Anwendbarkeit von Methoden der Sequenzausrichtung in der Bioinformatik auf Quelltext-Ähnlichkeitsanalyse

Jonas Scholl

Institut für Informationssicherheit und Verlässlichkeit (KASTEL)
Betreuender Mitarbeiter: Robin Maisch, M. Sc.

Plagiate in Programmieraufgaben sind ein häufiges Problem. Diese Arbeit untersucht, inwieweit Methoden des Sequenzalignments aus der Bioinformatik zur Verbesserung bestehender Systeme zur Plagiatserkennung, wie JPlag, MOSS oder DOLOS, beitragen können. Dafür werden kurz ein Verfahren der Plagiatserkennung sowie gängige Methoden des Sequenzalignments vorgestellt und verglichen. Die Analyse zeigt, dass einige Verfahren vielversprechende Impulse für eine Weiterentwicklung der Plagiatserkennung bieten. Einschränkungen ergeben sich jedoch durch die semantische Komplexität von Quellcode und die begrenzte Übertragbarkeit mancher Verfahren.

1 Motivation

Plagiate sind in der heutigen Bildungswelt ein ernstzunehmendes Problem. Man spricht von einem Plagiat, wenn die Arbeit einer anderen Person übernommen und als die eigene ausgegeben wird, ohne dies zu kennzeichnen. Eine präzise Definition von Plagiarismus im akademischen Zusammenhang mit Quellcode liefern Cosma und Joy:
„Source-code plagiarism in programming assignments can occur when a student reuses [...] source-code authored by someone else and, intentionally or unintentionally, fails to acknowledge it adequately [...], thus submitting it as his/her own work. This involves obtaining [...] the source-code, either with or without the permission of the original author, and reusing [...] source-code produced as part of another assessment [...] without adequate acknowledgement [...]“ [1].
Diese Definition macht deutlich, wie vielfältig und schwer erkennbar Plagiate sein können. Deshalb ist es wichtig, zuverlässige Methoden zur Erkennung von Plagiaten zu entwickeln und anzuwenden. Ein Beispiel für ein solches Erkennungssystem ist *JPlag*[1], ein Tool, das speziell zur Aufdeckung von Plagiaten in Programmieraufgaben entwickelt wurde, und

[1] https://github.com/jplag/JPlag

das in diesem Zusammenhang eingesetzt wird [16]. Die Funktionsweise und der Einsatz des Tools werden im nächsten Kapitel erläutert.

Ein ähnliches Problem gibt es in der Bioinformatik: das sogenannte Sequenzalignment. Dabei werden DNA-, RNA- oder auch Proteinsequenzen sequenziert und ausgerichtet. Ziel ist es, Teilsequenzen zu identifizieren, die auf eine gemeinsame Herkunft hindeuten. Das Sequenzalignment ist dabei nur ein Zwischenschritt, der anschließend für weitere Analysen verwendet wird. Dazu zählen die Identifizierung und Quantifizierung bestimmter Regionen oder Motive. Außerdem können genetisch bedingte Krankheiten erkannt und Stammbäume identifiziert und vorhergesagt werden. [15]

In der vorliegenden Arbeit wird zunächst auf das Plagiatserkennungssystem JPlag eingegangen und anschließend werden einige Methoden des Sequenzalignments vorgestellt. Im weiteren Verlauf werden die Gemeinsamkeiten und Unterschiede zu den aktuellen Verfahren der Plagiatserkennung untersucht. Abschließend werden die Aspekte des Sequenzalignments zusammengefasst, die zur Verbesserung der Plagiatserkennung geeignet sind.

2 Aktueller Stand Plagiatserkennung

Die automatisierte Erkennung von Plagiaten ist ein aktives Forschungsfeld. Ziel ist es, trotz Umbenennungen oder Umstrukturierungen Ähnlichkeiten im Quellcode zu finden und zu analysieren. Dazu wurden verschiedene Verfahren entwickelt. Im Folgenden wird zunächst die tokenbasierte Plagiatserkennung vorgestellt und anschließend das Tool JPlag.

2.1 Tokenbasierte Plagiatserkennung

Ein Ansatz zur Erkennung von Plagiaten ist der tokenbasierte Ansatz. Bei diesem Ansatz wird der Quelltext nicht direkt verglichen, sondern zunächst in Tokens umgewandelt. Dabei werden Details wie Variablennamen, Leerzeilen und Kommentare weggelassen. Anschließend werden diese Token-Sequenzen paarweise miteinander verglichen. In Abb. 1 ist ein Beispiel [16] zum Code-Mapping der Token zu sehen. Es wird ein weiterer Parameter verwendet, der die minimale Token-Länge festlegt. Dadurch werden zu kleine ähnliche Sequenzen nicht als Plagiat gewertet und es werden weniger falsch-positive Plagiate erkannt [17].

2.2 JPlag

JPlag wurde 1996 am Karlsruher Institut für Technologie (KIT) entwickelt, um Plagiate in Programmieraufgaben einfacher aufzuspüren. Dabei handelt es sich um ein tokenbasiertes Tool zur Plagiatserkennung in Source-Code, welches anfangs nur ein paar wenige Programmiersprachen (Java, C, C++ und Scheme) unterstützte. Es wird aktiv weiterentwickelt und ist das am häufigsten in wissenschaftlicher Literatur referenzierte Tool [16]. Mittlerweile werden 18 Programmiersprachen von dem Tool unterstützt. Dabei nimmt JPlag mehrere

```
void printSquares() {     method start
  int i = 1;              variable
  while (i <= 10) {       loop start
    int square = i * i;   variable
    println(square);      apply
    i++;                  assignment
  }                       loop end
}                         method end
```

Abbildung 1: Hier sieht man eine beispielhafte Zuweisung von Tokens (rechts) zum Code (links).

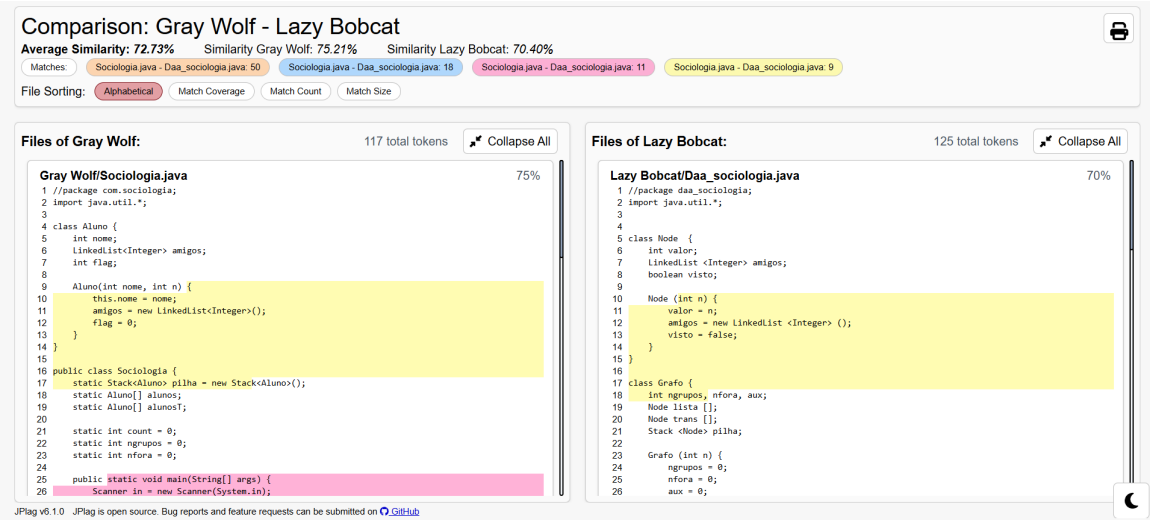

Abbildung 2: JPlag Vergleichsansicht

Programme als Eingabe und vergleicht sie paarweise. Als Ausgabe erhält man eine Datei, welche eine Report-Ansicht ermöglicht, die die gefundenen Stellen aufzeigt und somit bei der Prüfung unterstützt(siehe Abb. 2) [14].

