Institut für Visualisierung und Interaktive Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

# Vergleich von Augenbewegungen mit Multiplem Sequenzalignment und einer Consensus Matrix

Niklas Kleinhans

| | |
|---|---|
| **Studiengang:** | Informatik |
| **Prüfer/in:** | Prof. Dr. Daniel Weiskopf |
| **Betreuer/in:** | Dr. Michael Burch |
| **Beginn am:** | 15. Mai 2017 |
| **Beendet am:** | 15. November 2017 |

## Kurzfassung

Augenbewegungen können durch Koordinatenpunkte und deren Verbindung im Raum als eine Abfolge von Fixationspunkten modelliert werden. Die Analyse dieser raum-zeitlichen Daten ist eine Herausforderung. Weit entwickelte Systeme werden nicht nur präziser, sondern auch immer kompakter. Somit können künftig auch Daten über mobile Endgeräte gesammelt werden, wodurch das Spektrum der zu analysierenden Daten stetig erweitert wird. Diese Vielzahl an Daten gilt es zu verarbeiten. Insbesondere beim analysieren von Daten mehrerer Teilnehmer wird enorme Rechenleistung benötigt und fordert auch die Weiterentwicklung unterschiedlicher Analyseverfahren. Um diese Daten analysieren zu können, bilden Eye-Tracking Metriken eine Grundlage für die Reduzierung der Komplexität der Daten.

In dieser Arbeit soll ein Verfahren vorgestellt werden, welches es ermöglicht, die aufgezeichneten Daten von mehreren Personen zu vergleichen. Dabei werden die aufgezeichneten Daten durch Metrikenwerte zu Sequenzen transformiert und anschließend verglichen. Ähnlich wie bei DNA-Sequenzen in der Bioinformatik, wird hierbei ein multipler Sequenzalignment Algorithmus verwendet. Außerdem wird ein Webinterface entwickelt das es ermöglicht die Daten zu vergleichen und anschließend durch eine Consenus Matrix visualisiert.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Verzeichnis der Listings

# Verzeichnis der Algorithmen

# 1 Einleitung

In der Visualisierung wird das Interesse an der Auswertung von Augenbewegungen immer größer. Die Augenbewegungen werden aufgenommen, in einen raum-zeitlichen Kontext gebracht und anschließend weiterverarbeitet. Dieses Verfahren nennt sich Eye-Tracking. Die Systeme um Augenbewegungen aufzunehmen werden immer mobiler und günstiger. Das erweitert die Einsatzmöglichkeiten und trägt dazu bei den Sehprozess immer besser zu verstehen.

Diese Präsenz und Weiterentwicklung erfordert neue Analyseverfahren die zum einen, die Masse an Daten verarbeiten können und zum anderen die Daten auf mögliche Zusammenhänge analysieren und diese so aufbereiten, dass sie so informationsreich wie möglich visualisiert werden können. Bereits im 19. Jahrhundert wurde versucht, durch das Analysieren der Augenbewegungen den Leseprozess zu optimieren. Um die Daten messbar darzustellen wurden einige Metriken definiert. Viele der bekannten Verfahren stellen mit Hilfe dieser Metriken und unterschiedlichen Visualisierungen die Konzentration der Daten dar. Ein bekanntes Visualisierungsverfahren sind Heatmaps. Mit Heatmaps können stark konzentrierte Bereiche durch farbliche Markierung hervorgehoben werden. Beziehungen unter den Daten herzustellen fordert bei großen Datenmengen oft intensive Rechenleistungen. In der Bioinformatik ist es schon lange bekannt, dass durch das Vergleichen von Protein/DNA-Sequenzen evolutionäre Beziehungen zwischen Organismen hergestellt werden können. Es werden DNA-Sequenzen wie die folgende

CGC AAT GCG ATA TAC

mit Sequenzalignment Algorithmen verglichen und dabei übereinstimmende Muster oder Unterschiede erkannt. Sequenzen sind strukturierte Zeichenfolgen. Wenden wir eine der Metriken auf die Daten an, welche beispielsweise bei einer Studie mit mehreren Personen durch Eye-Tracking Systeme aufgenommen wurden und ordnen diese über die Zeit, erhalten wir ebenfalls Sequenzen, die analog mit einem Sequenzalignment Algorithmus verglichen werden können. Dabei ist das Ziel die Ähnlichkeiten und Übereinstimmungen in dem Sehprozess einzelner Personen herauszufinden. In dieser Arbeit wird ein Konzept entwickelt wie aus den Eye-Tracking Daten Sequenzen erzeugt werden können um diese anschließend mit einem multiplen Sequenzalignment zu vergleichen. Das Ergebnis wird mit Hilfe einer Consensus Matrix visualisiert.

## Gliederung

Die Arbeit ist in folgender Weise gegliedert:

**Kapitel 2 – Verwandte Arbeiten** Es werden verwandte Arbeiten besprochen und diese Arbeit differenziert.

**Kapitel 3 – Grundlagen** Aus der Bioinformatik und der Visualisierung werden die nötigen Grundlagen beschrieben.

**Kapitel 4 – Konzept** Es wird ein Konzept zum Aufbau und Vergleich von Sequenzen mit Eye-Tracking Daten vorgestellt.

**Kapitel 5 – Implementierung** In diesem Kapitel wird die Implementierung beschrieben.

**Kapitel 6 – Anwendung an einem Beispieldatensatz** Es wird ein Beispieldatensatz mit der Implementierung verglichen und die Ergebnisse diskutiert.

**Kapitel 7 – Zusammenfassung und Ausblick** Eine Zusammenfassung der Arbeit.

# 2 Verwandte Arbeiten

In diesem Kapitel werden verwandte Arbeiten besprochen und mit dieser Arbeit verglichen. Da in dieser Arbeit sowohl Komponenten aus dem Eye-Tracking, als auch Verfahren aus der Bioinformatik kombiniert werden, sind Ansätze aus beiden Bereichen interessant.

## 2.1 Aus der Bioinformatik

Zeichenketten werden in der Genanalyse schon länger miteinander verglichen. Schon früh wurde entdeckt, dass Sequenzen von zwei verwandten Genen als ähnlich beschrieben werden können, wenn ein Maximum an übereinstimmenden Mustern entlang der Sequenzen zu finden sind [Dur+98]. Für das Vergleichen von Sequenzen gibt es viele Ansätze, welche auch teilweise in dieser Arbeit besprochen werden.

### 2.1.1 Paarweise Vergleiche

In der Arbeit von Saul B. Needleman und Christian D. Wunsch wird der, nach ihnen benannte, **Needleman-Wunsch-Algorithmus** [NW70] vorgestellt. Dabei handelt es sich um einen globalen Vergleichsalgorithmus, der garantiert den optimalen Vergleich zweier Sequenzen findet. Es wird zu Beginn eine Distanzmatrix aufgebaut und anschließend ein Pfad von der ersten bis zur letzten Position mit dem optimalen Gesamtwert gesucht. Dieses Verfahren eignet sich sehr gut um zwei Sequenzen miteinander zu vergleichen. Der Algorithmus wird in dieser Arbeit verwendet. Weitere bekannte Verfahren um zwei Sequenzen miteinander zu vergleichen sind die Arbeiten von T.F.Smith und M.S.Waterman [SW81], Daniel S. Hirschberg [Hir75] und Gotoh Osamu [Got82].

### 2.1.2 Multiple Vergleiche

Um mehrere Zeichenketten miteinander zu vergleichen, wurde 1994 von Julie D.Thompson, Desmond G.Higgins und Toby J.Gibson ein Verfahren vorgestellt, mit dem mehrere Proteinsequenzen miteinander verglichen werden können. Der von ihnen entwickelte *ClustalW* Algorithmus benötigt dafür eine kubische Laufzeit, garantiert aber nicht das Auffinden eines optimalen Alignments. Mit geringen Laufzeiteinbußen aber einer optimierten Vergleichsqualität zum Auffinden eines Alignments, wurde im Jahr 2010 der *T-Coffee* Algorithmus[NHH00] von Cédric Notredame, Desmond G. Higgins und Jaap Heringa vorgestellt. Die Funktionsweisen der beiden Algorithmen werden in dieser Arbeit genauer beschrieben und angewendet.

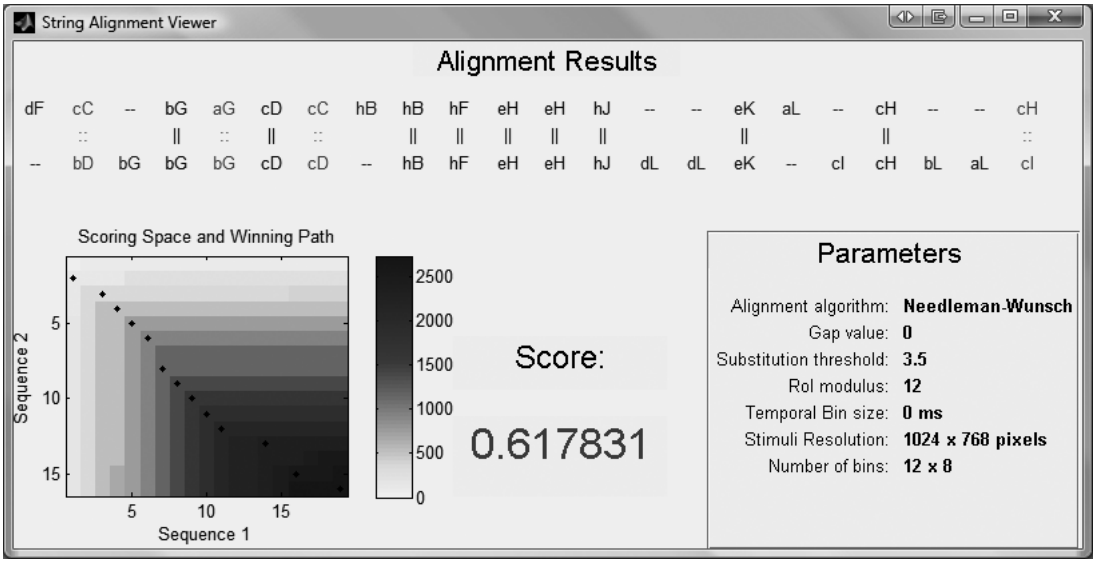


**Abbildung 2.1:** Benutzeroberfläche der *T-Coffee* Toolbox.

## 2.2 Aus dem Eye-Tracking

Die bisher besprochenen Arbeiten stammen aus der Bioinformatik und sind für den Vergleich von Proteinsequenzen konzipiert worden. In der Visualisierung und speziell in der Analyse von Eye-Tracking Daten wurden bereits unterschiedliche Verfahren veröffentlicht, um die Daten untereinander zu vergleichen.

### 2.2.1 Paarweise Vergleiche

In dieser Arbeit sollen aus den Eye-Tracking Daten Sequenzen erstellt und anschließend mit einem multiplen Sequenzalgorithmus verglichen werden. Einen paarweisen Ansatz verfolgen Filipe Cristino et al. in ihrem Tool *T-Coffee*[1], welches sie 2010 veröffentlichten [Cri+10]. In ihrer Methode werden die Daten aufbereitet, indem sie vorerst nach Sakkaden und Fixationen gefiltert werden. Zur Transformation der Daten in Sequenzen erstellen sie *Regions of Interests* ROIs und benennen diese alphabetisch. Diese Regionen können einfach generierte Raster sein, oder definierte Bereiche auf dem Stimulus (z.B. Mund, Augen, etc.). Um die Fixationsdauer mit einfließen zu lassen, haben sie eine Datenstruktur gewählt in der die Anzahl der Zeichen proportional zur Fixationsdauer gewählt wird. Bei einer Schrittweite von 50 Millisekunden und einer Fixation „A“ mit „150 Millisekunden“ Fixationsdauer würde dieser Bereich durch „AAA“ in der Sequenz repräsentiert werden. Da oftmals mehr Regionen benötigt werden, als das englische Alphabet Zeichen besitzt, werden die ROIs durch zwei Zeichen repräsentiert. Das erste Zeichen mit einem kleinen und das zweite Zeichen mit einem großen Buchstaben. Daraus resultiert eine Zielmengen der Größe von $26 \times 26$ Zeichen. Eine vereinfachte Rasterung könnte dann mit der x-Achse als erstes und der y-Achse als zweites Zeichen aufgebaut werden. Die aus der Datenaufbereitung resultierenden Sequenzen werden mit dem Needlemann-Wunsch Algorithmus verglichen. Dieser wird mit einer an die ROIs

[1] www.scanmatch.co.uk

angepassten Substiutionsmatrix und einer Lückenbestrafung initialisiert. *T-Coffee* ist eine frei verfügbare MATLAB Toolbox. In Abbildung 2.1 ist die Oberfläche der Applikation zu sehen. Diese ist in vier Bereiche aufgeteilt. Der obere Bereich zeigt eine Matrixrepräsentation der verglichenen Sequenzen. Die Lücken werden durch ein „-“repräsentiert. Weitere Sonderzeichen visualisieren den Vergleichsunterschied. In den weiteren Bereichen werden die Vergleichsparameter, der Vergleichswert (Alignment-Score) und eine Distanzmatrix mit eingezeichnetem Vergleichspfad angezeigt.

Die *T-Coffee* Toolbox beschränkt sich auf den Vergleich von zwei Sequenzen. In dieser Arbeit wird ein Konzept entwickelt, welches es ermöglicht Daten aus dem Eye-Tracking mit einem multiplen Sequenzalignment Algorithmus zu vergleichen. Es wird eine Applikation entwickelt, die Eye-Tracking Daten in strukturierte Sequenzen transformiert und diese vergleicht. Zusätzlich soll die Datenaufbereitung abhängig von der Wahl der Metrik weiter konfiguriert werden können.

### 2.2.2 Multiple Vergleiche

Adrian Madsen et al. haben in ihrer Arbeit „Using *T-Coffee* Scores to Understand Differences in Eye Movements Between Correct and Incorrect Solvers on Physics Problems“ die *T-Coffee* Methode dazu verwendet, um die Augenbewegungsdaten von 24 Personen in einer Studie zu vergleichen. Den Probanden, die teilnehmenden Personen einer Studie, bekamen die Aufgabe, physikalische Probleme zu beantworten und wurden anschließend nach richtigen und falschen Antworten klassifiziert. Anschließend wurden die Daten aus den jeweiligen Klassen permutiert und mit dem *T-Coffee* Verfahren verglichen. Wir werden in den folgenden Kapiteln sehen, dass wenn ein Vergleichswert, ein sogenannter Alignment-Score, aller Vergleiche gefordert wird und somit alle Sequenzen der Probanden miteinander verglichen werden müssen (Permutiert), dann ist eine sehr hohe Rechenleistung notwendig. Daher wird in dieser Arbeit ein multipler Sequenzalignment Algorithmus wie die *T-Coffee* Implementierung verwendet, um alle Daten miteinander zu vergleichen.

# 3 Grundlagen

Die zu vergleichenden Sequenzen werden durch die Berechnung von Metriken und deren Transformation auf Metrikenwerte aufgebaut. Diese werden mit einem Sequenzalignment Algorithmus verglichen und das Ergebnis durch eine Consensus Matrix visualisiert. In diesem Kapitel werden die für das Verfahren relevanten Metriken und Vergleichsalgorithmen besprochen.

