.

2.2 Verwandte Arbeiten

2.2.1 PREON

PREON ist ein String-Matching-Algorithmus implementiert als Python-Bibliothek, deren Funktion es ist, Tumor- und Medikamentennamen zu normalisieren. Dabei werden Medikamentennamen auf eine Chembl-ID gemapped, welche eine ID in der Chembl-Datenbank enthält und wiederum eine biochemische Entität meist im Bereich der Medikamente beschreibt. Tumornamen werden auf DOID's gemapped. Diese sind einer Krankheit zugeordnete einzigartige ID. PREON wurde ursprünglich als Alternative zu MetaKB entwickelt, welches als Web-Service angeboten wird, jedoch aufgrund dessen in größeren Integrationsprojekten nicht verwendet werden kann.

Im Rahmen der Arbeit an PREON [1] und dessen Evaluation wurden in Zusammenarbeit mit Klinikern und Forschern der Charité Berlin verschiedene Goldstandards sowohl für Tumornamen als auch für Medikamentennamen erstellt. PREON hat zur Evaluation eine Precision-Recall-Analyse durchgeführt. Bei dieser kam heraus, dass das partielle Matching sehr genau ist und einen F1-Wert von 95% auf den getesteten Daten erreicht. Da dies jedoch der vom Rechenaufwand her teuerste Schritt ist, wollen wir diesen beschleunigen, ohne

die Genauigkeit negativ zu beeinflussen. PREON's Workflow besteht aus mehreren Schritten. Nach einem Preprocessing Schritt wird geguckt, es exakte oder Teil-Matchings gibt.Schlagen diese alle fehl, wird im letzten Schritt geschaut, ob mithilfe der Edit-Distanz und einem gegebenen Schwellenwert zwei Strings zueinander passen. Dazu wird die reduzierte Eingabe mit jedem Eintrag aus dem Referenzwörterbuch verglichen.

3 Methoden

PREON verwendet für die Berechnung der Edit-Distanz oder auch Levenshtein-Distanz eine einfache Implementierung aus der Python-Bibliothek Jellyfish [6]. Da die Berechnung der Levenshtein-Distanz sehr rechenaufwendig ist, wollen wir versuchen, dies zu beschleunigen. Wichtig dabei ist es, die Genauigkeit von PREON beizubehalten und gleichzeitig die gewünschte Laufzeitreduktion zu erzielen.Da die Berechnung der Levenshtein-Distanz von der Laufzeit her sehr rechenaufwendig ist, wollen wir mit dieser Arbeit versuchen, dies zu beschleunigen. Um das zu erreichen, stellen wir in diesem Abschnitt Methoden vor, welche wir verwenden möchten.

3.1 Filter-Ansatz

PREON [1] vergleicht momentan einen Eingabestring mit allen Referenzen aus einem Referenzwörterbuch. Dadurch wird sehr oft für offensichtlich schlechte Kandidaten die Levensthein-Distanz berechnet. Ziel des Filter-Ansatzes ist es, möglichst viele dieser schlechten Kandidaten mit einer geringeren Laufzeit zu verwerfen. Dazu verwenden wir die in 2.1 vorgestellten Filtermethoden. Diese sollen einzeln oder in verschiedenen Kombinationen mit in den Workflow von PREON eingebunden werden. Erhalten wir nun also einen Eingabestring, müssen die Kandidaten zuerst alle Filter passieren, bevor die Levensthein-Distanz berechnet wird. Wir erhoffen uns davon, dass die Summe der Gesamtlaufzeit zur Berechnung der Filter die Summe der Laufzeit aller Levensthein-Distanz-Berechnungen unterschreitet und wir dadurch die Laufzeit optimieren.

3.2 Trie-Index Verfahren (minIL)

Unabhängig vom Filter Ansatz wollen wir auch minIL [8] erproben. Hierbei erhoffen wir uns, dass viele Medikamenten- und Tumornamen ähnliche Sketches erzeugen. Dadurch könnten wir viele Kandidaten für eine gegebene Query auf einmal verwerfen. Während dem durchlaufen des Suchbaumes können wir außerdem Längen- und Positionsfilter verwenden, um die Laufzeit weiter zu minimieren.

4 Evaluation

Da diese Arbeit zwei Ansätze implementiert, welche beide auf der bestehenden Version von PREON aufbauen, werden wir die aktuelle Version, den Filteransatz und das Trie-Index Verfahren [8] miteinander vergleichen. Dafür steht uns ein Datensatz zur Verfügung, welcher im PREDICT-Projekt [2] erstellt worden ist und circa 100.000 Medikamentennamen sowie 90.000 Tumornamen enthält. Bei der Evaluation des Filteransatzes werden wir zuerst jeden Filter alleine evaluieren und anschließend alle sinnvollen Kombinationen von Filtern. Bei allen Experimenten werden die minimale, durchschnittliche, maximale sowie die Gesamtlaufzeit analysiert und diskutiert. Abschließend werden wir konkrete Anwendungsfälle diskutieren, in welchen bestimmte Methoden besser oder schlechter abschneiden.

References

[1] Arik Ermshaus, Michael Piechotta, Gina Rüterand Ulrich Keilholz, Ulf Leser, and Manuela Benary. preon: Fast and accurate entity normalization for drug names and cancer types in precision oncology. unpublished, 2022.

[2] Prof. Dr. Ulf Leser, Prof. Dr. Nils Blüthgen, Prof. Dr. med. Ulrich Keilholz, Prof. Dr. Christine Sers, Prof. Dr. Reinhold Schäfe, and Prof. Dr. Marius Kloft. Predict. `https://pathologie-ccm.charite.de/forschung/arbeitsgruppen/ag_sers_molekulare_tumorpathologie/predict/`, 2019.

[3] Guoliang Li, Dong Deng, Jiannan Wang, and Jianhua Feng. Pass-join: A partition-based method for similarity joins, 2011.

[4] Astrid Rheinländer, Martin Knobloch, Nicky Hochmuth, and Ulf Leser. Prefix tree indexing for similarity search and similarity joins on genomic data. In Michael Gertz and Bertram Ludäscher, editors, *Scientific and Statistical Database Management*, pages 519–536, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[5] Johannes Starlinger, Steffen Pallarz, Jurica Ševa, Damian Rieke, Christine Sers, Ulrich Keilholz, and Ulf Leser. Variant information systems for precision oncology. *BMC Medical Informatics and Decision Making*, 18(1):107, Nov 2018.

[6] James Turk. https://pypi.org/project/jellyfish/, 27.02.2022.

[7] Minghe Yu, Guoliang Li, Dong Deng, and Jianhua Feng. String similarity search and join: a survey. *Frontiers of Computer Science*, 10(3):399–417, Jun 2016.

[8] Haoyu Zhang and Qin Zhang. Minsearch: An efficient algorithm for similarity search under edit distance. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*, KDD '20, page 566–576, New York, NY, USA, 2020. Association for Computing Machinery.

[9] Jon Zugazagoitia, Cristiano Guedes, Santiago Ponce, Irene Ferrer, Sonia Molina-Pinelo, and Luis Paz-Ares. Current challenges in cancer treatment. *Clinical Therapeutics*, 38(7):1551–1566, 2016.