Der Vergleichsalgorithmus zum Vergleichen arbeitet dabei in drei Schritten:

1. Als Erstes wird der Quellcode in eine Baumstruktur, beispielsweise einen abstrakten Syntaxbaum (AST), übertragen [17].

2. Nun werden aus dieser Baumstruktur die relevanten Token extrahiert [17].

3. Als nächstes werden diese Token-Strings nun paarweise verglichen und die Ähnlichkeit dieser Paare bestimmt. Dabei kommt *Greedy String Tiling* mit *Karp-Rabing Matching* zum Einsatz, wodurch ähnliche Teilstrings erkannt werden können. Dadurch können viele Programme in kurzer Zeit verglichen werden [16].

Als Ausgabe erhält man eine Datei, die eine Übersicht über alle Abgaben und deren Ähnlichkeitswerte bietet (Abb. 3).

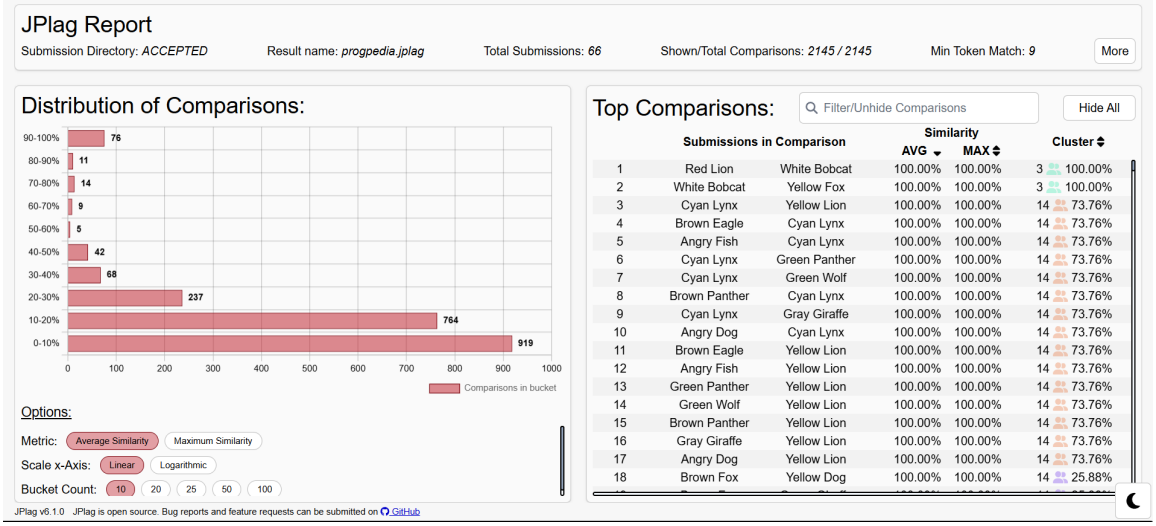

Abbildung 3: JPlag Report-Ansicht

3 Sequenzalignment-Methoden

Das Sequenzalignment ist eine zentrale Methode der Bioinformatik zur Analyse von biologischen Sequenzen wie DNA, RNA oder Proteinen. Ziel ist es, Ähnlichkeiten zwischen diesen Sequenzen zu identifizieren und somit funktionelle oder evolutionäre Beziehungen zu entdecken. In diesem Kapitel werden verschiedene Methoden vorgestellt und anhand ihrer Funktionsweisen klassifiziert. Hierbei sind sie in *Pairwise Alignment* und *Multiple Alignment* unterteilt. Beim Pairwise Alignment werden zwei Sequenzen miteinander verglichen. Beim Multiple Alignment werden drei oder mehr betrachtet. Die Grundlagen und klassischen Methoden (Needleman-Wunsch und Smith-Waterman) basieren auf den Arbeiten von Rosenberg [15], List [8] und Gupta et. al. [6].

3.1 Grundlagen

Man kann grundlegend zwischen zwei Arten von Methoden des Sequenzalignments unterscheiden, dem *Pairwise Sequence Alignment* (PSA) und dem *Multiple Sequence Alignment* (MSA). Beim PSA werden zwei Sequenzen miteinander verglichen und es können die bestmöglichen Lösungen bestimmt werden. Dabei könnte man alle Variationen durchgehen, welche es gibt, aber dies würde, besonders bei langen Sequenzen, zu einem hohen Aufwand führen. Dies kennt man bereits unter brute force. In Abb. 5 sieht man dabei ein schlechtes Alignment der Wörter "Wasser" und "Water", während Abb. 6 das bestmögliche Alignment der beiden Wörter zeigt. Beim Vergleich der Zeichen können drei verschiedene Fälle auftreten:

1. Die beiden Zeichen sind identisch.
2. Die beiden Zeichen sind unterschiedlich.
3. Ein Zeichen wird mit einer Lücke (*Gap*) ausgerichtet.

Vergleich der Zeichen	Punkte	Beispiel
Zeichen sind identisch	1	A/A
Zeichen sind unterschiedlich	-3	A/B
Zeichen wird mit Lücke ausgerichtet	-4	A/-

Abbildung 4: Das ist die Punkteverteilung je nach möglichem Fall. Mit dieser Verteilung probiert man zu erreichen, das beste Alignment zu finden, bei dem sich möglichst viele Zeichen überschneiden und wenig Lücken eingefügt werden. Dadurch wird nämlich eine höhere Punktezahl erreicht.

W	A	S	S	E	R	-	-	-	-	-
-	-	-	-	-	-	W	A	T	E	R

Abbildung 5: Ein mögliches Alignment der Wörter Wasser und Water, welches aber nicht ideal ist.

Je nach Fall wird ein Punktewert vergeben, wie in Abb. 4 dargestellt. Aus der Summe dieser Werte ergibt sich ein Gesamtwert, welcher das aktuell betrachtete Alignment bewertet. Somit ergibt sich für das Beispiel in Abb. 5 ein Wert von $11 * (-4) = -44$, für das beste Alignment jedoch $4 * 1 + 1 * (-3) + 1 * (-4) = -3$.

Beim MSA werden drei oder mehr Sequenzen gleichzeitig verglichen. Aufgrund der exponentiell wachsenden Anzahl an möglichen Kombinationen bei vielen und langen Sequenzen ist es sehr rechenintensiv, eine exakte Lösung zu berechnen. Daher basieren MSA-Methoden oft auf heuristischen Verfahren, die näherungsweise gute, aber nicht immer optimale Ergebnisse liefern.

Neben der Unterteilung in MSA und PSA lässt sich auch beim PSA zwischen zwei Ausrichtungsstrategien unterscheiden: dem *Local Alignment* und dem *Global Alignment*.