## 3.1 Metriken

Technologisch fortgeschrittene Systeme bieten stetig mehr Möglichkeiten in unterschiedlichsten Anwendungsbereichen Augenbewegungen aufzunehmen. Es gilt nun die daraus resultierenden Datenbestände zu analysieren, visualisieren und auszuwerten. Um die Daten zu strukturieren und den Informationsgehalt messbar darzustellen, wurden Metriken definiert. Damit das menschliche Auge ein Objekt wahrnehmen kann, finden Augenbewegungen im Millisekundenbereich statt. Diese Bewegungen werden als *Gaze Points* bezeichnet. Wie in Abbildung 3.1 zu sehen, werden die *Gaze Points* zu Fixationen zusammengefasst, welche als Koordinatenpunkte abgespeichert werden. Die Koordinaten und der Zeitpunkt der Fixationen stellen informationshaltige Daten dar, die es gilt mit Hilfe von Metriken auszuwerten. Die Metriken aus Tabelle 3.1 listen einige Metriken bezogen auf die Fixationen auf. Um die Metriken zu erläutern, wird ein vereinfachtes Studienszenario angenommen. Dabei wird der Studienteilnehmer als Proband bezeichnet und als beobachtetes Objekt, der Stimulus, wird von einem zweidimensionales Bild ausgegangen.

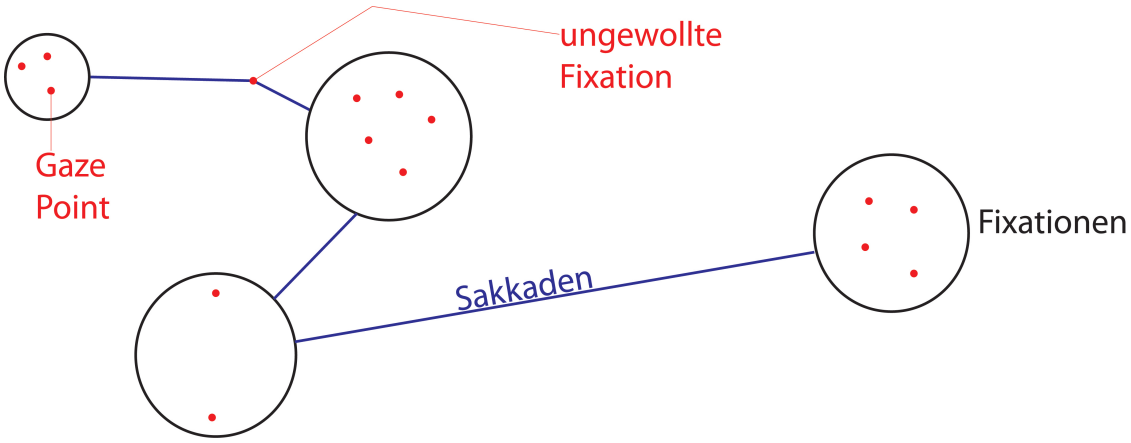


**Abbildung 3.1:** Zusammenhang zwischen „Gaze Points“, „Fixationen“ und „Sakkaden“.

Die Fixationen werden in unterschiedliche Klassen unterteilt. Man unterscheidet zwischen gewollten und ungewollten Fixationen. Diese werden lediglich in der Fixationsdauer, die Zeit in der keine weiteren Fixationen stattgefunden haben, unterschieden. Unter 240 Millisekunden gelten sie als ungewollte Fixationen. Dauer, Zeitpunkt, Koordinaten und Anzahl der Fixationen werden zur messbaren Darstellung aufgenommen und berechnet. Die Ansammlung mehrerer gewollter Fixationen in einem Bereich wird als Area of Interest (AOI) bezeichnet. Diese sind nicht zu verwechseln mit den ROIs aus Kapitel 2. Die ROIs sind frei definierte Regionen. Die AOIs sind Resultate aus den Aufnahmedaten. Die gesamte Abbildung 3.1 stellt einen AOI dar. Für die AOIs können analog zu den Fixationen auch die Dauer, Zeitpunkt, Koordinaten und Anzahl aufgenommen und berechnet werden. In der Analyse spielen AOIs eine große Rolle. Diese Interessenbereiche werden oft von den Probanden mehrfach fokussiert und in Beziehung gesetzt.

Die Verbindungen zwischen den Fixationen bezeichnet man als Sakkade. Tabelle 3.2 listet einige Metriken abhängig von den Sakkaden auf. In einer Sakkade finden keine weiteren Fixationen statt. Durch sie wird die Beziehung zwischen den Fixationen hergestellt und der Pfad der Augenbewegungen kann durch sie nachvollzogen werden. Mehrere Sakkaden bilden den Scanpath, den Pfad mit dem der Proband den Stimulus betrachtet hat. Tabelle 3.3 listet einige Metriken abhängig von den Sakkaden auf. Alle hier angesprochenen Metriken können durch Abbildungen in einen reellen Zahlenraum dargestellt werden. Abhängig von den Koordinaten und dem Zeitpunkt der Koordinatenaufnahme werden die Werte auf einen Funktionswert abgebildet. In dieser Arbeit werden diese Funktionswerte in einer Sequenz zusammengefügt und durch einen Sequenzalgorithmus miteinander verglichen. Die Funktionsweise eines solchen Algorithmus, der genaue Aufbau und Vergleich der Sequenzen werden in Kapitel 3.2 und 4 besprochen.

## 3.2 Sequenzalignment

Beim Sequenzalignment werden zwei oder mehrere Sequenzen miteinander verglichen, durch Einfügen von Lücken angepasst und die Übereinstimmung (der *Alignment-Score*) berechnet. Sequenzen sind einer Reihenfolge nach angeordnete Zeichenketten. Beim Vergleich dieser Sequenzen wird die Reihenfolge der Zeichen nicht verändert. DNA- oder RNA-Ketten sind bekannte Sequenzen aus der Bioinformatik, in denen übereinstimmende Muster mithilfe von Sequenzalignment Algorithmen gesucht werden. Beim Vergleichen von Protein/DNA-Sequenzen werden zwei Ziele verfolgt. Einerseits eine Schätzung der evolutionären Beziehungen zwischen Organismen und die Erkennung der funktionellen Standorte dieser Moleküle mit derselben biologischen Aktivität. Das Problem beim Vergleichen dieser Sequenzen ist, dass diese oft sehr lang sind und um evolutionäre Beziehungen herzustellen, werden oftmals mehr als nur zwei Zeichenketten zum Vergleichen benötigt. Die dafür benötigte Rechenleistung ist oft nicht vorhanden. Um diese Daten effizient zu vergleichen, wurden im Laufe der Zeit einige Ansätze für Vergleichsalgorithmen entworfen. Die für dieses Konzept relevanten Ansätze werden in diesem Kapitel besprochen.

**Tabelle 3.1:** Beschreibung der Metriken bezogen auf Fixationen [JK03].

| Metrik | Beschreibung | Einheit |
|---|---|---|
| Anzahl Fixationen | Anzahl an Fixationen. Eine hohe Anzahl an Fixationen lässt auf eine ineffizientere Suche schließen. | Anzahl |
| Anzahl ungewollter Fixationen | Fixationen unter 240 ms. | Anzahl |
| Anzahl gewollte Fixationen | Fixationen länger als 320 ms. | Anzahl |
| Anzahl Fixationen in den AOI | Anzahl aller Fixationen in einer AOI. Eine hohe Anzahl an Fixationen in einer AOI, lässt darauf schließen, dass dieser Bereich für den Probanden mehr von Bedeutung ist, als andere. | Anzahl |
| Fixationsdauer | Fixationsdauer eines Probanden. Eine längere Fixationsdauer lässt darauf schließen, dass dem Probanden das Extrahieren der Information an dieser Stelle schwerer gefallen ist. | ms |
| Fixationsdauer in den AOI | Summe der Zeiten aller Fixationen innerhalb einer AOI, in Millisekunden gemessen. | ms |
| Räumliche Verteilung von Fixationen | Die durch die Fixationen eingenommene Fläche. Eine geringere Fläche lässt auf eine effizientere Suche schließen. | Pixel x Pixel |
| Anzahl wiederholter Fixationen (post-target fixation) | Fixationen nachdem fixieren des Zielobjekts. Lässt auf ein nicht klar ersichtliches Zielobjekt schließen. | Anzahl |
| Zeit bis zur ersten Fixation | Dauer bis zur ersten Fixation. Je kürzer die Dauer, desto höher die aufmerksamkeitserregende Eigenschaft des Stimulus. | ms |
| Erste Fixationsdauer | Dauer der ersten Fixation. | ms |

**Tabelle 3.2:** Beschreibung der Metriken bezogen auf Sakkaden [JK03].

| Metrik | Beschreibung | Einheit |
|---|---|---|
| Sakkadenanzahl | Anzahl der Sakkaden. Eine hohe Anzahl an Sakkaden impliziert einen höheren Suchaufwand. | ms |
| Sakkaden-amplitude | Die Größe der Sakkade. Je größer eine Sakkade, desto breiter verteilt sind die Umgebungsreize eines Stimulus. | Winkelgrad |
| Sakkaden-geschwindigkeit | Die Geschwindigkeit der Sakkaden. | Winkelgrad |
| Sakkadendauer | Die Dauer einer Sakkade. | ms |
| Sakkadenrichtung | Die Richtung einer Sakkade | Grad |
| Anzahl der rückläufigen Sakkaden | Ändert sich die Richtung einer Sakkade um mehr als 90 Grad, dann wird diese als rückläufige Sakkade bezeichnet. Je größer die Anzahl, desto weniger Umgebungsreize sind vorhanden. | Anzahl |

**Tabelle 3.3:** Beschreibung der Metriken bezogen auf den Scanpath [JK03].

| Metrik | Beschreibung und Deutung | Einheit |
|---|---|---|
| Scanpathdauer | Die Dauer der Sakkaden und Fixationen. Je länger ein Scanpath dauert, desto ineffizienter ist die Suchstrategie. | ms |
| Scanpathlänge | Die Länge aller Sakkaden. Je länger der Scanpath, desto ineffizienter ist die Suchstrategie (evtl. auf Grund eines suboptimalen Layouts). | Pixel |
| Räumliche Dichte | Es wird ein Verhältnis zwischen der Anzahl von Fixationen und der Fläche des aufgespannten Pixelgitters über dem Scanpath erstellt. | Anzahl Fixationen / Pixel |
| Scanpathrichtung | Der Richtungsverlauf eines Scanpath. | |
| Sakkaden/ Fixations verhältnis | Das Verhältnis zwischen Fixationsdauer und Sakkadendauer. Es wird ersichtlich, ob der Proband mehr sucht, oder mehr an einer Stelle bleibt und sich konzentriert. | ms/ms |

### 3.2.1 Paarweises Alignment

Beim paarweisen Alignment werden zwei Zeichenfolgen miteinander verglichen. Dabei werden gleichartige oder ähnliche Positionen gegenübergestellt und Lücken an den Stellen eingefügt, an denen sich die Zeichenfolgen durch Substitution, Insertionen oder Deletionen unterscheiden [Dur+98]. Die Reihenfolge bleibt dabei weiterhin erhalten. Insertion, Substiution und Deletion stammen aus der Bioinformatik. Eine vereinfachte Darstellung ist in Abbildung 3.2 zu sehen. In (1) kann das optimale Alignment durch ersetzen des 3. Zeichens „G“ mit „C“ erreicht werden. In (2) kann das 5. Zeichen „A“ entfernt werden und bei (3) kann als 5. Zeichen ein A eingefügt werden. Diese eingefügten Lücken wirken sich auf die Übereinstimmung aus. Nach einem Vergleich wird ein Vergleichswert berechnet. Dieser wird als *Alignment-Score* bezeichnet.

| Substitution | Deletion | Insertion |
|---|---|---|
| CGGAAT | CGGAAT | CGCA - T |
| CGCAAT | CGCA - T | CGCA AT |
| (1) | (2) | (3) |

**Abbildung 3.2:** Darstellung von Substitution, Insertion und Deletion.

#### Bewertungstechniken

Ein Sequenzalignment Algorithmus fügt Lücken ein damit die gesamte Übereinstimmung der Zeichenketten optimiert wird. Um die Ähnlichkeit der Zeichenketten zu bestimmen, gibt es unterschiedliche Bewertungsverfahren. Dabei müssen die Zeichenübereinstimmungen belohnt und die Unterschiede bestraft werden. Das Einfügen verändert die Zeichenketten und muss daher ebenfalls bewertet werden. Um die Übereinstimmungen bzw. die Unterschiede zu bewerten, werden Substitutionsmatritzen (Scroningmatrix) verwendet. In der Bioinformatik gibt es beispielsweise die aus Abbildung 3.3 dargestellte *Blosum62* Matrix.

| | C | S | T | P | A | G | N | D | E | Q | H | R | K | M | I | L | V | F | Y | W | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | -1 | 1 | 0 | 2 | 1 | 1 | 2 | 1 | 2 | 0 | 0 | 2 | 4 | 1 | 5 | 1 | 2 | -2 | 5 | C |
| | | 2 | 0 | -2 | 0 | -1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | -1 | 1 | 1 | -1 | S |
| C | 9 | | 2 | -1 | -1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | -1 | 1 | 0 | 1 | 1 | 3 | T |
| S | -1 | 4 | | 2 | -2 | -1 | -1 | 0 | 0 | -1 | -1 | -1 | 1 | 1 | 0 | -1 | 0 | 0 | 2 | 1 | P |
| T | -1 | 1 | 5 | | 2 | -1 | -2 | -2 | -1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 2 | A |
| P | -3 | -1 | -1 | 7 | | 2 | 0 | -1 | -2 | 0 | 1 | 1 | 0 | 0 | -1 | 0 | -1 | 1 | 2 | 4 | G |
| A | 0 | 1 | 0 | -1 | 4 | | 3 | -1 | -1 | 0 | 0 | 1 | -1 | 0 | -1 | 0 | -1 | 0 | 0 | 0 | N |
| G | -3 | 0 | -2 | -2 | 0 | 6 | | 2 | -1 | -1 | -1 | 0 | -1 | 0 | 0 | 0 | 0 | 2 | 1 | 3 | D |
| N | -3 | 1 | 0 | -2 | -2 | 0 | 6 | | 1 | 0 | 0 | 2 | 2 | 1 | -1 | 0 | 0 | 2 | 2 | 4 | E |
| D | -3 | 0 | -1 | -1 | -2 | -1 | 1 | 6 | | 0 | -2 | 0 | 1 | 1 | -1 | 0 | 0 | 1 | 3 | 3 | Q |
| E | -4 | 0 | -1 | -1 | -1 | -2 | 0 | 2 | 5 | | 2 | -1 | 0 | 1 | 0 | -1 | 0 | 1 | 2 | 2 | H |
| Q | -3 | 0 | -1 | -1 | -1 | -2 | 0 | 0 | 2 | 5 | | -1 | -1 | 0 | -1 | 1 | 0 | 1 | 3 | -4 | R |
| H | -3 | -1 | -2 | -2 | -2 | -2 | 1 | -1 | 0 | 0 | 8 | | 1 | -2 | -1 | 1 | 1 | 2 | 3 | 1 | K |
| R | -3 | -1 | -1 | -2 | -1 | -2 | 0 | -2 | 0 | 1 | 0 | 5 | | -2 | -1 | -1 | 0 | 1 | 2 | 4 | M |
| K | -3 | 0 | -1 | -1 | -1 | -2 | 0 | -1 | 1 | 1 | -1 | 2 | 5 | | -1 | 1 | 0 | 0 | 1 | 3 | I |
| M | -1 | -1 | -1 | -2 | -1 | -3 | -2 | -3 | -2 | 0 | -2 | -1 | -1 | 5 | | -1 | 0 | -1 | 1 | 2 | L |
| I | -1 | -2 | -1 | -3 | -1 | -4 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | 1 | 4 | | 0 | 1 | 2 | 4 | V |
| L | -1 | -2 | -1 | -3 | -1 | -4 | -3 | -4 | -3 | -2 | -3 | -2 | -2 | 2 | 2 | 4 | | -1 | -2 | 1 | F |
| V | -1 | -2 | 0 | -2 | 0 | -3 | -3 | -3 | -2 | -2 | -3 | -3 | -2 | 1 | 3 | 1 | 4 | | -1 | 2 | Y |
| F | -2 | -2 | -2 | -4 | -2 | -3 | -3 | -3 | -3 | -1 | -3 | -3 | 0 | 0 | 0 | -1 | 6 | | | -1 | W |
| Y | -2 | -2 | -2 | -3 | -2 | -3 | -2 | -3 | -2 | -1 | 2 | -2 | -2 | -1 | -1 | -1 | -1 | 3 | 7 | | |
| W | -2 | -3 | -2 | -4 | -3 | -2 | -4 | -4 | -3 | -2 | -2 | -3 | -3 | -1 | -3 | -2 | -3 | 1 | 2 | 11 | |
| | C | S | T | P | A | G | N | D | E | Q | H | R | K | M | I | L | V | F | Y | W | |

**Abbildung 3.3:** Darstellung der *Blosum62* Matrix [HH92]

Diese weißt jedem möglichen Vergleichspaar einen bestimmten Wert zu und gewichtet die Proteine. Die Zeichenketten werden verglichen und basierend auf den Elementen $i, j$ der *Blosum62* Matrix $B_{i,j}$ werden die Werte übernommen und zu dem „Alignment-Score“ addiert. Der resultierende „Alignment-Score“ spielt in Kapitel 3.2.2 für den Vergleich von mehreren Zeichenketten eine weitere Rolle. Das Einfügen von Lücken wird meist mit „affinen *Gap-Penalty*“ bewertet. Dabei werden den Lücken nicht nur naiv negative Gewichtungen zugeordnet, sondern je nach Länge wird der Wert bestimmt. Dabei wird die Anzahl korrelierend mit der Länge der einzelnen Lücken berechnet. Beispielsweise ist eine fünf Zeichen lange Lücke besser zu bewerten, als fünf einzelne Lücken.