Beim *Global Alignment* werden zwei Sequenzen über ihre komplette Länge verglichen. Ein bekanntes Verfahren dafür ist der *Needleman-Wunsch-Algorithmus*, der auf *Dynamischer-Programmierung* (DP) basiert. Im Gegensatz dazu sucht das *Local Alignment* nach Abschnitten in den Sequenzen, welche sich ähnlich sind, sich sonst aber stark unterscheiden. Der *Smith-Waterman-Algorithmus* wäre ein Beispiel für ein solches *lokales Alignment*, welcher auch auf DP beruht. Es könnte in einigen Fällen beim *Global Alignment* dazu kommen, dass am Anfang oder Ende Lücken eingefügt werden müssen, was somit zu hohen Strafen in der Punkteverteilung führen würde. Deshalb benötigt man einen Mix aus *Local* und *Global Alignment*, dem *Semiglobal Alignment*. Dabei werden Start- und Endlücken eines *Global Alignment* nicht bestraft. Somit können sich die Enden der Sequenzen wieder ausrichten, ohne eine Strafe zu bekommen.

W	A	S	S	E	R
W	A	-	T	E	R

Abbildung 6: Das beste Alignment der Wörter Wasser und Water.

3.2 Pairwise Alignment

Beim Pairwise Sequence Alignment (PSA) werden zwei Sequenzen direkt miteinander verglichen, um Ähnlichkeiten zu identifizieren. Im Folgenden werden einige Verfahren des PSA vorgestellt.

3.2.1 Needleman-Wunsch-Algorithmus

Der *Needleman-Wunsch-Algorithmus* [11] war die erste bekannte Lösung für das Alignment-Problem. Er wurde 1970 von Saul B. Needleman und Christian D. Wunsch veröffentlicht. Der Algorithmus basiert auf der Idee des DP. Dabei wird, anstelle der Prüfung aller möglichen Alignments zweier Sequenzen, die optimale Lösung vorheriger kleinerer Teilsequenzen weiterverwendet, um ein globales Alignment zu bestimmen. Dabei unterscheidet man dieselben Fälle wie zuvor. Wir nehmen hier zur Illustration folgende Punkteverteilung an:

- Zeichen sind identisch: 1
- Zeichen sind unterschiedlich: -1
- Zeichen wird mit Lücke (Gap) ausgerichtet: -1

Es werden zwei Sequenzen der Länge n und m verwendet. Der Algorithmus arbeitet nun drei Schritte ab:

1. Es wird eine Matrix S der Größe $(n + 1) \times (m + 1)$ erstellt und initialisiert. Dabei werden die erste Zeile und Spalte mit Vielfachen der Kosten einer Lücke gefüllt.
2. Hier findet die eigentliche Rekursion statt. Die Matrix wird dabei mit folgender Formel befüllt:

$$S(i,j) = \max \begin{cases} S(i-1, j-1) + \text{score}(x_i, y_j) & \text{(Diagonal: Match/Mismatch)} \\ S(i-1, j) + \text{gap} & \text{(Oben: Lücke in y)} \\ S(i, j-1) + \text{gap} & \text{(Links: Lücke in x)} \end{cases}$$

   wobei ”score" für die Kosten eines Match oder Mismatch zweier Token und "gap" für die Kosten einer Lücke steht. Der optimale Alignment-Wert steht nun an Stelle $S(n, m)$. Dabei wird gespeichert, welcher Fall angewendet wurde, da im ersten Fall von einem Match oder Mismatch ausgegangen wird, im zweiten Fall wird eine Lücke in das Wort, das oben steht, eingefügt und im letzten Fall in dem Wort, das links steht.
3. Nun kann mithilfe von Backtracking das beste Alignment zurückverfolgt werden.

In Abb. 7 und 8 wird die initialisierte und befüllte Matrix für die Wörter „Wasser” und „Water” veranschaulicht. In der befüllten Matrix sind dabei noch Pfeile zu sehen, welche zeigen, welchen Wert man für die Berechnung genutzt hat. Aus Abb. 8 kann das optimale Alignment hergeleitet werden, welches bereits in Abb. 6 zu sehen ist.

-		W	A	T	E	R
	0	-1	-2	-3	-4	-5
W	-1					
A	-2					
S	-3					
S	-4					
E	-5					
R	-6					

Abbildung 7: Initialisierung der Matrix des Needleman-Wunsch-Algortihmus

-		W	A	T	E	R
	0	-1 ←	-2 ←	-3 ←	-4 ←	-5 ←
W	-1 ↑	1 ↖	0 ←	-1 ←	-2 ←	-3 ←
A	-2 ↑	0 ↑	2 ↖	1 ←	0 ←	-1 ←
S	-3 ↑	-1 ↑	1 ↑	1 ↖	0 ↖	-1 ←
S	-4 ↑	-2 ↑	0 ↑	0 ↖	0 ↖	-1 ←
E	-5 ↑	-3 ↑	-1 ↑	-1 ↑	1 ↖	0 ←
R	-6 ↑	-4 ↑	-2 ↑	-2 ↑	0 ↑	2 ↖

Abbildung 8: Befüllte Matrix anhand der Formel des Needleman-Wunsch-Algorithmus

3.2.2 Smith-Waterman-Algorithmus

Der *Smith-Waterman-Algorithmus* ist eine Erweiterung des *Needleman-Wunsch-Algorithmus*, welcher 1981 von T. F. Smith und M. S. Waterman veröffentlicht wurde. Mit ihm kann ein lokales Alignment zweier Sequenzen berechnet werden. Dabei wird ein weiterer Fall zur Punkteverteilung hinzugefügt, welcher verhindert, dass die Wertung in der Matrix negativ wird. In diesem Fall wird eine Null in der Matrix gespeichert und es werden keine Pfeile für das Backtracking gespeichert. Damit ergibt sich dann folgende Formel zum Befüllen der Matrix:

$$S(i,j) = \max \begin{cases} S(i-1, j-1) + \text{score}(x_i, y_j) & \text{(Diagonal: Match/Mismatch)} \\ S(i-1, j) + \text{gap} & \text{(Oben: Lücke in y)} \\ S(i, j-1) + \text{gap} & \text{(Links: Lücke in x)} \\ 0 \end{cases}$$

Eine weitere Änderung ist, dass bei der Initialisierung die erste Spalte und Zeile hier mit Nullen gefüllt werden.
Um letztendlich das lokale Alignment zu bestimmen, muss man die maximale Wertung in der Matrix finden und von dort aus das Backtracking starten. Es ist dabei möglich, dass es mehrere bestmögliche Alignments geben kann. Dies wird an den Wörtern "Nebel" und "Leben" in den Abb. 9 und 10 dargestellt.

		L	E	B	E	N
	0	0	0	0	0	0
N	0					
E	0					
B	0					
E	0					
L	0					

Abbildung 9: Initialisierung der Matrix des Smith-Waterman-Algorithmus.

		L	E	B	E	N
	0	0	0	0	0	0
N	0	0	0	0	0	1 ↖
E	0	0	1 ↖	0	1 ↖	0
B	0	0	0	2 ↖	1 ←	0
E	0	0	1 ↖	1 ↑	3 ↖	2 ←
L	0	1 ↖	0	0	2 ↑	2 ↖

Abbildung 10: Befüllte Matrix mit dem Smith-Waterman-Algorithmus mit Backtracking-Pfeilen

In Abb. 10 befindet sich der maximale Wert 3 an Stelle $S(5,5)$. Nun kann man von da aus mit Hilfe des Backtrackings das beste lokale Alignment bestimmen, welches in Abb. 11 zu sehen ist. Dabei kann es vorkommen, dass man nur eine Teilsequenz der Eingabe erhält.

3.2.3 BLAST

Das BLAST-Tool ("Basic Local Alignment Search Tool") verwendet einen heuristischen Ansatz, bei dem kurze Übereinstimmungen zwischen Sequenzen, sogenannte *Hot-Spots*, gefunden werden. Dabei vergleicht es Sequenzen mit Sequenzdatenbanken. Von diesen "Hot-Spots" werden anschließend Alignments gesucht. Dabei liefert es nicht nur ein Alignment, sondern auch den *E-Wert*, welcher eine Wahrscheinlichkeit ist, dass ein Treffer zufällig zustande gekommen ist. Dabei gilt, dass ein kleiner E-Wert dafür spricht, dass die Sequenz nicht nur zufällig übereinstimmt [6].

E B E
E B E

Abbildung 11: Lokales Alignment der Wörter Nebel und Leben, welches man durch die Ausführung von Smith-Waterman erhält. Da es ein lokales Alignment ist, sieht man auch nur ein Teil der Eingabe.

3.2.4 FASTA

Die FASTA-Methode ist ein Verfahren zum Vergleich von DNA- oder Proteinsequenzen. Sie wurde als Erweiterung der FASTP-Methode [7] entwickelt. Der Algorithmus läuft in vier Schritten ab: [13]

1. Im ersten Schritt werden mit Hilfe einer Lookup-Tabelle die Übereinstimmungen der zwei Sequenzen gefunden. Dabei definiert der *ktup* (k-tupel) Parameter wie viele gleiche Zeichen hintereinander folgen müssen. Nun werden die zehn besten Übereinstimmungen anhand einer Formel bestimmt, die auf dem ktup-Parameter und dem Abstand der Übereinstimmung in den Sequenzen basiert.

2. Im nächsten Schritt werden diese zehn Übereinstimmung mit einer Bewertungsmatrix neu bewertet, wobei Ersetzungen und Zeichenfolgen kleiner als ktup berücksichtigt werden. Daraus ergeben sich die sogenannten *inital regions* [13].

3. Im dritten Schritt werden nun mehrere nahe liegende *initial regions* zu einem Alignment kombiniert, unter Berücksichtigung einer Strafe für das Verknüpfen. In anderen Verfahren wären das die Kosten für eine Lücke. In FASTP würde hier die beste *initial region* ausgegeben werden, ohne sie zu kombinieren.

4. Im letzten Schritt wird im Bereich um die zuvor am besten bewertete Region eine Variante der Needleman-Wunsch- und Smith-Waterman-Algorithmen genutzt. Dabei werden lokale Alignments der Sequenzen in diesem begrenzten Abschnitt bestimmt, um das optimale lokale Alignment mit dem höchsten Score zu ermitteln.

3.2.5 TransIndel

TransIndel [19] ist ein Tool, welches zur Detektion von Insertions (Einfügeoperationen) und Deletions (Löschoperationen), den sogenannten *Indels*, geschaffen wurde. Dabei vergleicht es Genomsequenzen, die aus bestimmten Sequenzsegmenten bestehen, die auch *chimeric read* genannt werden. Dafür wird über zwei Sequenzen mit einer festen Fenstergröße iteriert und mögliche Alignments gesucht. In diesen werden dann Einfügungen und Löschungen ermittelt. Wird ein zusätzliches chimeric read in der ”Zielsequenz" gefunden, dann wird es als Löschung in der Lesesequenz markiert. Umgekehrt würde eine zusätzliche Sequenz als Einfügung markiert werden. Somit ist TransIndel robust gegenüber Indels und kann Sequenzen ausrichten [16].

3.3 Multiple Alignment

Beim Multiple Sequence Alignment (MSA) werden, im Gegensatz zu Pairwise Alignment, mehr als zwei Sequenzen gleichzeitig verglichen. Beide verfolgen aber dasselbe Ziel: Eine möglichst gute Ausrichtung für die gegebenen Sequenzen zu bestimmen. Da mit steigender Anzahl an Sequenzen die Anzahl an möglichen Alignments exponentiell steigt, werden hierbei öfter heuristische oder approximative Methoden verwendet. Ein MSA induziert dabei auch ein PSA zwischen allen beteiligten Sequenzen, diese sind jedoch nicht zwingend optimal. Das MSA probiert die ideale Ausrichtung über mehrere Sequenzen

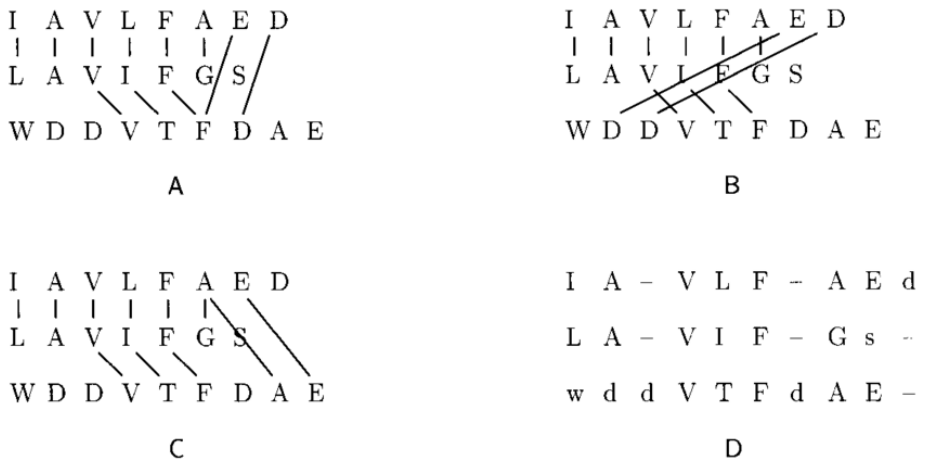

Abbildung 12: Hier sieht man konsistente und nicht-konsistente Fragmente. A und B sind nicht konsistent, da in A das F in der dritten Sequenz zwei anderen zugeordnet ist und B überkreuzen sich die Zuordnungen. C ist eine konsistente Zuordnung und durch das Einfügen von Lücken erhält man D [10].

zu finden, wodurch Details auf Paarebene verloren gehen. Im Folgenden werden einige MSA-Verfahren vorgestellt.

3.3.1 DIALIGN

DIALIGN ist eine Methode, welche zum Pairwise und Multiple Alignment genutzt werden kann, welche 1998 veröffentlicht wurde [10]. Dabei werden ganze Segmente verglichen, statt wie bislang einzelne Token. Es wird außerdem auf die Strafe für das Einfügen einer Lücke verzichtet. Dies ist vor allem nützlich, wenn Sequenzen nur lokale Ähnlichkeiten aufweisen. Die möglichen Alignments, die so gefunden werden, werden Fragmente oder Diagonalen genannt [15]. Man kann den Algorithmus in 5 Schritte aufteilen [10]:

1. Zunächst werden alle paarweisen Alignments zwischen den Sequenzen durchgeführt.

2. Nun werden die Alignments/Fragmente nach ihrem Score und ihrer Konsistenz sortiert. Die Konsistenz gibt dabei an, dass sich keine Alignments überschneiden oder widersprechen. In Abb. 12 ist ein Beispiel von B. Morgenstern, welches die Konsistenz zeigt [10].

3. Man wählt das beste Alignment/Fragment und nimmt es für das Alignment an.

4. Im nächsten Schritt wählt man so lange das nächste Alignment/Fragment und prüft es auf Konsistenz, bis keines mehr übrig ist.

5. Nun fügt man Lücken so ein, dass alle Alignments/Fragmente miteinander verbunden sind. Dabei wird in Großbuchstaben geschrieben, welche Token genutzt und in Kleinbuchstaben, welche nicht genutzt werden.