### Alignment Ansätze

Um zwei Zeichenketten miteinander zu vergleichen, gibt es verschiedene Ansätze. Je nach Art der Zeichenketten eignen sich unterschiedliche Verfahren.

**Globales Alignment** vergleicht alle Zeichen miteinander. Es wird auf jedes Zeichen eine Scoringfunktion angewendet und die Zeichen auf exakt gleiche Länge angepasst. Dieses Verfahren eignet sich gut, wenn sich die Sequenzen schon bei Eingabe stark ähneln. Ein Beispiel ist der Needleman-Wunsch Algorithmus [NW70]. Dieser löst das Optimierungsproblem, den optimalen *Alignment-Score* zu finden, für zwei Zeichenketten. Der Algorithmus erzeugt eine $(n+1)\times(m+1)$ große Matrix $M_{i,j}$ mit den Zeichenkettenlängen $n, m$. Matrix 3.1 zeigt die Berechnung von zwei Sequenzen $a, b$.

$$M_{i,j} = \begin{pmatrix} & - & a_1 & ... & a_n \\ - & 0 & -1 & ... & -n \\ b_1 & -1 & s(a_{[0,1]}, b_{[0,1]}) & s(a_{[0,i]}, b_{[0,1]}) & s(a_{[0,n]}, b_{[0,1]}) \\ ... & ... & s(a_{[0,1]}, b_{[0,j]}) & s(a_{[0,i]}, b_{[0,j]}) & s(a_{[0,n]}, b_{[0,j]}) \\ b_m & -m & s(a_{[0,1]}, b_{[0,m]}) & s(a_{[0,i]}, b_{[0,m]}) & s(a_{[0,n]}, b_{[0,m]}) \end{pmatrix} \tag{3.1}$$

Es werden rekursiv alle Präfixe der Zeichenketten miteinander verglichen. $s(a_{[0,i]}, b_{[0,j]})$ enthält den optimalen *Alignment-Score* der Zeichenkettenpräfixe zur Position $a_i, b_j$. Funktion $s()$ ist die Scoring Funktion, beispielsweise Sum-of-Pairs. In dem Sum-of-Pairs Verfahren werden die umliegenden Werte verrechnet, der Maximalwert verwendet und die *Gap-Penalty* dazu addiert. Auf diese Funktion wird in dieser Arbeit nicht weiter eingegangen werden. In Element $M_{n,m}$ der Matrix, steht der textitAlignment-Score des gesamten Zeichenkettenvergleichs. Der Algorithmus benötigt dabei eine Laufzeit von

$$O(max(n, m)^3). \tag{3.2}$$

Dabie beschreiben $n$, $m$ die Längen der Sequenzen.

**Free-Shift Alignment** arbeitet ähnlich zum globalen Alignment Verfahren. Bei dem Vergleich werden aber bei stark unterschiedlichen Sequenzen die eingefügten Lücken zu Beginn und am Ende der Zeichenkette ignoriert. Die negativen Wertungen der Lücken fließen somit nicht mit in den *Alignment-Score* ein.

**Lokales Alignment** verwendet Teilzeichenketten der zu vergleichenden Zeichenketten zum Berechnen des optimalen *Alignment-Scores*. Dabei müssen die optimalen Teilzeichenketten gefunden und verglichen werden. Ein hierfür bekannter Algorithmus ist der Smith and Waterman Algorithmus[SW81] und die Implementierung von Lalign [HM91].

### 3.2.2 Multiples Alignment

Die Laufzeit des Needleman-Wunsch Algorithmus ist von der kubischen Laufzeit auf eine quadratische Laufzeit optimierbar, indem man den Algorithmus auf konstante *Gap-Penalty* beschränkt. Paarweises Alignment ist somit mit einer Laufzeit von $O(max(n, m)^2)$ realisierbar. Dies macht es möglich, lange Zeichenketten effizient zu vergleichen. In der Praxis ist es aber notwendig, mehrere Zeichenketten miteinander effizient vergleichen zu können. In der Bioinformatik werden viele Sequenzen benötigt, um beispielsweise evolutionäre Gemeinsamkeiten oder Unterschiede festzustellen. Zudem sind DNA-Sequenzen oft sehr lang. Vergleicht man alle Sequenzen miteinander, erhält man eine Laufzeit von

$$O(2^k * max(n, ..., n_k)^k) \tag{3.3}$$

$k$ ist die Anzahl und $n$ die Länge der Sequenzen. Mit dieser Laufzeit stößt selbst leistungsstarke Hardware schnell an ihre Grenzen. Es werden daher zwei Ansätze zur Lösung des Laufzeitproblems betrachtet, wobei es noch weitere Ansätze gibt, auf die hier jedoch nicht eingegangen wird. Dazu gehört unter anderem die probabilistische Herangehensweise mit Hidden Markov Modellen. Es werden daher zwei Ansätze um das Laufzeitproblem zu lösen angesprochen. Dieser Ansatz wurde in der *Clustal Omega* Implementierung angewendet [Sie+11]. Die Implementierung ermöglicht es, um die 4000 Zeichenketten miteinander zu vergleichen. Diese Leistungssteigerung hat eine intensive Minderung der Genauigkeit zur Folge und sind zudem sehr komplex. In dieser Arbeit werden kleinere Datensätze angewendet, weshalb nicht weiter auf diese Methoden eingegangen wird.

#### Progressives Alignment

Um die Laufzeit zu verbessern, muss die Anzahl der Vergleiche reduziert werden. Im progressiven Alignment nimmt man hierzu einen Verlust der Genauigkeit in Kauf und verwendet Heuristiken um die zu vergleichenden Zeichenketten auszuwählen. In der Bioinformatik können phylogenetische Bäume aufgebaut werden. Da die von uns zu vergleichenden Zeichenketten keine evolutionären Abhängigkeiten besitzen, werden wir uns mit einem reinen Guide-Tree Ansatz beschäftigen. Einer der bekanntesten Ansätze ist der *ClustalW* Algorithmus.

**ClustalW** ist ein globaler progressiver Alignment Algorithmus der „affines *Gap-Penalty*“ anwendet. Er unterstützt als Substitutionsmatrix unter anderem die besprochene *Blosum62* Matrix und hat eine kubische Laufzeit. In Abbildung 3.4 ist der Ablauf des Algorithmus dargestellt. Die Sequenzen werden nach folgendem Vorgehen miteinander verglichen[Tho94]:

1. Die Zeichenketten werden paarweise verglichen. Beispielsweise mit dem Needleman-Wunsch Algorithmus. Das Ergebnis ist eine Distanzmatrix mit allen Alignment-Scrores.

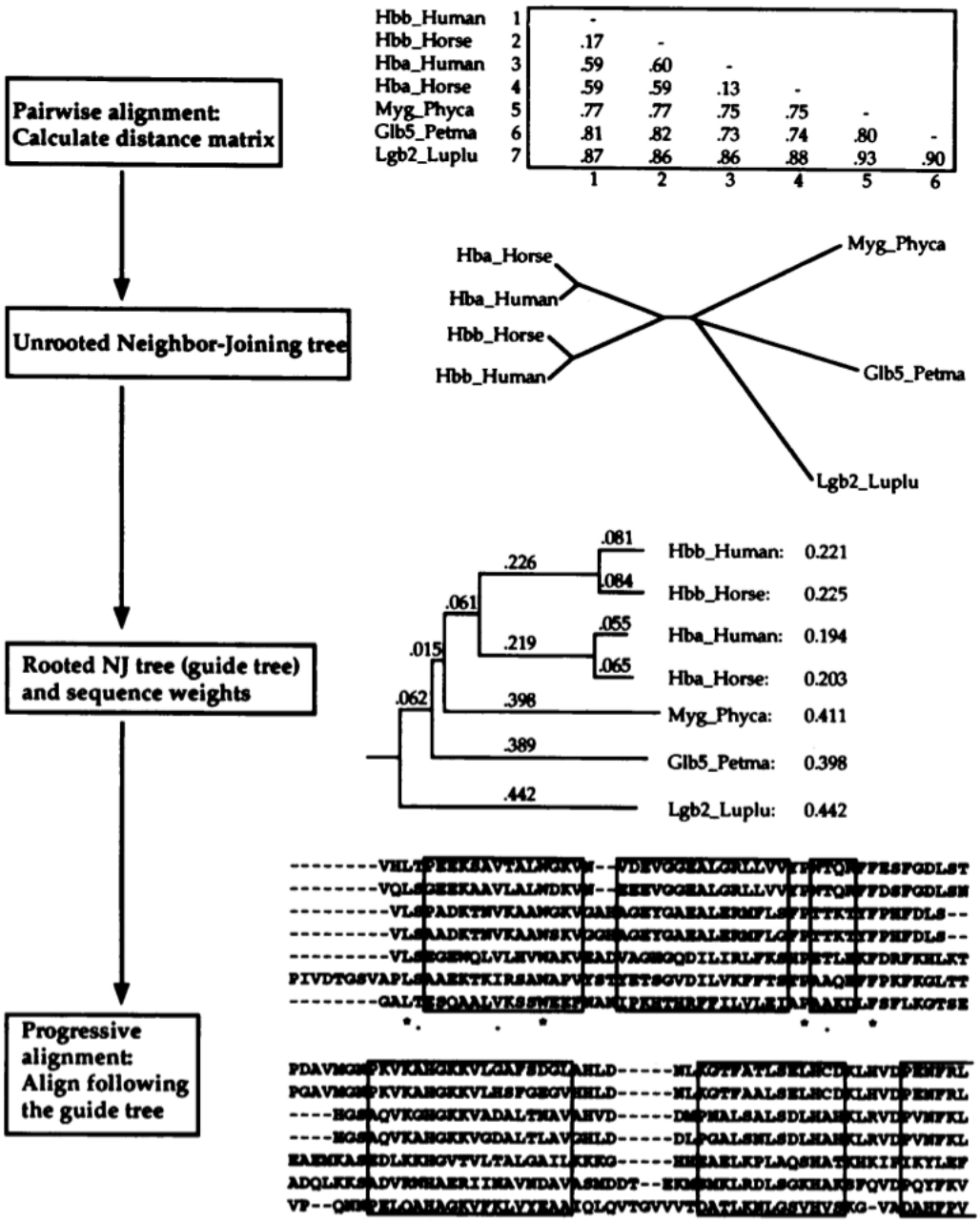


**Abbildung 3.4:** Progressives Alignment anhand des *ClustalW* Algorithmus [Tho94].

2. Mithilfe des Nighbour-Joining Algorithmus[SN87] wird basierend auf der Distanzmatrix ein wurzelloser Guide-Tree, mit Astlängen proportional zur Divergenz erstellt.
3. Auf dem initiierten Guide-Tree wird ein strukturierter Guide-Tree abgeleitet.
4. Entsprechend der Guide-Tree Struktur werden die Sequenzen anschließend paarweise verglichen.

Der paarweise Vergleich aus Abbilung 3.4 verläuft analog zu dem Algorithmus aus Kapitel 3.2.1 Der gesamte *Alignment-Score* kann beispielsweise durch die Formel 3.4 errechnet werden.

$$S(m) = G + \sum_i s(m_i) \tag{3.4}$$

$s(m_i)$ ist die Summe der Vergleichswerte jeder Spalte i und G ist die Summe aller *Gap-Penaltys*. Der Vorteil des *ClustalW* Algorithmus ist die kubische Laufzeit. Der progressive Ansatz fordert aber einen Verlust der Genauigkeit. Ein weiterer progressiver Ansatz ist die Implementierung von *T-Coffee*, welche mit ähnlicher Laufzeit einen geringeren Genauigkeitsverlust aufweist.

**T-Coffee** ist eine weiterentwicklung des progressiven Ansatz. Der progressive Ansatz ist ein „gieriger Algorithmus“, bei dem Fehler, die in der Anfangsphase der Ausrichtung gemacht wurden, später nicht korrigiert werden können. Um diesem Effekt entgegenzuwirken, wurde das Konsistenzprinzip entwickelt [Sie+11]. *T-Coffee* nutzt zum paarweisen Vergleichen der Sequenzen, einen globalen und lokalen Ansatz. Dabei werden die Sequenzen jeweils mit dem ClustlW und dem Lalign Algorithmus verglichen, wie in Abbildung 3.5 dargestellt. Bei Duplikaten werden die Werte summiert und zu einem Eintrag zusammengefügt, anderenfalls wird ein neuer Eintrag erstellt. Dieser Ansatz optimiert die Qualität der ersten Vergleiche und reduziert somit den Genauigkeitsverlust, der aus dem progressiven Ansatz resultiert. Um die Qualität noch weiter zu verbessern, wird zu jedem gewichteten Paar eine Konsistenzprüfung ausgeführt, indem alle möglichen weiteren Anordnungen verglichen werden. Dieses Problem wird auch als „maximum weight trace Problem"[Kec93] bezeichnet und wird in die Klassen der NP-Vollständigen Probleme eingeordnet. Es ist somit nichtdeterministisch in polynomieller Zeit lösbar. Die angegebene Laufzeit beträgt

$$O(N^2L^2) + O(N^3L) + O(N^3) + O(NL^3). \tag{3.5}$$

$N$ ist die Anzahl und $L$ die Länge der Zeichenketten. Daraus folgt eine kubische Laufzeit, welche es erlaubt eine hohe Anzahl von Zeichenketten miteinander zu vergleichen. Die Laufzeit setzt sich zusammen aus den Berechnungen für die paarweisen Vergleiche mit *ClustalW* und Lalign ($O(N^2L^2)$, des Guide Trees ($O(N^3L)$), der erweiterten Bibliothek ($O(N^3)$) und der Komposition des progressiven Alignments ($O(NL^3)$).