3.3.2 ClustalW

ClustalW ist eine weit verbreitete Methode des Multiple Sequence Alignments (MSA), welche 1994 veröffentlicht wurde [18]. Die Methode erstellt dabei ein globales Alignment und basiert auf dem Prinzip des *progressiven Alignments*. Das *progressive Alignment* ist dabei ein heuristischer Ansatz, welcher keinen Alignment-Score optimiert. Dafür werden zunächst PSA durchgeführt, wobei mit den ähnlichsten Sequenzen begonnen wird und anschließend immer weniger ähnliche Sequenzen verwendet werden. Durch die Aufteilung in mehrere PSA und anschließende Erstellung eines *guide trees* [6] erreicht man eine Laufzeit-Optimierung des MSA. Der *guide tree* ist dabei ein Baum, der auf der Ähnlichkeit der Sequenzen basiert und somit die Reihenfolge des Alignments bestimmt [15]. So funktioniert auch Clustal, welcher in 3 Schritten arbeitet [18]:

1. Es wird eine Distanzmatrix für jedes Paar von Sequenzen erstellt.

2. Nun wird ein *guide tree* mit Hilfe der Distanzmatrix erstellt.

3. Im letzten Schritt werden die Sequenzen entsprechend der Verzweigungsreihenfolge im *guide tree* ausgerichtet.

Es können dabei aber Fehler entstehen, vor allem zu Beginn, da der guide tree mit einem greedy-Ansatz erstellt wird. Um diese Fehler auszugleichen, müsste man noch ein weiteres iteratives oder stochastisches Verfahren anwenden [18].

3.3.3 MUSCLE

MUSCLE (multiple sequence comparison by log-expectation) wurde 2004 von Robert C. Edgar veröffentlicht [4]. Es erstellt ein multiple Alignment mit hoher Genauigkeit und Effizienz. Der Algorithmus besteht aus drei Stufen [4]:

1. *Draft progressive*: Hierbei ist das Ziel ein schnelles Alignment, statt eines genauen, zu erstellen.

2. *Improved progressive*: Hier wird nun der Fehler, der durch die vorherige Stufe entsteht, minimiert. Der Fehler entsteht durch die Nutzung der k-mer-Distanz (ähnlich zu einem k-tupel). Um den Fehler zu minimieren, wird nun die Kimura-Distanz verwendet, welche aber ein Alignment voraussetzt.

3. *Refinement*: In der Refinement-Phase wird das vorherige Alignment verbessert, indem der Baum in Teilbereiche aufgeteilt und diese gezielt neu ausgerichtet werden. Verbesserungen werden nur übernommen, wenn sie die Qualität des Alignments nachweislich steigern. Dieser Prozess wird so lange wiederholt, bis er konvergiert oder ein Abbruchkriterium erreicht ist (z. B. ein bestimmter Score des Alignments).

In Abb. 13 ist ein genauer Überblick über die Schritte des Algorithmus dargestellt. [3]

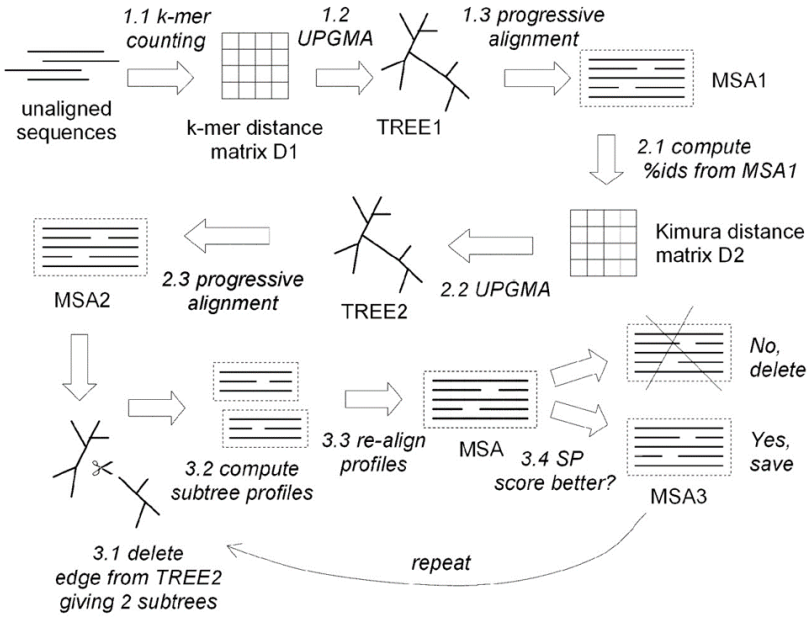

Abbildung 13: Ein Überblick über die genauen Schritte des Algorithmus von MUSCLE. (Abb. von Robert C Edgar[4])

3.3.4 T-Coffee

T-Coffee (Tree-based Consistency Objective Function for alignment Evaluation) ist ein Algorithmus für ein multiples Alignment, welcher durch die Kombination von globalem und lokalem Alignment eine hohe Genauigkeit erzielt [5]. Dabei wird ClustalW für die globalen Alignments genutzt und LAlign, welcher ein Teil von FASTA ist, für die lokalen Alignments. Die Methode läuft in vier Schritten ab [12]:

1. Es werden zuerst die sogenannten *Primär-Bibliotheken* erzeugt, welche jeweils die globalen Alignments aller Paare und die zehn besten lokalen Alignments pro Paar speichert. Jedes Alignment erhält ein Gewicht, welches der prozentualen Gleichheit des Alignments mit der Sequenz, von der es stammt, angibt.

2. Im nächsten Schritt werden diese zwei Bibliotheken zusammengefügt. Wenn sich Paare doppeln, werden die Gewichte summiert .

3. Nun wird die Bibliothek erweitert. Mithilfe von „Triplet“-Vergleichen (Dreiervergleiche) wird geprüft, ob ein Paar durch andere Sequenzen zusätzlich gestützt wird. Je mehr indirekte Unterstützung vorhanden ist, desto höher wird das Gewicht. So fließt konsistente Information aus allen Sequenzen in die Bewertung ein.

4. Nun wird progressives Alignment genutzt, ähnlich wie bei ClustalW. Dabei wird jedoch auf die Kosten einer Lücke verzichtet, da diese bereits in früheren Phasen berücksichtigt wurden.

In Abb. 14 ist ein Beispiel von C. Notredame et al. [12], das zeigt, wie diese Primär-Bibliothek und die Erweiterung aussehen.