Mit diesem Ansatz sind, im Vergleich zu den progressiven Methoden, 5-10% genauere Vergleiche möglich. Wir werden die *T-Coffee* Implementierung verwenden, um die Zeichenketten aus den Eye-Tracking Daten zu vergleichen.

### Simultaneous Alignment

Beim simultananeous Alignment werden alle Zeichenketten gleichzeitig verglichen. Dies erfordert eine sehr hohe Rechenleistung und wird daher in dieser Arbeit nicht angewendet.

### Iterative Alignment

Der Unterschied zu progressiven Methoden ist, dass hierbei die anfänglich verglichenen Zeichenketten immer wieder angepasst werden. Dabei wird die Qualität der Vergleiche optimiert, aber die Laufzeit wiederum erhöht sich im Vergleich zum progressiven Ansatz. Zwei bekannte

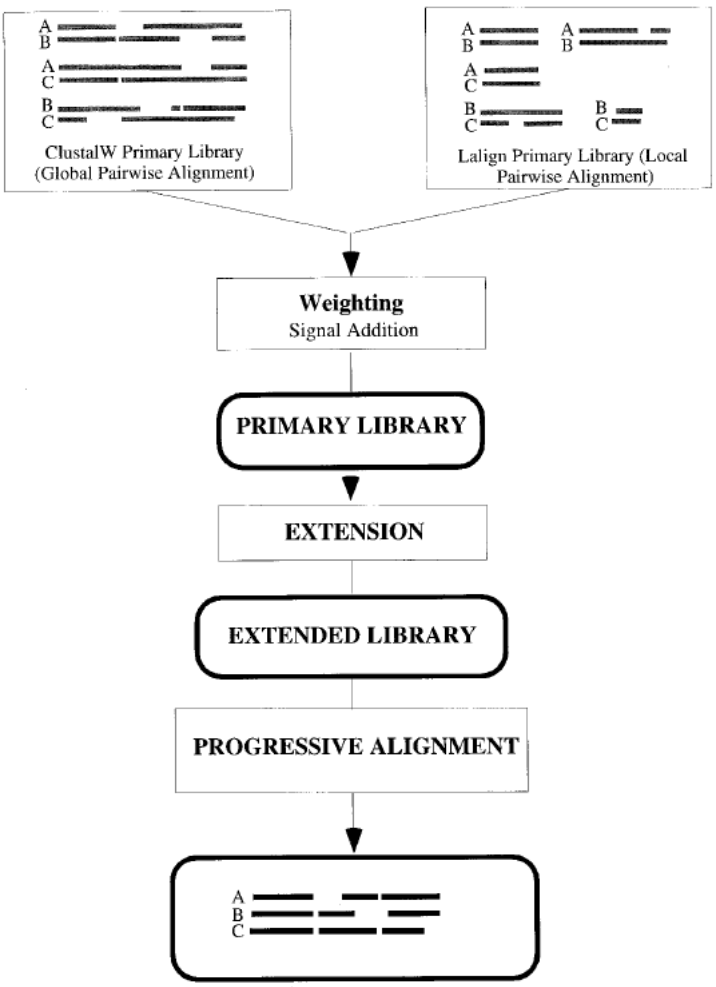


**Abbildung 3.5:** Progressives Alignment anhand des *T-Coffee* Algorithmus [NHH00].

Veröffentlichungen verwenden diesen Ansatz „A tool for multiple sequence alignment“ [LAK89] und „DCA: an efficient implementation of the divide-and-conquer approach to simultaneous multiple sequence alignment“ [SMD97].

## 3.3 Consensus Matrix

Der Begriff Consensus kommt aus dem Lateinischen und bedeutet übersetzt so viel wie „Übereinstimmung“. Eine Consensussequenz ist die Sequenz, die zu weiteren Mustersequenzen die höchste Übereinstimmung aufweist. Der Vergleich dieser Sequenzen kann durch einen mutliplen Sequenzalignment Algorithmus ausgeführt werden. Mit Hilfe einer Consenus Matrix können die verglichenen Sequenzen so dargestellt werden, dass Übereinstimmungen und Unterschiede sichtbar gemacht werden. Dabei sind die einzelnen Zeichenketten übereinander gelegt. Durch Einfärbung der unterschiedlichen Zeichen, werden die Übereinstimmungen durch Farbstreifen sichtbar. In Abbildung 3.6 ist eine Consensus Matrix zu sehen. Diese wurde mit dem *T-Coffee*

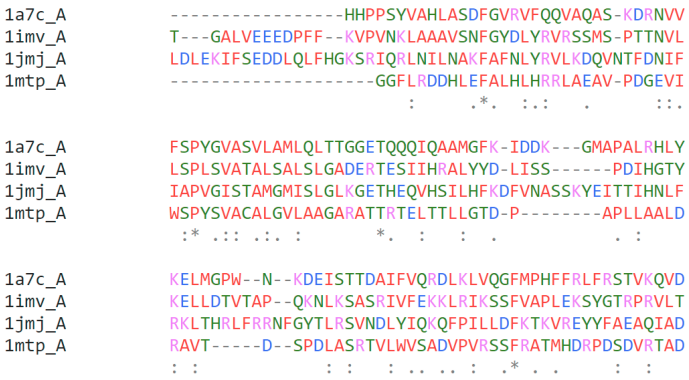


**Abbildung 3.6:** Consensus Matrix. Erstellt mit dem *T-Coffee* Algorithmus von EMBL-EBI.

Algorithmus von EMBL-EBI[1] erstellt. Diese Variante der Consensus Matrix wird oft bei dem Vergleichen von DNA- und RNA-Sequenzen verwendet. Zur Darstellung des Vergleichs werden die Lücken mit einem „-“ repräsentiert. Dadurch stehen alle Zeichen genau übereinander.

Durch die zusätzliche Einfärbung wird die Ähnlichkeit aller Spalten sichtbar. Nach 50 Zeichen werden die Zeichenketten unterbrochen und in einem neuen Block weiter aufgeführt. In der letzten Zeile markieren Sonderzeichen ( . : * ) stark übereinstimmende Spalten. Wobei * eine optimale Übereinstimmung beschreibt.

Eine erweiterte Darstellung von Consensus Matrizen wurde in der Arbeit „iHAT: interactive Hierarchical Aggregation Table for Genetic Association Data“ im Jahr 2011 vorgestellt [JH12]. In dieser Arbeit wurde das Tool „interaktive Hierarchische Aggregationstabelle (iHAT)“entwickelt, welches die Visualisierung von mehrfachen Sequenzanpassungen, zugehörigen Metadaten und hierarchischen Clusterings ermöglicht. Dabei werden die Zeichen auf Basis der erzeugten phylogenetischen Bäume angeordnet und eingefärbt. Die daraus resultierende Matrix hat einen noch höheren Informationsgehalt.

[1] https://www.ebi.ac.uk/Tools/msa/tcoffee/

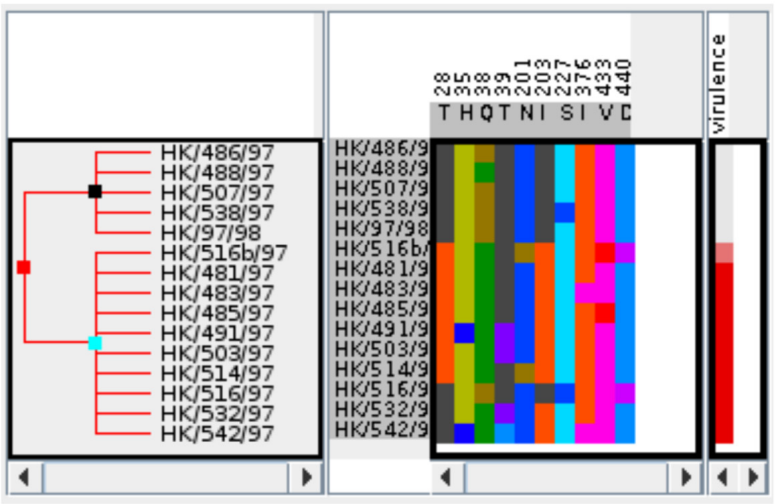


**Abbildung 3.7:** Matrixdarstellung des iHAT Tools [JH12].

# 4 Konzept

Die in Kapitel 3 besprochenen Grundlagen sollen in diesem Kapitel angewendet werden. Dabei wird der Aufbau der Sequenzen und die Anwendung eines multiplen Sequenzalignment Algorithmus beschrieben.

## 4.1 Aufbau der Zeichenketten

Die Zeichenketten setzen sich aus den Werten der Metriken zusammen. In der vorliegenden Arbeit sollen jedoch nicht alle Metriken angewendet werden, sondern vielmehr ein Konzept entwickelt werden, das die Zusammensetzung der Zeichenketten beschreibt, um weitere Metriken zu realisieren und vergleichen zu können. Dabei wird der Zeichenkettenaufbau anhand des Beispieldatensatzes aus Tabelle 4.3 und der Abbildung 4.1 wird der Zeichenkettenaufbau beispielhaft erklärt. Der Aufbau der Zeichenketten wird in Datenaufbereitung und Granularität unterteilt.

### 4.1.1 Datenaufbereitung

Um die Daten verwenden zu können, müssen diese angeordnet werden. Da die Reihenfolge der Zeichenketten durch den Algorithmus nicht verändert wird, spielt diese Anordnung eine große Rolle. Der Algorithmus versucht einen optimalen *Alignment-Score* zu erreichen, es sollen also möglichst viele Zeichen übereinstimmen und wenig Lücken eingefügt werden. Daher ist es eher unwahrscheinlich, dass die ersten und letzten Zeichen miteinander verglichen werden. Die Anordnung legt somit den Grundstein für die Qualität des Vergleichs. In dieser Arbeit werden zwei Ansätze besprochen. **Zeitabhängige** und **koordinatenabhängige** Anordnungen.

Bei der zeitabhängigen Anordnung wird die Chronologie der Daten nicht verändert. Anhand der Zeitstempel werden die Datensätze sortiert und in eine Zeichenkette umgerechnet. Eine koordinatenabhängige Anordnung hat zur Folge, dass die Chronologie verloren geht. Bei einem zweidimensionalen Stimulus können die Werte den x- oder y-Koordinaten auf- oder absteigend sortiert werden. Die Wahl der Anordnung ist abhängig von der Wahl der Metrik und dem gewollten Informationsgewinn. Tabelle 4.3 zeigt einen Beispieldatensatz der über die Zeit angeordnet ist. Aus ihm ergibt sich ein Scanpath indem die Koordinatenpunkte (Fixationen) chronologisch wie in Abbildung 4.1 verbunden werden. Die Fixationsdauer zu jeder Fixation wurde mit aufgenommen. Diese ist auch eine in Kapitel 3.1 besprochene Metrik. Naiv können die zu vergleichenden Zeichenketten wie in Tabelle 4.1 aufgebaut werden. Dabei werden die Zeiten in Millisekunden der zeitlichen Folge nach in die Zeichenkette eingefügt. Anhand dieser Anordnung werden zu bestimmten Zeitpunkten die Fixationsdauern verglichen. Das bedeutet, wenn innerhalb eines bestimmten Zeitraums die Fixationspunkte gleich lange fixiert wurden, werden diese als Übereinstimmung erkannt. Wie in der Tabelle 4.1 zu sehen ist, existieren in diesem Fall nahezu keine

**Tabelle 4.1:** Zeichenkette mit Werten der Fixationsdauer.

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| p1 | 250 | 150 | 283 | 433 | 183 | 333 | 300 | 416 | 183 | 250 | 183 | 550 | 150 |
| p2 | 266 | 266 | 216 | 600 | 433 | 150 | 316 | 167 | 333 | 949 | 150 | 167 | 167 |
| p3 | 283 | 300 | 283 | 167 | 133 | 233 | 183 | 233 | 416 | 533 | 216 | 233 | 233 |

**Tabelle 4.2:** Zeichenkette nach der Rasterung.

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| p1 | 7 | 4 | 4 | 8 | 5 | 5 | 5 | 8 | 5 | 5 | 5 | 5 | 5 |
| p2 | 7 | 4 | 7 | 4 | 4 | 7 | 8 | 5 | 8 | 8 | 8 | 5 | 5 |
| p3 | 4 | 7 | 4 | 4 | 4 | 7 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |

Übereinstimmungen. Dies hat zwei Gründe. Zum einen ist die Anordnung der Daten in diesem Fall möglicherweise nicht sinnvoll und zum andern liegt der Wertebereich der Zeichen sehr weit auseinander. Daher ist die Wahrscheinlichkeit einer Übereinstimmung sehr gering. Existiert bei der Wahl der Anordnung kein Zusammenhang der Daten, wird bei diesem Vergleichsverfahren kein verwertbares Ergebnis zu erwarten sein. Die Eingabe der Daten muss dementsprechend aufbereitet werden. Um die Anordnung zu wählen muss man sich anhand der Metrik den möglichen Informationsgewinn überlegen. In dem Beispiel aus Tabelle 4.1 wird die Fixationsdauer verglichen. Eine lange Fixationsdauer lässt meist auf einen Bereich schließen, auf den sich der Proband besonders konzentriert hat. Daraus folgt ein Zusammenhang zwischen dem Bereich und der Fixationsdauer. Eine möglicherweise bessere Anordnung wäre in diesem Fall die Anordnung über die x-Koordinaten. Die Werte würden dann anhand ihrer Position und der Fixationsdauer verglichen. Das bedeutet, wenn innerhalb eines bestimmten Bereichs die Fixationspunkte gleich lang fixiert wurden, werden diese als Übereinstimmung erkannt. Viele lange Fixationen in einem Bereich werden zu einem AOI zusammengefasst. Mit diesem Vergleich können somit Personenübergreifende AOI definiert werden. Um die Wahrscheinlichkeit der Übereinstimmung noch weiter zu optimieren, kann der Wertebereich noch auf eine Genauigkeit angepasst werden. Die Umsetzung wird in Kapitel 4.1.2 besprochen.

Ein weiteres Beispiel für die Anordnung über x-Koordinaten ist die Anzahl gewollter Fixationen. Dabei handelt es sich vorerst um einen totalen Wert. Durch zusätzliche Rasterung des Stimulus wie in Abbildung 4.1 und einer Indizierung der Raster von 1-9 (von oben nach unten rechts) können die Sequenzen aus Tabelle 4.2 erstellt werden. Für jede Fixation mit den Koordinatenpunkten $x, y$ wird der Positionsindex in die Sequenz eingefügt. Sammeln sich in einem Raster viele Fixationen an, spiegelt sich das in der Sequenz als eine Folge gleicher Indizes wieder. Heatmaps beispielsweise würden diese Ansammlung durch Einfärbung hervorheben. Der Unterschied zu den Heatmaps ist bei dieser Analyse, dass zusätzlich der Vergleich zu allen Probanden dargestellt werden kann. In Tabelle 4.2 farblich markiert sind alle Koordinatenpunkte, welche sich probandenübergreifend im selben Raster befinden.

### 4.1.2 Granularität

Um ein hohen *Alignment-Score* zu erreichen, müssen möglichst viele Zeichen übereinstimmen. Dabei ist der Wertebereich der Zielmenge einer Metrik entscheidend. Je größer der Wertebereich, desto geringer ist die Wahrscheinlichkeit einer Übereinstimmung. Um den Wertebereich zu

**Tabelle 4.3:** Ein Beispieldatensatz mit normierten Koordinaten.

| Ursprungsdaten | | | | | Zieldaten | |
|---|---|---|---|---|---|---|
| Zeitstempel | Fixationsdauer | Fixations-punkt X | Fixations-punkt Y | Proband | accuracy | Index |
| 2586 | 250 | 1 | 1 | p1 | 300 | 2 |
| 2836 | 150 | 5 | 3 | p1 | 200 | 1 |
| 2986 | 283 | 9 | 1 | p1 | 300 | 2 |
| 3269 | 433 | 7 | 5 | p1 | 400 | 3 |
| 3702 | 183 | 11 | 7 | p1 | 200 | 1 |
| 3885 | 333 | 7 | 9 | p1 | 300 | 2 |
| 4218 | 300 | 5 | 6 | p1 | 300 | 2 |
| 4518 | 516 | 2 | 5 | p1 | 500 | 4 |
| 5035 | 183 | 2 | 10 | p1 | 200 | 1 |
| 5218 | 250 | 10 | 10 | p1 | 300 | 2 |
| 5468 | 183 | 10 | 9 | p1 | 200 | 1 |
| 5651 | 550 | 11 | 9 | p1 | 600 | 5 |
| 6200 | 150 | 11 | 10 | p1 | 200 | 1 |
| 326117 | 266 | 1 | 2 | p2 | 300 | 2 |
| 326383 | 266 | 3 | 1 | p2 | 300 | 2 |
| 326650 | 216 | 9 | 1 | p2 | 200 | 1 |
| 326866 | 600 | 11 | 1 | p2 | 600 | 5 |
| 327466 | 433 | 11 | 3 | p2 | 400 | 3 |
| 327899 | 150 | 7 | 3 | p2 | 200 | 1 |
| 328049 | 316 | 11 | 9 | p2 | 300 | 2 |
| 328365 | 167 | 11 | 11 | p2 | 200 | 1 |
| 328532 | 333 | 7 | 9 | p2 | 300 | 2 |
| 328865 | 949 | 3 | 7 | p2 | 900 | 6 |
| 329814 | 150 | 1 | 6 | p2 | 200 | 1 |
| 329964 | 167 | 3 | 6 | p2 | 200 | 1 |
| 330131 | 167 | 3 | 3 | p2 | 200 | 1 |
| 559960 | 283 | 7 | 7 | p3 | 300 | 2 |
| 560243 | 300 | 7 | 5 | p3 | 300 | 2 |
| 560543 | 283 | 5 | 5 | p3 | 300 | 2 |
| 560826 | 167 | 5 | 6 | p3 | 200 | 1 |
| 560993 | 133 | 6 | 7 | p3 | 100 | 0 |
| 561126 | 233 | 3 | 7 | p3 | 200 | 1 |
| 561359 | 183 | 3 | 7 | p3 | 200 | 1 |
| 561542 | 233 | 3 | 6 | p3 | 200 | 1 |
| 561776 | 416 | 1 | 5 | p3 | 400 | 3 |
| 562192 | 533 | 1 | 3 | p3 | 500 | 4 |
| 562725 | 216 | 3 | 3 | p3 | 200 | 1 |
| 562941 | 233 | 11 | 10 | p3 | 200 | 1 |
| 563175 | 233 | 10 | 11 | p3 | 200 | 1 |

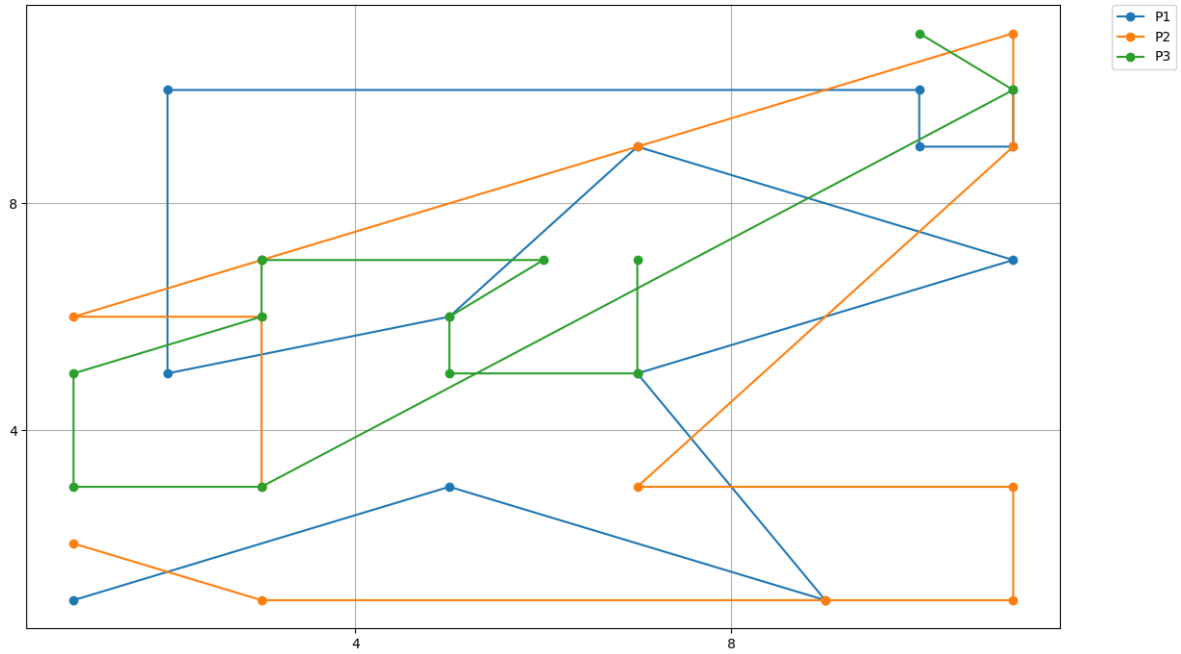


**Abbildung 4.1:** Plot der Koordinatenpunkte aus Tabelle 4.3.

reduzieren, kann die Granularität der Metrikenwerte angepasst und der Wertebereich durch einen Maximalwert begrenzt werden. Dieser Vorgang wird in drei Schritte unterteilt, wie in Abbildung 4.2 zu sehen ist. Bei vielen Metriken ist es nicht notwendig, millisekundengenaue Vergleiche durchzuführen. Daher können die Werte auf eine Genauigkeit angepasst werden.

Im ersten Schritt werden die Werte auf einen bestimmten Wert durch Faktorisierung und anschließendes Runden angepasst. In Tabelle 4.3 sind in den Zeilen der Zieldaten die Werte, welche durch $accuracy()$ auf $\frac{1}{10}$ Sekunden umgerechnet wurden, aufgelistet. Die Funktion $accuracy()$ welche die Faktorisierung und die Rundung durchführt ist in Algorithmus 4.1 beschrieben.

Im zweiten Schritt werden die Werte indiziert. Dabei werden die Ergebnisse aus $accuracy()$ aufsteigend sortiert und nach unterschiedlichen Wertevorkommen indiziert. In Tabelle 4.3 sind die berechneten Indizes aufgelistet. Aus dieser Indizierung resultiert kein Informationsverlust, da der Grad der Ungleichheit eines Wertevergleichs nicht gemessen wird. Dies wäre aber durch die Konzipierung einer Matrix für Eye-Tracking Daten analog zur *Blosum62* Matrix in der Bioinformatik denkbar. Durch die Indizierung erhält man eine möglichst gleich verteilte Zielmenge und schränkt damit den Wertebereich schon in diesem Schritt ein. Die Notwendigkeit wird auch in dem folgendem Kapitel nochmals genauer beschrieben.

Im dritten. Schritt wird der Wertebereich angepasst. Dazu wird der Maximalwert der Zielmenge angegeben. In Abbildung 4.2 ist der Maximalwert der Zielmenge $k$. Die Schrittweite von Eins ist fest definiert. Die Zielmenge erhalten wir durch *calculateTargetScaler* wie im Pseudocode 4.1 dargestellt. Nach Durchführung der drei Schritte erhalten wir eine möglichst gleich verteilte reduzierte Zielmenge. Die Umrechnung von den Ursprungsdaten auf die Zielmenge ist eine surjektive Abbildung. Da diese aber nicht injektiv und somit auch nicht bijektiv ist, können die Werte nicht durch eine Umkehrfunktion zurück gerechnet werden. Dies stellt aber keine

**Algorithmus 4.1** Berechnung der Zielmenge.

**procedure** ACCURACY($value$,$scaleValue$)
  **return** $round(value * scaleValue)$
**end procedure**

**procedure** CALCULATETARGETSCALER($maxOrigin$,$maxTarget$)
  $scalingValue = 1$
  **if** $maxOrigin > maxTarget$ **then**
    $scalingValue = \frac{maxTarget}{maxOrigin}$
  **end if**
  **return** $scalingValue$
**end procedure**

**procedure** TARGETSCALE($value$,$maxTargetValue$)
  $mvl = maxValueOfList(Sequence)$
  $targetScalerValue = calculateTargetScaler(mvl, maxTargetValue)$
  **return** $round(value * targetScalerValue)$
**end procedure**

**Tabelle 4.4:** Zeichenkette der Fixationsdauer von Person 1, Person 2 und Person 3, auf eine Genauigkeit von $\frac{1}{10}$ Sekunden indiziert.

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| p1 | 2 | 1 | 2 | 3 | 1 | 2 | 2 | 4 | 1 | 2 | 1 | 5 | 1 |
| p2 | 2 | 2 | 1 | 5 | 3 | 1 | 2 | 1 | 2 | 6 | 1 | 1 | 1 |
| p3 | 2 | 2 | 2 | 1 | 0 | 1 | 1 | 1 | 3 | 4 | 1 | 1 | 1 |

Einschränkung dar, da die Reihenfolgen der Zeichenketten nicht verändert wurden und somit die Ursprungswerte erhalten bleiben. Wird der Ablauf auf den Beispieldatensatz mit einem $scaleValue = 100$ und einem $maxTargetValue = 20$ angewendet, erhalten wir die Werte aus Tabelle 4.3 in Spalte Index. Die zu vergleichenden Zeichenketten aus Tabelle 4.4 können anschließend mit einem multiplen Sequenzalignment Algorithmus verglichen werden.

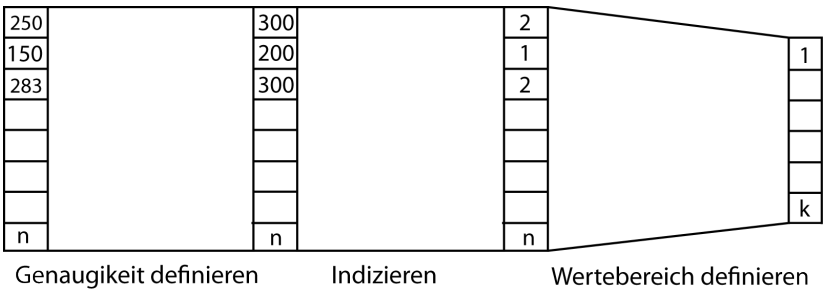


**Abbildung 4.2:** Anpassung der Genauigkeit und des Wertebereichs.

**Tabelle 4.5:** Simple Substitutionsmatrix.

| | $b_1$ | ... | $b_m$ |
|---|---|---|---|
| $a_1$ | 5 | -4 | -4 |
| ... | -4 | 5 | -4 |
| $a_n$ | -4 | -4 | 5 |

## 4.2 Konfiguration des Algorithmus

In Kapitel 3.2 wurden mehrere Ansätze von multiplen Sequenzalignment Algorithmen besprochen. In dieser Arbeit wird die *T-Coffee* Implementierung von der *Seqan* Bibliothek [Dör+08] verwendet. Der *T-Coffee* Algorithmus hat eine kubische Laufzeit und verwendet sowohl globales als auch lokales Alignment. Dabei nutzt er die globale *ClustalW* und die lokale Lalign Implementierung.

Da es für das Vergleichen von Eye-Tracking Daten bisher noch keine Substitutionsmatrix gibt, wird eine simple Matrix wie die in Tabelle 4.5 verwendet. Diese Matrix bestraft alle nicht übereinstimmenden Werte mit $-4$ und belohnt die Übereinstimmungen mit $5$. Zusätzlich werden neue Lücken mit $-13$ und die Erweiterung von Lücken mit $-1$ bestraft. Durch diese Werte erhält man eine Affine *Gap-Penalty*. In Kapitel 5 wird die *Seqan* Bibliothek und die Verwendung noch weiter beschrieben.

## 4.3 Visualisierung

Zur Darstellung der Vergleiche, wird eine Consensus Matrix verwendet. Nach dem Vergleichen sind alle Sequenzen gleich lang. Somit erhält man automatisch eine $n \times j$ Matrix, indem alle Sequenzen $n$ mit Länge $j$ in die Matrix eingetragen werden. Lücken werden mit einem Sonderzeichen „-“ dargestellt. Um die übereinstimmenden Zeichen hervorzuheben, werden diese farblich hervorgehoben, wie in Tabelle 4.2. Einfärbung mit weiteren Farben ist in diesem Konzept nicht vorgesehen, da die Wertebereiche möglicherweise zu groß sind um jedem Wert eine Farbe zuzuweisen. Das menschliche Auge könnte die Farbunterschiede nicht mehr wahrnehmen. In dem Tool *iHAT* [JH12] werden sogar die Metadaten in der Matrix mit visualisiert. Ein ähnliches Konzept wäre auch mit den hier verwendeten Daten denkbar. Beispielsweise könnte bei der Rasterung die Anzahl der entfernten Raster gewichtet und in der Consensus Matrix entsprechend unterschiedlich eingefärbt werden. Auch der Guide-Tree könnte analog zu *iHAT* mit in die Visualisierung eingebaut werden.

## 4.4 Einschränkungen

Diese Arbeit stellt ein erstes Konzept zum Vergleichen von Eye-Tracking Daten mit Hilfe eines multiplen Sequenzalignment Algorithmus zur Verfügung. Auf Grund des zeitlichen Rahmens beinhaltet diese Arbeit einige Einschränkungen:

- Der Grad der Unterscheidung fließt nicht in den *Alignment-Score* mit ein, da keine Substitutionsmatrix für den Vergleich konzipiert wurde. Die Methoden aus dem Vergleich von DNA-Sequenzen wurden übernommen.
- Der Algorithmus wurde nicht selbst implementiert. Die Implementierung von *Seqan* wurde zum Vergleichen der Daten verwendet. Diese stellt einige Stringformate für die Zeichenketten bereit. In dieser Arbeit wurde ein *String<char>* gewählt. Durch diese Datenstruktur sind keine mehrstelligen Werte möglich. Daher wurde der Indizierungsschritt und das Einschränken des Wertebereichs in der konzipierten Prozesskette aus Kapitel 4.1.2 eingeführt. Jeder Indizierte Wert kann durch ein selbst definiertes Alphabet repräsentiert und verglichen werden. Diese Repräsentation ist bijektiv und somit eindeutig. In Kapitel 5 wird dieses Verhalten näher beschrieben. *T-Coffee* verwendet ein zweistelliges Alphabet. Dies ermöglicht einen deutlich größeren Wertebereich.