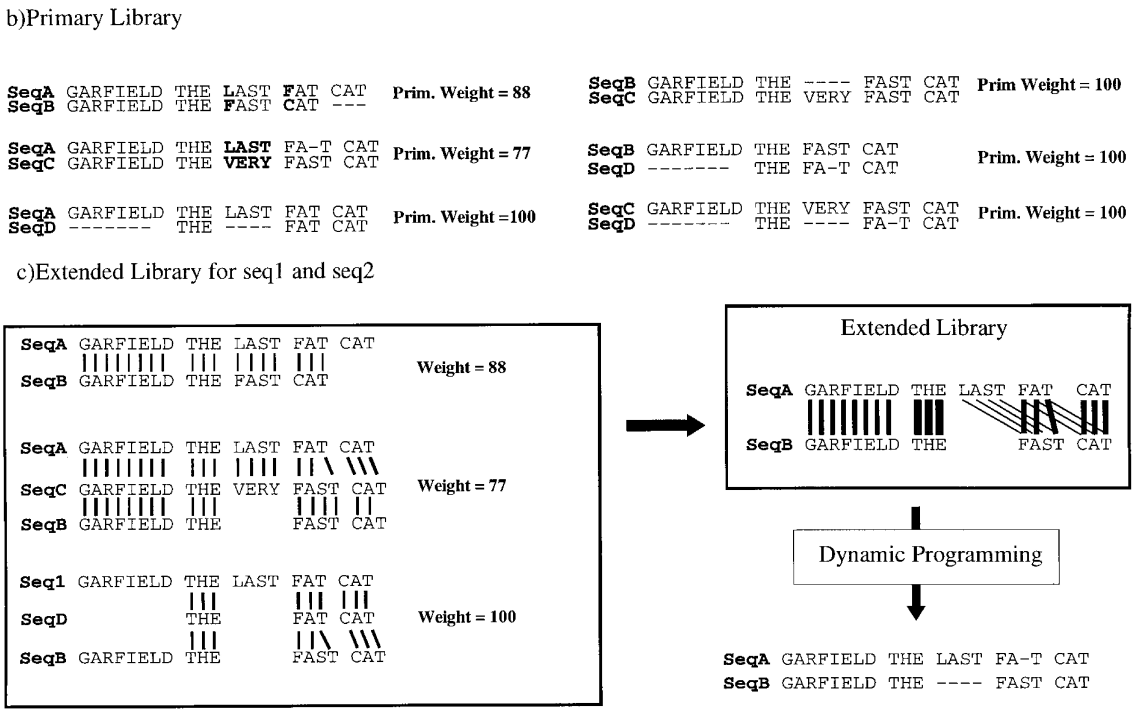

Abbildung 14: b zeigt eine primäre Bibliothek, mit dem zugeordnetem Gewicht, wobei die Nichtübereinstimmungen fettgedruckt sind. c zeigt einen Teil der erweiterten Bibliothek, in der alle drei Ausrichtungen zwischen A und B zu sehen sind. Die Dicke der Linien gibt dabei die Größe des Gewichts an. Mithilfe der dynamischen Programmierung wird schließlich das finale Alignment gefunden [12].

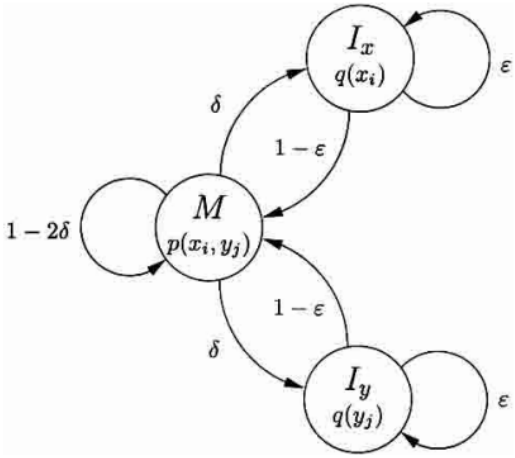

Abbildung 15: In dieser Abbildung sieht man ein beispielhaftes HMM. Zustand M steht dafür, dass aus beiden Sequenzen ein Buchstabe ausgerichtet wird. Zustand I_x bedeutet, dass ein Buchstabe zu einer Lücke ausgerichtet wird (Zustand I_y analog) [2]. (Abb. von Chuong B. Do et al., [2])

3.3.5 ProbCons

Der folgende Absatz basiert auf dem Artikel [2][Choung B. Do et al., 2005]. ProbCons (Probabilistic consistency-based multiple sequence alignment) ist ein Algorithmus, der auf *progressivem Alignment* und *pair-hidden Markov model* (HMM) basiert. Er verwendet eine maximal erwartete Genauigkeit als Maßstab für das Alignment und die probabilistische Konsistenztransformation zur Einbeziehung von Informationen über die Erhaltung mehrerer Sequenzen beim paarweisen Abgleich. Die probabilistische Konsistenztransformation hilft dabei, beim Vergleich von zwei Sequenzen auch Informationen aus den anderen Sequenzen zu berücksichtigen. Dabei wird folgendes HMM verwendet:

Der Algorithmus wird in 5 Schritte aufgeteilt:

1. Berechnung der Posterior-Wahrscheinlichkeitsmatrizen.
2. Berechnung der erwarteten Genauigkeiten.
3. Probabilistische Konsistenztransformation.
4. Berechung des *guide tree*.
5. Nun wird noch ein *Progressive Alignment* durchgeführt.

In Abb. 15 ist so ein beispielhaftes HMM abgebildet [2].

4 Vergleich Plagiatserkennung und Sequenzalignment

Nachdem im vorherigen Kapitel einige Methoden des Sequenzalignments vorgestellt wurden, soll in diesem Abschnitt untersucht werden, inwiefern sich die Ansätze mit bestehenden Verfahren der Plagiatserkennung (JPlag) vergleichen lassen. Sowohl die Bioinformatik

als auch die Informatik im Kontext der Plagiatserkennung stehen vor der Herausforderung, Ähnlichkeiten zwischen komplexen, strukturierten Daten zu identifizieren. Obwohl sich die Anwendungsgebiete unterscheiden, verfolgen beide das gleiche Ziel: In einer Vielzahl von verschiedenen Daten und Kombinationen mögliche Überschneidungen zu finden. Zunächst werden im Folgenden die Gemeinsamkeiten und anschließend die Unterschiede herausgearbeitet.

4.1 Gemeinsamkeiten

Obwohl die Plagiatserkennung von Quellcode und das Sequenzalignment der Bioinformatik auf den ersten Blick völlig unterschiedlichen Aufgabenbereichen entstammen, lassen sich bei genauerer Betrachtung Gemeinsamkeiten erkennen. Ziel beider Verfahren ist es, in großen Mengen von Daten bedeutungsvolle Überschneidungen zu finden, um beispielsweise Programmteile oder genetische Verwandtschaften aufzudecken.

Ein zentraler gemeinsamer Aspekt ist die Behandlung der Eingabedaten als lineare Sequenzen. In der Plagiatserkennung wird der Quellcode vorverarbeitet und in eine Folge von Tokens umgewandelt. Ähnlich ist es beim Sequenzalignment, bei dem DNA-, RNA- oder Proteinsequenzen analysiert werden. In beiden Fällen besteht die Aufgabe darin, ähnliche Abschnitte zu finden, wobei kleinere Unterschiede wie Umbenennungen oder Mutationen nicht als trennendes Merkmal auffallen sollten.

Zur Bewertung der Ähnlichkeit nutzen beide Disziplinen Scoring-Systeme. In der Plagiatserkennung wird beispielsweise ermittelt, wie viele identische oder ähnliche Token-Folgen zwischen zwei verschiedenen Quellcodes vorhanden sind. Beim Sequenzalignment werden hingegen Matches, Mismatches und Lücken (Gaps) unterschiedlich gewichtet, um ein optimales Alignment zu berechnen. Die Bewertungsfunktion beeinflusst somit direkt, welche Bereiche als ähnlich eingestuft werden. Dabei sind beide Ansätze darauf ausgelegt, kleinere Abweichungen zu tolerieren. In der Plagiatserkennung können Umstrukturierungen im Code, das Umbenennen von Variablen oder das Einfügen redundanter Zeilen auftreten. Vergleichbare Phänomene in der Bioinformatik sind beispielsweise Mutationen oder Indels in Sequenzen. Die Algorithmen müssen also auch bei solchen Modifikationen in der Lage sein, die Ähnlichkeiten zu erkennen. Dies kann teilweise durch die Scoring-Funktion bestimmt werden, indem man beispielsweise in der Bioinformatik Zeichen, welche sich durch Mutationen ähneln, anders bewertet, oder in der Plagiatserkennung einige Token-Typen zusammenfassen kann. Abb. 16, von R. Maisch et al., veranschaulicht die Gruppierung ähnlicher Token-Typen [9].