# 5 Implementierung

Teil dieser Arbeit ist die Implementierung des entworfenen Konzepts zum Vergleichen von Eye-Tracking Daten mit einem multiplen Sequenzalignment Algorithmus. Der Implementierung werden Eye-Tracking Daten zur Verfügung gestellt. Diese sollen eingelesen, verglichen und anschließend visualisiert werden. Die nötigen Vergleichsparameter sollen in einer grafischen Oberfläche einstellbar sein und die Visualisierung findet über eine Consensus Matrix statt. In diesem Kapitel wird die Umsetzung beschrieben und die Funktionsweise anhand der Oberfläche vorgestellt.

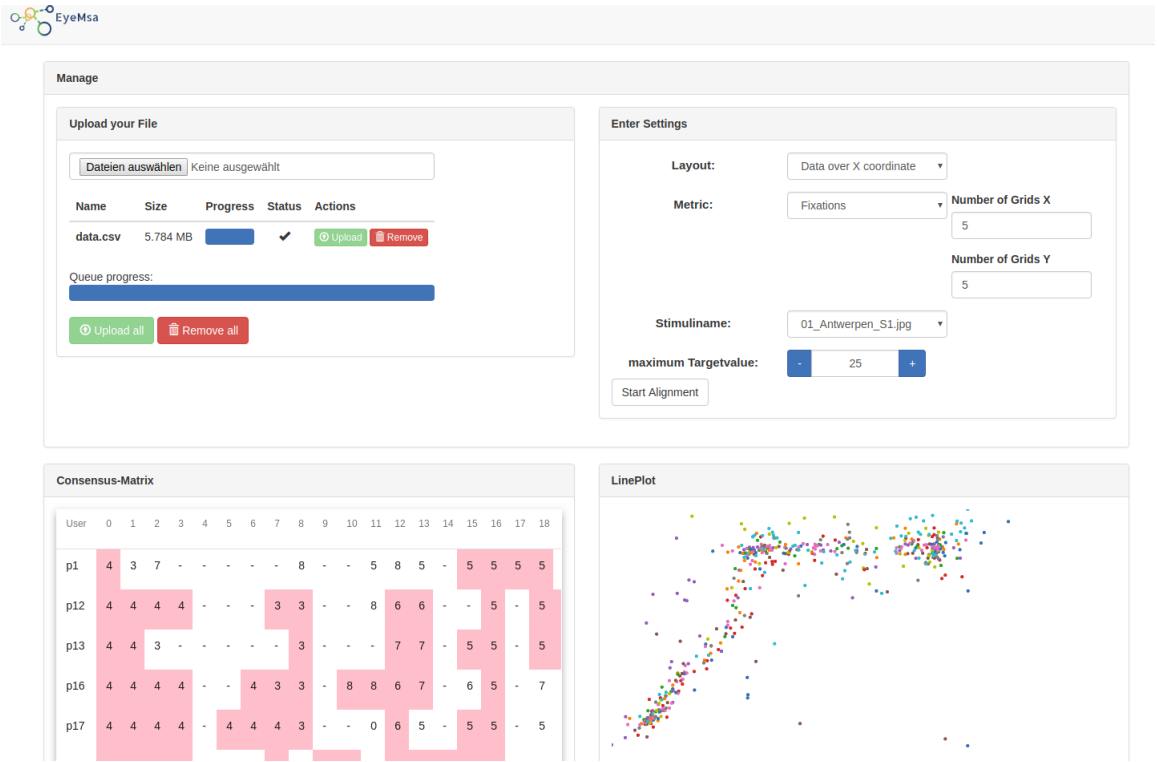


**Abbildung 5.1:** Oberfläche der EyeMsa Applikation.

**Tabelle 5.1:** *csv* Format für den Datenimport der Webapplikation.

| Kopfzeile | Datentyp | Beschreibung |
|---|---|---|
| Timestamp | int | Ein eindeutiger Zeitstempel. Über diesen können die Daten chronologisch sortiert werden |
| StimuliName | string | Der Name des Stimulis. Die Daten werden neben dem Namen auch nach dem Stimuliname gefiltert |
| FixationIndex | int | Ein Indizierung der Fixationspunkte eines Benutzers |
| FixationDuration | int | Die Dauer einer Fixation in ms |
| MappedFixationPointX | int | Koordinatenpunkt X der Fixation |
| MappedFixationPointY | int | Koordinatenpunkt Y der Fixation |
| user | string | Der Benutzername |
| description | string | Eine zusätzliche Beschreibung |

## 5.1 Fachliche Umsetzung

Die fachliche Umsetzung beschreibt die Funktionsweise der einzelnen Komponenten. In Abbildung 5.1 ist die Oberfläche der Applikation zu sehen. Diese stellt ein Webinterface bereit und wird im folgenden als „EyeMsa“bezeichnet. Sie ist in drei Hauptkomponenten eingeteilt. Das Hochladen der Daten (Abbildung 5.1 *Upload your File*), in dem die Daten auf den Server geladen werden. In die Konfiguration des Datenvergleichs (Abbildung 5.1 *Enter Settings*) und der anschließenden Visualisierung (Abbildung 5.1 *Consenus-Matrix*, *LinePlot*). Auf EyeMsa kann mit einem gängigen Browser über http://-IP-/Address-:3000 zugegriffen werden. Man wird direkt auf das „Dashboard“weitergeleitet. Beim Starten werden alle Bereiche bis auf den Bereich zum Hochladen einer Datei ausgeblendet. Alle weiteren Funktionen werden schrittweise angezeigt.

### 5.1.1 Hochladen der Daten

Zum Hochladen der Daten wird ein Filecontainer bereitgestellt. Die Daten können über das Webinterface in dem Bereich „Upload your File“ Abbildung 5.1 ausgewählt und an den Webserver gesendet werden. Die Daten müssen dabei in einer *csv* Datei mit der Struktur aus Tabelle 5.1 vorliegen.

### 5.1.2 Konfiguration

Nach dem Hochladen erscheint die Funktion „Enter Settings“. In dieser werden die Konfigurationen für die Datenaufbereitung vorgenommen. Dabei können die folgenden Parameter eingestellt werden:

| | | | |
|---|---|---|---|
| 0 | 3 | 6 | 9 |
| 1 | 4 | 7 | 10 |
| 2 | 5 | 8 | 11 |

**Abbildung 5.2:** Eine $3 \times 4$ Rasterung mit indizierten Feldern. Diese Indizes werden zum Aufbau der Sequenzen verwendet.

**Layout** Das Layout definiert die Anordnung der Daten. Die Daten können analog zum Konzept angeordnet werden. Dabei besteht die Auswahl von der Anordnung über die Zeit(Data over time), die x-Koordinaten (Data over X coordinate) und über die y-Koordinaten (Data over Y Koordinate).

**Metric** Die Zeichenketten werden anhand von Metriken erzeugt. In EyeMsa können die Metriken Fixationsdauer (Fixationduration) und Fixationen(Fixations)ausgewählt werden.

Um die Fixationswerte darzustellen, wird der Stimulus in Raster aufgeteilt. Dabei erscheinen bei der Auswahl von „Fixation“ zwei weitere Eingabefelder (Number of Grids X, Number of Grids Y) (Abbildung 5.1 (Enter Settings)). Es wird die Anzahl der Raster entlang der X (Number of Grids X) und Y-Achse (Number of Grids Y) eingegeben. Standardmäßig ist eine Rasterung von $5 \times 5$ vorkonfiguriert. Diese kann beliebig angepasst werden. Die Raster werden nach dem Schema in Abbildung 5.2 erstellt. Sie zeigt eine $4 \times 3$ Rasterung. Die Koordinatenpunkte werden in der Datenaufbereitung durch einen Rasterindex repräsentiert.

**Stimuliname** Beim Hochladen der Daten wird die *csv* Datei erstmals eingelesen um die zu Verfügung stehenden Stimulis aufzulisten. Diese können über „Stimuliname “ausgewählt werden. Initial wird der erste Wert übernommen.

**Maximum Targetvalue** Das Zielalphabet welches für den Vergleich zur Verfügung steht, kann durch eine maximale Anzahl beschränkt werden. Ein Wertebereich von 1-62 steht zur Verfügung. Die maximale Grenze von 62 wird in dem Kapitel 5.2 erläutert. Die Berechnung der Zielwerte ist in Algorithmus 4.1 *TargetScale* beschrieben. *maxTargetValue* ist der hier zu konfigurierende Wert.

**Scalevalue** Wird als Metrik „Fixationduration“gewählt, wird die Konfiguration der Rasterung nicht mehr angezeigt und es erscheint ein weiterer Wert „Scalevalue“, wie in Abbildung 5.3 zu sehen. Dieser Wert skaliert die Werte der Fixationsdauer auf eine definierte Genauigkeit. Die Berechnung ist in Algorithmus 4.1 *Accuracy* beschrieben. Anschließend wird die Konfiguration bestätigt.

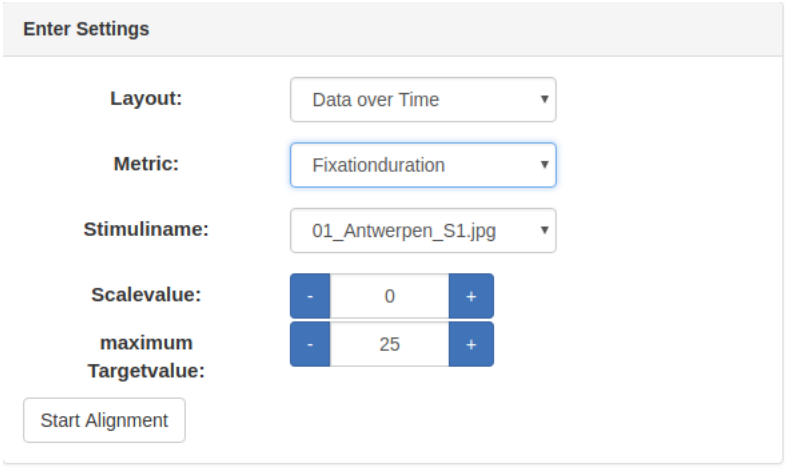


**Abbildung 5.3:** Konfigurationsbereich zum Einstellen der Vergleichsparameter.

### 5.1.3 Visualisierung

Durch die Bestätigung der Konfigurationsparameter über den Button „Start Alignment “ wird der Vergleich ausgeführt. Anschließend erscheinen die Felder zur Visualisierung. Es stehen 2 Bereiche zur Verfügung. Wie in Abbildung 5.1 (Consensus-Matrix) und (LinePlot) zu sehen ist, können die verglichenen Daten über eine Consensus Matrix und einem zusätzlichen Plot angezeigt werden. In der Matrix werden alle Sequenzen übereinander gestellt und Übereinstimmungen farblich hervorgehoben. Jede Zeile gehört zu einem Probanden. Der zusätzliche Plot zeigt alle Fixationen an. Dabei werden die Fixationspunkte entsprechend der Probanden eingefärbt. Nach der Konfiguration beginnt der Vergleich. Anhand der Daten und der Konfiguration werden die Daten zur Weiterverarbeitung in ein JSON Format umgewandelt und mithilfe der *Seqan* MSA Implementierung verglichen. Als Resultat erhält das Webinterface ein bearbeitetes JSON Format und visualisiert das Ergebnis durch eine Consensus Matrix.

## 5.2 Technische Umsetzung

Die Abbildung 5.4 zeigt eine technische Systemübersicht der Implementierung von EyeMsa. Anhand dieser wird die technische Umsetzung der einzelnen Komponenten erklärt.

### 5.2.1 Bibliotheken und Frameworks

Für die Implementierung der Applikation wurden Bibliotheken und Frameworks verwendet. Diese sind in Tabelle 5.2 aufgelistet und beschrieben.

[1] https://opensource.org/licenses
[2] https://raw.githubusercontent.com/nodejs/node/master/LICENSE

**Tabelle 5.2:** Verwendete Bibliotheken.

| Name | Lizenz | Beschreibung | Verwendung |
|---|---|---|---|
| *Seqan* [Dör+08] | BSD 2-Clause License[1] | *Seqan* ist ein Softwarebibliothek, die sich explizit auf die Entwicklung von hochperformanter Sequenzanalyse-Software konzentriert, indem sie eine Sammlung der gängigen algorithmischen Komponenten und Datenstrukturen bereitstellt [GDR09]. | Die *Seqan* Bibliothek wird für den Vergleich von mehreren Zeichenketten verwendet. Diese stellt eine implementierte Version des *T-Coffee* Algorithmus bereit. |
| D3[D3] | BSD 3-Clause License[1] | D3 ist eine JavaScript-Bibliothek zur Manipulation von Dokumenten auf Datenbasis | D3 wird für einen zusätzliche Plot darstellen im SVG Format verwendet. |
| nodejs | LICENSE[2] | Nodejs ist eine Serverseitige Javascript Plattform | Stellt den Webserver bereit |
| Angular | MIT License[1] | Angular ist eine Javascript Framework zum entwickeln von Webapplikation. Es unterstützt die Client seitige Entwicklung | Wird für die Clientseitige Webintrrface darstellung verwendet |
| ngx- Datatable | MIT License[1] | ngx-Datatable ist eine Component zum darstellen von großen Datenmengen. | Wird verwendet um die Consensus Matrix darzustellen |
| angular2- busy | MIT License[1] | Angular2-buys ist ein Angular Component um einen Ladevorgang darzustellen. | Wird verwendet um die Blockierung der Anwendung anzuzeigen |
| ng2- file- upload | MIT License[1] | ng2-file-upload stellt einen Filecontainer bereit und organisiert die Datenzuordnung | Wird verwendet um die Daten auf dem Server zu speichern |
| cython | Apache License[1] | Cython ist eine Programmiersprache, welche eine kommunikation zwischen C, C++ und python ermöglicht | Stellt die Verbindung zwischen python und c++ her. |
| Pandas | BSD 3-Clause License[1] | Pandas ist eine Softwarebibliothek für Python um Daen aufzubereiten und organisieren | Pandas wird verwendet um die csv Daten aufzubereiten |
| Numbpy | BSD 3-Clause License[1] | Numbpy ist eine Softwarebibiothek für Python die unterschiedliche Funktionen bereitstellt und den Umgang mit Vektoren, Matritzen und Arrays vereinfacht | Numbpy wird für die Datenumrechnung verwendet. |

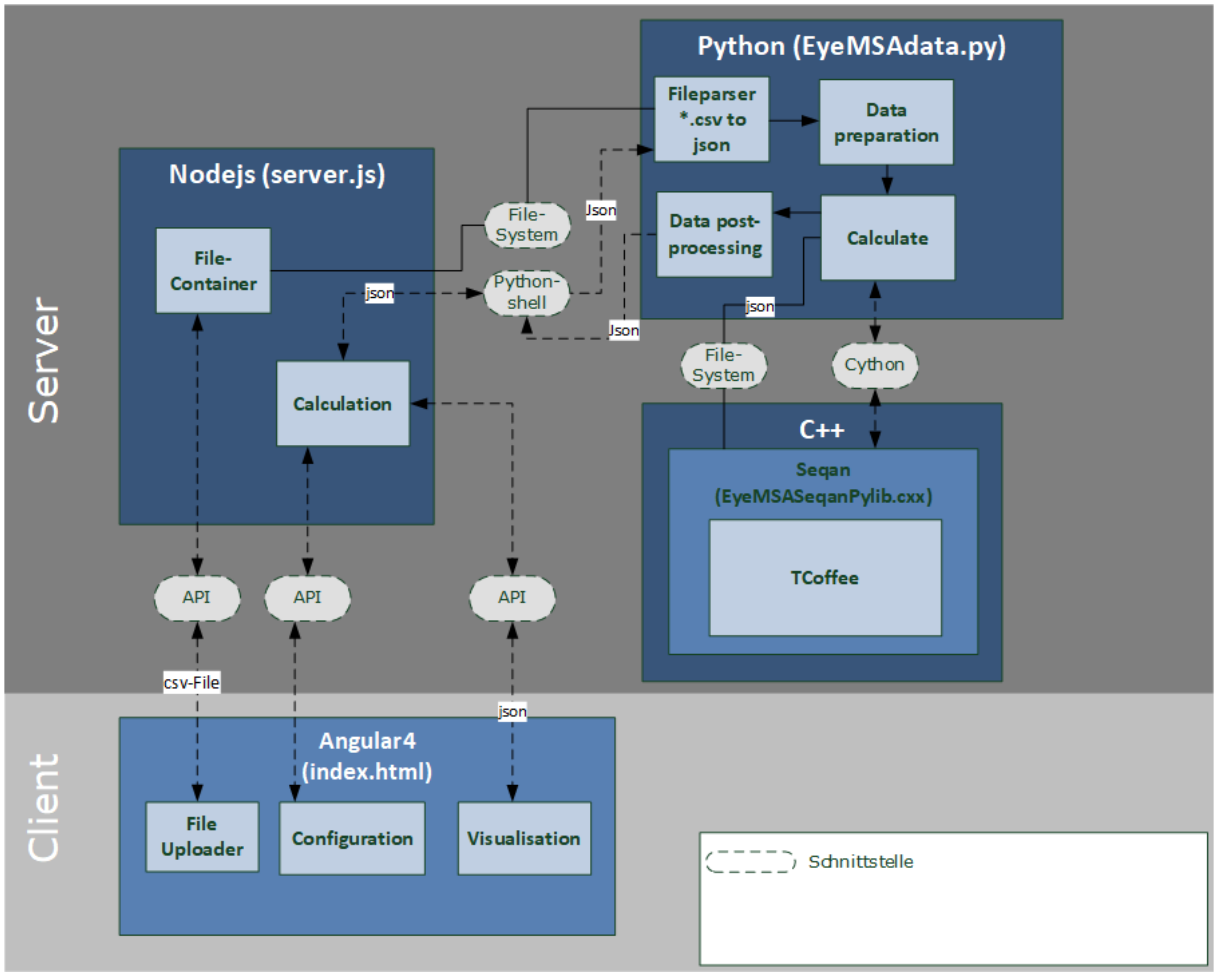


**Abbildung 5.4:** Systemübersicht der Webapplikation.

Das Webinterface wurde mit Angular realisiert und stellt die clientseitige Funktionalität zur Verfügung, wie in Tabelle 5.2. Serverseitig läuft ein Nodejs Webserver. Client und Server kommunizieren über eine RESTFUL API Schnittstelle.

### 5.2.2 Hochladen der Daten

Um die Daten über die API Schnittstelle vom Client an den Server zu senden, wurde das ngx-file-upload Modul verwendet. Dieses Modul stellt einen Filecontainer auf dem Server bereit. Über das Filesystem des Servers können die Daten anschließend weiterverarbeitet werden. Die Daten werden nach erfolgreicher Verarbeitung entfernt. Es wird keine Datenverwaltung bereitgestellt.

### 5.2.3 Konfiguration.

Die Einstellungen zum Vergleichen der Daten werden über das Webinterface übergeben und durch ein JSON File an den Server übermittelt. Listing 5.1 zeigt den Aufbau. Durch das bestätigen

**Listing 5.1** Konfiguration

```
{
      "configuration": {
            "scaleValue": float,
            "maxTargetValue": int,
            "layout": string,
            "metric": string,
            "grid": {"enabled": boolean, "x": int, "y": int}
            "filename": string
      }
}
```

**Listing 5.2** JSON Struktur der *csv* Daten.

```
{
      "eyedata": [{
            "person": string,
            "data":[{
                  "fixationduration": int,
                  "stimuliname": string,
                  "timestamp": int,
                  "y": int,
                  "x": int,
                  "id": 0
                  }],
            "rawSequence": [int],
            "alignedSequence": [int]
      }]
}
```

der Konfiguration wird eine POST-Anfrage $/api/calculateData$ gesendet und die JSON Struktur wird übergeben. Der Webserver bearbeitet die Anfrage und übergibt die Konfiguration an den Python Fileparser aus Abbildung 5.4. Der Fileparser fügt dem JSON aus der Konfiguration die Informationen aus dem *csv* File, wie in Listing 5.2, hinzu.

Die Id wird zur eindeutigen Identifizierung vom Fileparser zusätzlich vergeben.

### 5.2.4 Datenaufbereitung

Nach der Konfiguration werden die Daten in „Data Preparation“ aus Abbildung 5.4 anhand der Konfiguration aufbereitet und die zu vergleichenden Sequenzen mithilfe der Pandas und Numbpy Bibliotheken (Tabelle 5.2) erstellt. In Listing 5.3 sind einige Codeauszüge zur Datenaufbereitung. Nach der Initialisierung aus dem Dataparser werden die Randwerte berechnet. Darunter sind die maximal und minimal Werte der Sequenzen und Rasterkoordinaten. Je nach Metrik werden die nötigen „DataPreparation“ Methoden aus Listing 5.4 aufgerufen.

**Listing 5.3** Pseudocode der Hauptklasse zur Datenaufbereitung.

```
#//////////////////////////////////////////#
#                     Main                 #
#//////////////////////////////////////////#

#...

#=============================#
#     Initialising the Data   #
#=============================#
dP = dataPreparation()
dP.loadConfiguration()
dP.loadEyeData()
dP.setUsers()

#=============================#
#      Values for Metrics     #
#=============================#
# min and max values
dP.extremas();
# if Grid enabled calculate Grid
if(dP.data['configuration']['grid']['enabled']):
   dP.generateGrids()

#=============================#
#           Metrics           #
#=============================#
if dP.data['configuration']['metric']=='fixations':
   dP.fixation()
if dP.data['configuration']['metric']=='fixationduration':
   dP.fixationDuration()

#=============================#
#           Align             #
#=============================#
# begin Alignment
dP.align()

#=============================#
# postprocessing and writeback #
#=============================#
dP.postprocessing()
dP.setDataAfterAlignmentAndWriteback()
```

**Listing 5.4** Pseudocode zur Datenaufbereitung „DataPraparation“ Klasse.

```
#//////////////////////////////////////////#
#             dataPreparation              #
#//////////////////////////////////////////#
#...
def fixation():
   for user in users:
      sortByLayout()
      getGridIDs()
   retrun sequencesToAlign()
def fixationDuration():
   for user in users:
      sortByLayout()
      accuracy()
      IndexingValues()
   return sequencesToAlign
def align():
   for values in sequenceToAlign
      targetScale()
   SeqanEyeDataAlign(sequenceToAlign)
   return alignedSequences
def postprocessing():
   for user in users:
      for values in alignedSequences
         getValueOfIndex(value)
   return alignedSequences
#....
```

**Fixation()** sortiert alle Daten (*sortByLayout()*) und rechnet alle Koordinaten in GridIDs um (*getGridIDs*) und gibt anschließend die Sequenzen zurück.

**FixationDuration()** sortiert die Daten nach den Vorgaben der Konfiguration und skaliert die Daten auf eine Genauigkeit (*accuracy*). Nach der Skalierung werden die Werte indiziert (*IndexingValue*). Anschließend werden die Daten in **align()** auf den Wertebereich skaliert und mit Cython, wie in Abbildung 5.4 zu sehen, an die *Seqan* Implementierung übergeben. Dort werden sie verglichen.

**Listing 5.5** Pseudocode zur Implementierung des *T-Coffee* Algorithmus.

```
// EyeMSA alphabet Size 62
char myCharacters[] = { 'a', 'b', 'c', 'd', 'e', 'f',
                        'g', 'h', 'i', 'j', 'k', 'l',
                        'm', 'n', 'o', 'p', 'q', 'r',
                        's', 't', 'u', 'v', 'w', 'x',
                        'y', 'z', 'A', 'B', 'C', 'D',
                        'E', 'F', 'G', 'H', 'I', 'J',
                        'K', 'L', 'M', 'N', 'O', 'P',
                        'Q', 'R', 'S', 'T', 'U', 'V',
                        'W', 'X', 'Y', 'Z', '1', '2',
                        '3', '4', '5', '6', '7', '8',
                        '9', '0'
                      };

// returns Char of myCharacters
char convertIntToChar(int value){...}

// returns value in myCharacters
int convertChartToInt(char value){...}

// MSA initialisation
_initMsaParams()
{
   MsaOptions<TAlphabet, TScore> msaOpt;
   // Use global and lokal alignment
   appendValue(msaOpt.method, 0); // lokal Align
   appendValue(msaOpt.method, 1); // global Align
   msaOpt.pairwiseAlignmentMethod = 1; // unbanded Pairwise
   msaOpt.sc = scMat;         // Simple ScoreMatrix
   msaOpt.sc.data_gap_open= -13; // New Gap
   msaOpt.sc.data_gap_extend=-1; // Extend exising Gap
   msaOpt.sc.data_match=5;    // Match value
   msaOpt.sc.data_mismatch=-4; // Mismatch value
   msaOpt.build = 0;          // GuideTree Nighbor-joining
}

// Cython connection
std::vector< std::vector<int> > EyeMSA::EyeData::getResult(std::vector< std::vector<int> >
     _data)
{
  convertIntToChar()
  // run MSA
  convertChartToInt()
  retrun result;
}
```

**Listing 5.6** JSON Struktur des Vergleichs.

```
{
"configuration": {
"scaleValue": float,
"maxTargetValue": int,
"layout": string,
"metric": string,
"grid": {"enabled": boolean, "x": int, "y": int}
"filename": string
},
"eyedata": [{
"person": string,
"data":[{
"fixationduration": int,
"stimuliname": string,
"timestamp": int,
"y": int,
"x": int,
"id": 0
}],
"rawSequence": [int],
"alignedSequence": [int]
}
```

### 5.2.5 Vergleich

Die *Seqan* Implementierung erhält einen zweidimensionalen Vector mit den Sequenzen. Diese werden durch *convertIntToChar()* aus Listing 5.5 in die Alphabetwerte umgerechnet. Das hier beschränkte Alphabet ist der Grund für die Beschränkung von „Maximum Targetvalue“ aus Kapitel 5.1.2. In *_initMsaParams()* werden die *T-Coffee* Parameter konfiguriert. Analog zum T-Coffe Algorithmus werden die Sequenzen paarweise lokal *msaOpt.method=0* und global *msaOpt.method=1* verglichen. Da keine Matrix erstellt wurde um die Ähnlichkeit der Werte zu bestimmen, wurde eine simple Matrix definiert. Diese belohnt Übereinstimmungen mit $5$ *msaOpt.sc.data_match=5* und bestraft unterschiedliche Werte mit $4$ *msaOpt.sc:mismatch=-4.* Neue Lücken werden mit $-13$ *msaOpt.sc.data_gap_open* und die Erweiterung von Lücken wird mit $-1$ *msaOpt.sc.data_gap_extend=-1* bestraft. Nach dem Vergleich werden die Sequenzen über Cython wie in Abbildung 5.4 beschrieben als zweidimensionaler Vektor übergeben. „Data postprocessing“ fügt die verglichenen Sequenzen der JSON Struktur $[alignedSequences]$ hinzu. Die JSON Struktur aus Listing 5.6 enthält anschließend alle nötigen Informationen und wird über den Callback zuerst an den Webserver und anschließend an den Client übergeben. Die Daten werden in einem Angular-service abgelegt und ist somit für alle Komponenten im Client verwendbar.

### 5.2.6 Visualisierung

Die Ergebnisse des Vergleichs werden durch eine Consensus Matrix dargestellt. Diese wird mithilfe des *ngx-Datatable* Modul aus Tabelle 5.2 umgesetzt. Alle übereinander stehenden Werte, welche übereinstimmen, werden markiert. Zusätzlich wurde ein Vergleichsalgorithmus implementiert, der alle benachbarten Felder vergleicht und gleiche Werte einer Gruppe hinzufügt. Die Sequenten sind in einem Zweidimensionalem Array abgespeichet. Mit $i$ Sequenzen und $j$ Zeichen ergibt sich die darstellbare Matrix aus Abbildung 5.5.

| | | |
|---|---|---|
| i-1,j-1 | i-1,j | i+1,i+1 |
| i,j-1 | i,j | i,j+1 |
| i+1,j-1 | i+1,j | i+1,j+1 |

**Abbildung 5.5:** Matrixrepräsentation.

Somit muss der Vergleichsalgorithmus die Zeichen horizontal und vertikal vergleichen und bei Übereinstimmung entweder einer bestehenden Gruppe hinzufügen oder einer neuen zuordnen. Die Gruppierungen erweitern die Funktionalität der Funktion. Beispielsweise können dann anschließend alle Werte einer Gruppe in einem Plot dargestellt werden. Die Gruppen werden ebenso benutzt um die Werte in der Tabellendarstellung einzufärben. Ist ein Wert in einer Gruppe enthalten, wird dieser Markiert.

**Algorithmus 5.1** Pseudocode zum Erzeugen der Übereinstimmungsgruppen.

```
procedure EQUALGROUP(value,scaleValue)
    for values in alignedSequences do
        if value is equal to([i, ii − 1], [i, ii + 1], [i − 1, ii], [i + 1, ii]) then
            if equalValue Not in a equalGroup then
                set values into new equalGroup
            else add value into equalGroup of equalValue
            end if
        end if
    end for
end procedure
```

# 6 Anwendung an einem Beispieldatensatz

In diesem Kapitel soll mit Hilfe von EyeMsa ein Beispieldatensatz verglichen werden. Dabei werden die Daten nach dem entworfenen Konzept konfiguriert und die Plausibilität der Daten anschließend diskutiert.

## 6.1 Datensatz

Bei dem verwendeten Datensatz handelt es sich um die Augenbewegungsdaten einer Studie. Die Probanden hatten dabei die Aufgabe auf einem Straßenbahnnetz von einem Startpunkt aus, einen Weg zum Zielpunkt zu finden. Der Stimulus ist das Bahnnetz der belgischen Stadt Antwerpen. Der Start ist durch einen roten Kreis und das Ziel durch eine grüne Hand dargestellt. In Abbildung 6.1 wurden die Bewegungsdaten auf den Stimulus geplottet.