Diese Überschneidungen legen nahe, dass Erkenntnisse aus dem Sequenzalignment auch nützlich für die Plagiatserkennung sein könnten.

4.2 Unterschiede

Trotz der Gemeinsamkeiten zwischen der Plagiatserkennung und dem Sequenzalignment gibt es auch Unterschiede, die sich vor allem aus der jeweiligen Anwendung und den

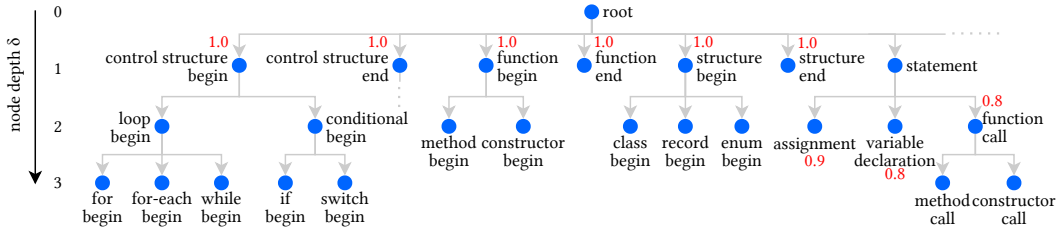

Abbildung 16: Hier sieht man einen Baum von Java-Token-Typen. Die Token sind hierbei nach ihrer Ähnlichkeit angeordnet. Je tiefer dabei ein Token liegt, desto spezifischer ist der Token-Typ in seinem Kontext.

Datenstrukturen ergeben. Diese Unterschiede betreffen sowohl die Art der Daten als auch die Zielsetzung der Analyseverfahren.
Ein zentraler Unterschied liegt in der Art der zu vergleichenden Sequenzen. Beim Sequenzalignment handelt es sich um biologische Daten (DNA-, RNA- oder Proteinsequenzen), die aus einer festen und begrenzten Anzahl von Typen bestehen. Im Quellcode hingegen gibt es deutlich komplexere Strukturen. Programmiersprachen enthalten viele logische Strukturen, wie Schleifen, Abfragen und Methoden. Diese logische Hierarchie muss bei der Plagiatserkennung zusätzlich berücksichtigt werden, wodurch die Betrachtung der Daten nochmal schwerer wird.
Auch die Bedeutung der Reihenfolge unterscheidet sich. In der Bioinformatik ist die Reihenfolge in den Sequenzen entscheidend, da sie direkt die biologische Funktion einer Sequenz beeinflusst. Beim Quellcode kann dagegen die Reihenfolge bestimmter Codeabschnitte verändert werden, ohne dass sich die Funktion ändert (z.B. durch Umstrukturierung von Methoden oder das Verschieben von Funktionen in andere Dateien). Plagiatserkennung muss also robuster gegenüber solchen Umstellungen sein, was eine zusätzliche Anforderung darstellt.
Ein weiterer Unterschied besteht in der Zielsetzung. Während man beim Sequenzalignment biologische Verwandtschaften entdecken möchte, zielt die Plagiatserkennung darauf ab, absichtliches Kopieren und Umgestalten von Code aufzudecken. Dabei muss noch berücksichtigt werden, dass der Code oft bewusst verschleiert wird, in der Hoffnung, nicht erkannt zu werden. Entsprechend müssen diese Systeme auch stark modifizierte, aber semantisch gleiche Abschnitte erkennen.
Schließlich unterscheidet sich auch die Anzahl der zu vergleichenden Daten. In der Bioinformatik werden oft große Sequenzdatenbanken durchsucht, während sich die Plagiatserkennung auf kleinere Gruppen konzentriert. Dies wird in der Arbeit von T. Sağlam deutlich, in der eine Umfrage ergeben hat, dass Datensätze am häufigsten zwischen 50 und 200 Programme enthalten: "The most common reply was regarding datasets containing between 50 and 200 programs, with 38 percent of replies" [16, S.26]. Dafür ist aber die Varianz der "Mutationen” im Code jedoch größer, wie im vorherigen Absatz bereits beschrieben.
Diese Unterschiede verdeutlichen, dass sich Methoden des Sequenzalignments nicht ohne Anpassung auf die Plagiatserkennung anwenden lassen. Dennoch können bestimmte

Prinzipien, wie die Toleranz gegenüber Abweichungen, für die Weiterentwicklung genutzt werden.

5 Anwendbarkeit auf Plagiatserkennung

Die in Kapitel 4 dargestellten Gemeinsamkeiten deuten darauf hin, dass sich bestimmte Konzepte aus der Bioinformatik auch zur Optimierung bestehender Verfahren der Plagiatserkennung eignen könnten. In diesem Kapitel wird daher untersucht, welche Methoden des Sequenzalignments sich auf die Plagiatserkennung übertragen lassen und welche Herausforderungen sich dabei ergeben.

Die grundlegenden Algorithmen der Bioinformatik sind der Needleman-Wunsch-Algorithmus und der Smith-Waterman-Algorithmus. Sie arbeiten beide auf dem Prinzip der dynamischen Programmierung. Somit würden diese beiden Algorithmen aufgrund der hohen Rechenzeit einen hohen Aufwand mit sich bringen. Aber ihre Konzepte können als methodische Grundlage für weitere Ansätze verwendet werden. Vor allem die des Smith-Waterman-Algorithmus, da er lokale Alignments berechnet und somit verwendet werden kann, wenn nicht das gesamte Programm, sondern nur einzelne Fragmente kopiert wurden.

Ein vielversprechenderer Ansatz ist das Prinzip heuristischer Verfahren, wie es durch Tools wie BLAST oder FASTA umgesetzt wurde. Diese suchen nicht nach der global besten Ausrichtung, sondern beginnen mit dem Suchen sogenannter *Hotspots* und bauen darauf Alignments auf. Solche Verfahren wurden entwickelt, da sie schneller auf großen Datenmengen skalieren. Das ist für die Plagiatserkennung interessant, da Programme sich oft nur in wenigen, kurzen Abschnitten überschneiden. Deshalb wäre eine Übertragung der Strategie möglich.

Das TransIndel-Konzept, bei dem gezielt nach Einfügungen und Löschungen gesucht wird, lässt sich gut auf das Problem der *code-obfuscation* übertragen. Der Ansatz, bei dem mit einem festen Fenster iteriert wird, ist in JPlag bereits mit *greedy-string-tiling* umgesetzt. Dadurch wird die Robustheit gegenüber code-obfuscation sichergestellt [16].

DIALIGN vergleicht direkt Segmente (Fragmente) und verzichtet dabei darauf, die Lücken zu bestrafen. Dadurch wird das Verfahren robuster gegenüber lokalen Ähnlichkeiten bei gleichzeitig größeren globalen Unterschieden, was einem typischen Muster bei Plagiaten entspricht. Dies würde dabei helfen, einzelne Methoden oder Schleifen auf Plagiate zu prüfen. Allerdings führt der fragmentbasierte Ansatz in Kombination mit der erforderlichen Konsistenzprüfung zu einer hohen Komplexität. Ohne zusätzliche Optimierung wäre das Verfahren daher nur eingeschränkt skalierbar und für viele Vergleichspaare wenig effizient.

ClustalW verwendet ein progressives Alignment, welches mit einem guide tree funktioniert, der aus einer Distanzmatrix abgeleitet wird. Fehler, die früh im Prozess entstehen, lassen sich später nicht mehr korrigieren. Daher eignet sich das Verfahren nur bedingt für

die Plagiatserkennung, da es sich nicht um eine verlässliche Methode handelt.

MUSCLE ist eine Erweiterung von ClustalW, die dessen Nachteile verbessert. Es liefert genauere Ergebnisse, leidet dabei jedoch unter einem hohen Rechenaufwand, insbesondere bei vielen Dateien. MUSCLE ist somit gut geeignet, um verdächtige Gruppen detaillierter zu analysieren, für den normalen Gebrauch jedoch weniger.

T-Coffee arbeitet mit verschiedenen Arten von Alignments (lokal und global) und speichert diese in einer Bibliothek. Diese werden zusammengeführt und erweitert. Aufgrund der komplexen Bibliothekserstellung und der dabei entstehenden vielen Vergleiche ist sie eher ungeeignet für die Plagiatserkennung.

ProbCons basiert auf HMMs und einem probabilistischen Ansatz, um ein Alignment zu erstellen. Diese Methode ist aber aufgrund der stochastischen Modellierung und der dabei entstehenden hohen Rechenkomplexität kaum auf Quellcode zu übertragen.

6 Fazit

In dieser Arbeit wurden Methoden des Sequenzalignments aus der Bioinformatik analysiert und mit bestehenden Verfahren der Plagiatserkennung im Quellcode verglichen. Dabei zeigte sich, dass beide Fachgebiete trotz unterschiedlicher Anwendungsbereiche ähnliche Herausforderungen teilen: die Suche nach strukturellen Ähnlichkeiten in Sequenzen. Insbesondere Konzepte wie das lokale Alignment (Smith-Waterman), heuristische Verfahren (BLAST, FASTA) oder fragmentbasierte Ansätze (DIALIGN) bieten interessante Ansätze, um bestehende Plagiatserkennungswerkzeuge, vor allem JPlag, zu verbessern oder weiterzuentwickeln. Es zeigte sich allerdings auch, dass sich nicht alle bioinformatischen Methoden ohne Weiteres auf die Plagiatserkennung übertragen lassen. Gründe hierfür sind die höhere semantische Komplexität von Quellcode und das gezielte Verschleiern von Plagiaten. Darüber hinaus sind einige Verfahren des Sequenzalignments aufgrund ihrer Rechenintensität oder Komplexität für den praktischen Einsatz in der Plagiatserkennung nur eingeschränkt geeignet. Zusammenfassend lässt sich also sagen, dass Methoden der Bioinformatik ein vielversprechendes Potenzial für die Weiterentwicklung der Plagiatserkennung bieten, da beide dieselbe grundlegende Idee verfolgen. Insbesondere wenn diese Methoden gezielt angepasst und mit bestehenden Techniken bzw. Algorithmen kombiniert werden.

Literatur

[1] Georgina Cosma und Mike Joy. „Towards a Definition of Source-Code Plagiarism“. In: *IEEE Transactions on Education* 51.2 (2008), S. 195–200. DOI: 10.1109/TE.2007.906776.

[2] Chuong B Do u. a. „ProbCons: Probabilistic consistency-based multiple sequence alignment“. In: *Genome research* 15.2 (2005), S. 330–340.

[3] Robert C Edgar. „MUSCLE: a multiple sequence alignment method with reduced time and space complexity“. In: *BMC bioinformatics* 5 (2004), S. 1–19.

[4] Robert C Edgar. „MUSCLE: multiple sequence alignment with high accuracy and high throughput“. In: *Nucleic acids research* 32.5 (2004), S. 1792–1797.

[5] Robert C Edgar und Serafim Batzoglou. „Multiple sequence alignment“. In: *Current Opinion in Structural Biology* 16.3 (2006). Nucleic acids/Sequences and topology, S. 368–373. ISSN: 0959-440X. DOI: https://doi.org/10.1016/j.sbi.2006.04.004. URL: https://www.sciencedirect.com/science/article/pii/S0959440X06000704.

[6] Manoj Kumar Gupta u. a. „Sequence alignment“. In: *Bioinformatics in Rice Research: Theories and Techniques* (2021), S. 129–162.

[7] David J Lipman und William R Pearson. „Rapid and sensitive protein similarity searches“. In: *Science* 227.4693 (1985), S. 1435–1441.

[8] Mattis List. *Sequence comparison in historical linguistics*. Bd. 1. Walter de Gruyter GmbH & Co KG, 2014.

[9] Robin Maisch, Nathan Hagel und Alexander Bartel. „Towards Robust Plagiarism Detection in Programming Education: Introducing Tolerant Token Matching Techniques to Counter Novel Obfuscation Methods“. Englisch. In: *ECSEE '25: Proceedings of the 6th European Conference on Software Engineering Education, 2nd – 4th June 2025, Seeon, Germany*. 6th European Conference on Software Engineering Education. 2025 (Seeon, 2.–4. Juni 2025). Association for Computing Machinery (ACM), 2025, S. 11–19. ISBN: 979-8-4007-1282-1. DOI: 10.1145/3723010.3723019.

[10] Burkhard Morgenstern u. a. „DIALIGN: finding local similarities by multiple sequence alignment.“ In: *Bioinformatics (Oxford, England)* 14.3 (1998), S. 290–294.

[11] Saul B. Needleman und Christian D. Wunsch. „A general method applicable to the search for similarities in the amino acid sequence of two proteins“. In: *Journal of Molecular Biology* 48.3 (1970), S. 443–453. ISSN: 0022-2836. DOI: https://doi.org/10.1016/0022-2836(70)90057-4. URL: https://www.sciencedirect.com/science/article/pii/0022283670900574.

[12] Cédric Notredame, Desmond G Higgins und Jaap Heringa. „T-coffee: a novel method for fast and accurate multiple sequence alignment11Edited by J. Thornton“. In: *Journal of Molecular Biology* 302.1 (2000), S. 205–217. ISSN: 0022-2836. DOI: https://doi.org/10.1006/jmbi.2000.4042. URL: https://www.sciencedirect.com/science/article/pii/S0022283600940427.

[13] William R Pearson und David J Lipman. „Improved tools for biological sequence comparison.“ In: *Proceedings of the National Academy of Sciences* 85.8 (1988), S. 2444–2448.

[14] Lutz Prechelt, Guido Malpohl und Michael Philippsen. *JPlag: Finding plagiarisms among a set of programs*. Univ., Fak. für Informatik, 2000.

[15] Michael S Rosenberg. *Sequence alignment: methods, models, concepts, and strategies*. Univ of California Press, 2009.

[16] Timur Sağlam. „Mitigating Automated Obfuscation Attacks on Software Plagiarism Detection Systems“. Englisch. Diss. Karlsruher Institut für Technologie (KIT), 2025. 264 S. DOI: 10.5445/IR/1000179018/v2.

[17] Timur Sağlam u. a. „Token-based plagiarism detection for metamodels“. In: *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*. 2022, S. 138–141.

[18] Julie D Thompson, Desmond G Higgins und Toby J Gibson. „CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice“. In: *Nucleic acids research* 22.22 (1994), S. 4673–4680.

[19] Rendong Yang, Jamie L Van Etten und Scott M Dehm. „Indel detection from DNA and RNA sequencing data with transIndel“. In: *BMC genomics* 19 (2018), S. 1–11.