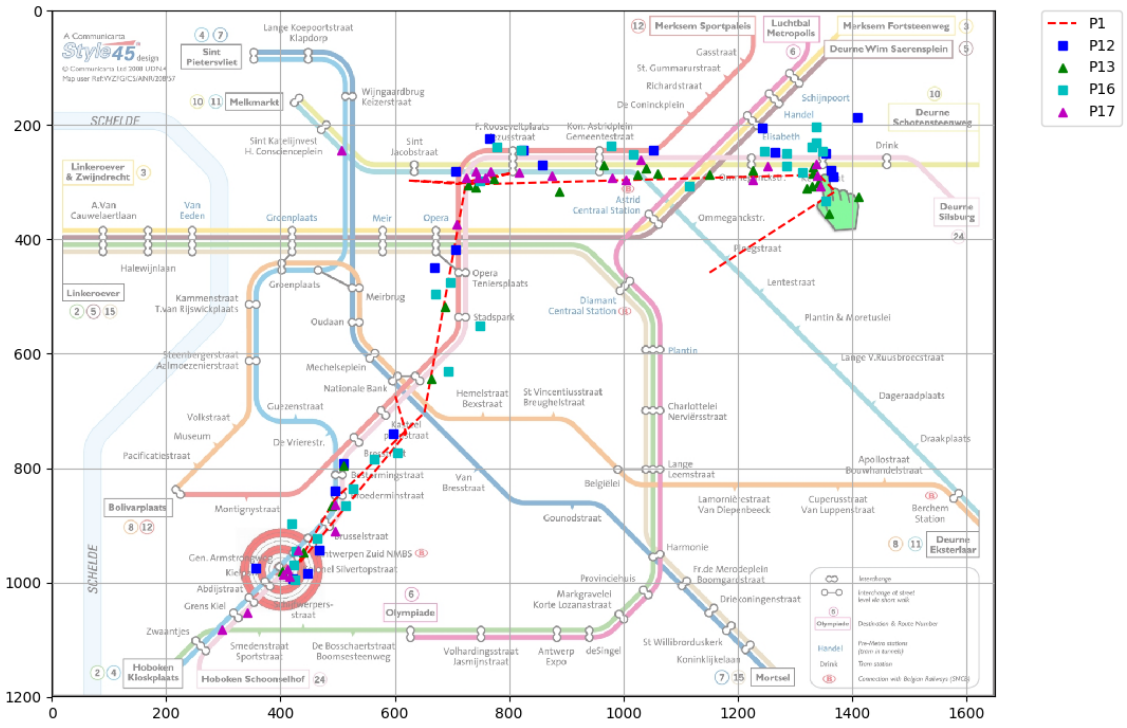


**Abbildung 6.1:** Daten von fünf Personen aus dem Beispieldatensatz geplottet auf den Stimulus.

In diesem werden die Fixationspunkte der Probanden mit unterschiedlichen Farben und Formen dargestellt, siehe Legende der Abbildung 6.1. Die Daten sind schon bereinigt und daher verlaufen die Fixationen sehr genau entlang der Bahnlinien.

## 6.2 Anwendung

Zu Beginn wird die Datei *Data.csv* importiert. Dazu kann die Datei im Webinterface ausgewählt werden und durch „Upload All“ wird diese an den Server gesendet. In Abbildung 6.2 ist der Vorgang bereits abgeschlossen.

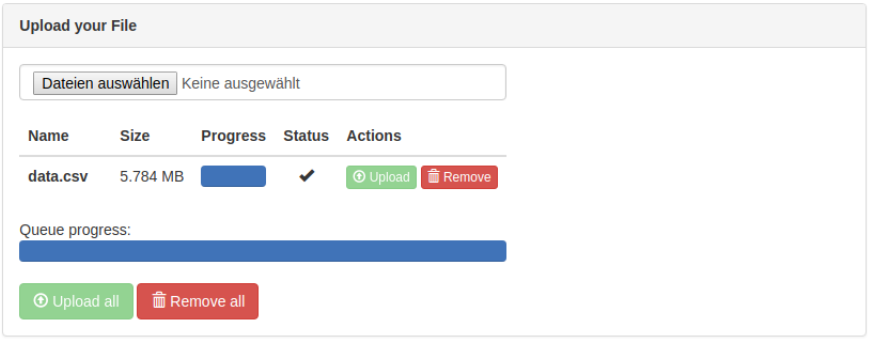


**Abbildung 6.2:** Hochladen der Daten auf EyeMsa.

Die Abbildung 6.3 zeigt eine fertige Konfigurationseinstellung. Es wurde die Fixation als Metrik eingestellt und ein Raster von $7 \times 6$ konfiguriert. Im Stimuliname werden alle in der *csv* enthaltenen Stimulis aufgelistet. In diesem Fall wird „01_Antwerpen_S1.jpg“ ausgewählt und anschließend ein maximale Zielmengengröße von 61 konfiguriert. Bei einer Rasterung von 7x6 werden 42 Raster erstellt. Ein Maximalwert von 42 wäre somit ausreichend.

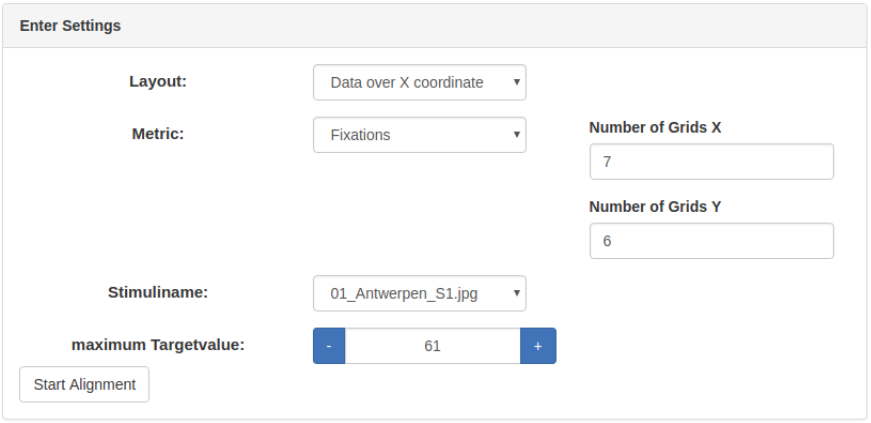


**Abbildung 6.3:** Konfiguration der Datenaufbereitung auf EyeMsa.

Das Ergebnis des Vergleichs ist in Abbildung 6.4 und Abbildung 6.5 dargestellt. In der Consensus Matrix ist jeder Person eine Zeichenkette von Rasterindizes zugeordnet. Zudem werden die übereinstimmenden Werte markiert.

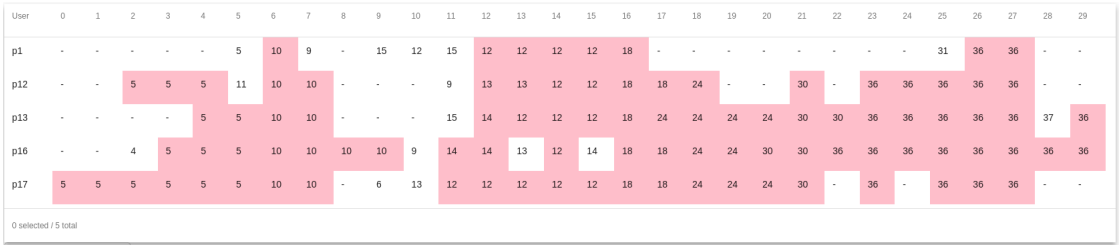


**Abbildung 6.4:** Visualisierung durch die Consensus Matrix auf EyeMsa.

Der Plot zeigt die Koordinatenpunkte der Personen. Auf dem Plot wurde das generierte Raster gezeichnet und die Farben der Personen zur Beschreibung der Plausibilität zugeordnet. Um das Raster zu zeichnen, wird der JSON Struktur in $data[configuration]$ die Achsenschnittpunkte der Rasterlinien übergeben. Zur Analyse der Daten kann somit jederzeit ein Raster gezeichnet werden.

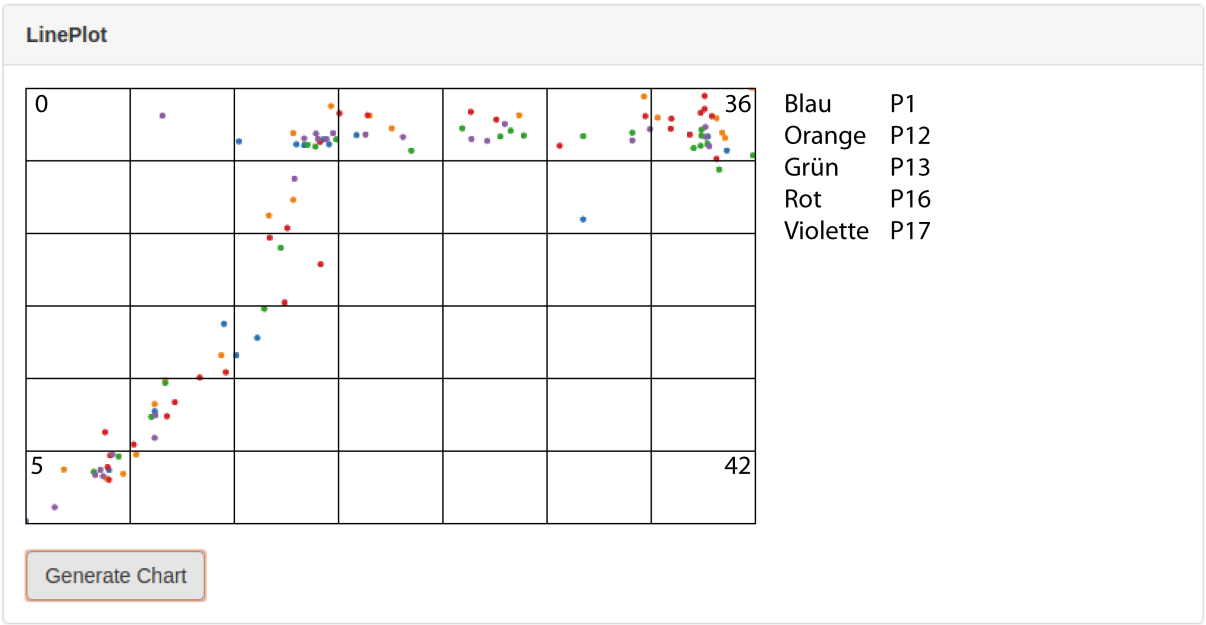


**Abbildung 6.5:** Visualisierung durch den Plot auf EyeMsa.

## 6.3 Plausibilität und Korrektheit

Die Rasterung erzeugt 42 Raster mit der in Abbildung 6.5 dargestellten Indizierung. Vergleicht man den Plot mit der Consensus Matrix, so scheinen die Werte plausibel. Person 17 (Violett) hat, nach Anordnung über die x-Koordinaten, den ersten Punkt in der Plotdarstellung im fünften Raster. Dieser Wert stimmt mit der Matrixdarstellung überein. Die letzten Fixationen der Probanden liegen im 36. Quadranten. In der Consensus Matrix macht sich das durch die Anhäufung der übereinstimmenden Zeichen am Ende der Matrix bemerkbar. In der Consensus Matrix stimmen die ersten Werte der Person 16 und 17 nicht überein. In dem Plot ist zu sehen, dass der erste Punkt von Person 16 (Rot) im vierten Feld liegt und der von Person 17. Somit ist die nicht Übereinstimmung korrekt dargestellt. Die häufigen Übereinstimmungen resultieren aus den sehr ähnlichen Scanpfaden der Probanden. Die Hauptunterschiede liegen bei der Anzahl der benötigten Fixationen und der Anzahl der Fixationen an den AOIs. Diese Unterschiede werden durch das Einfügen der Lücken ausgeglichen. Beispielsweise ist der Scanpath, angeordnet über x, von Proband 1 und Proband 17 sehr ähnlich. Dennoch unterscheiden sich die Zeichenketten sehr stark. Durch das Einfügen der Lücken wird eine sehr hohe Übereinstimmung erzielt. Unter anderem auch, da das Verlängern einer Lücke nicht so intensiv bestraft wird, wie das Einfügen neuer Lücken. Daraus folgt, dass durch einen Vergleich von Zeichenketten aus Eye-Tracking Daten mit Hilfe des implementierten multiplen Sequenzalignment Algorithmus eine Gesamtähnlichkeit und Muster erkannt werden können. Aus der Matrix kann noch mehr gelesen werden. Am Ende der Matrix stimmen in Punkt 25 bis 27 fast alle Werte überein. Diese Anhäufung kann mit einer Einfärbung bei Heatmaps verglichen werden. Der zusätzliche Vorteil ist die direkte Sichtbarkeit von Personen, die aus dem Muster fallen. Beispielsweise stimmen in den Punkten 12-16 fast alle Werte überein. Nur Person 16 weicht ab. Diese Darstellung wäre durch Heatmaps nicht so einfach möglich.

Die aufgebaute Datenstruktur ist in Listing 6.1. Als Beispiel wurden die Daten von Proband 1 exportiert

**Listing 6.1** JSON Struktur des angewendeten Beispieldatensatz.

```
{
"configuration": {
      "maxTargetValue": 61,
      "layout": "x",
      "stimuliname":
      "01_Antwerpen_S1.jpg",
      "gridresult": {
            "y": [
                  186.0, 335.33333333333337, 484.66666666666674,
                  634.0000000000001, 783.3333333333335,
                  932.6666666666669, 1082.0000000000002
                  ],
            "x": [
                  298.0, 457.0, 616.0, 775.0, 934.0, 1093.0,
                  1252.0, 1411.0
                  ]
            },
      "metric": "fixations",
      "filename": "file-1510492035629.csv",
      "scaleValue": 0,
      "grid": {"y": 6,"x": 7,"enabled": true}
      }
"eyedata": [
      {"alignedSequence": [
            -1, -1, -1, -1, -1, 5,10,9,-1, 15,12,15,
            12,12,12,12,18,-1, -1, -1, -1, -1, -1,
            -1, -1, 31,36,36,-1, -1
            ],
      "data": [
            {
            "fixationduration": 250,
            "stimuliname": "01_Antwerpen_S1.jpg",
            "timestamp": 2586,
            "y": 458,
            "x": 1151,
            "id": 0
            },...]
      "user": "p12",
      "rawsequence": [
            5.0, 5.0, 5.0, 11.0, 10.0, 10.0, 9.0, 13.0, 13.0,
            12.0, 12.0, 18.0, 18.0, 24.0, 30.0, 36.0, 36.0,
            36.0, 36.0, 36.0]
            },
      {... P2 ...} ...
}
```

# 7 Zusammenfassung und Ausblick

**Zusammenfassung** Das Ziel dieser Arbeit ist die Entwicklung eines neuen Konzepts, indem Augenbewegungsdaten so aufbereitet werden, dass diese mit einem multiplen Sequenzalignment Algorithmus verglichen werden können. Dazu wird ein Webinterface „EyeMsa“ entwickelt, welches die Daten importiert, vergleicht und durch eine Consensus Matrix visualisiert.

Ähnlich zu dem Tool „ScanMatch“ werden die Daten zweier Probanden vor verarbeitet und anschließend miteinander verglichen. Der Unterschied ist, dass zum Einen die Datenvorverarbeitung mit unterschiedlichen Metriken initialisiert werden kann und zum anderen die Möglichkeit besteht mehrere Sequenzen gemeinsam zu vergleichen. In der Bioinformatik sind im Laufe der Zeit viele Ansätze zum vergleichen von Sequenzen entstanden. Die für den Anwendungszweck relevanten Methoden und Ansätze wurden in dieser Arbeit diskutiert. Da es sich bei den Daten um sehr ähnliche lange und strukturierte Daten handelt und die Anzahl der Sequenzen proportional zur Anzahl der Probanden ist, wurde eine progressive Vergleichsmethode gewählt. Dieser verwendet zur Laufzeitoptimierung einen heuristischen Ansatz. Die Sequenzen werden paarweise global und lokal Verglichen und daraus ein Guide-Tree erzeugt. Entlang dieses Guide-Trees werden die Sequenzen mit einander Verglichen. Als Ergebnis erhält man die bearbeitet Sequenzen, deren Zeichenreihenfolge nicht verändert wurde, sondern nur durch Einfügen von Lücken angepasst wurden. Für die Implementierung wird das *T-Coffee* Tool verwendet, welches von der Seqan Bibliothek bereitgestellt wird. Dieses bietet ein breites Spektrum an Konfigurationsmöglichkeiten Möglichkeiten, was eine Anpassung an die vorhandene Datenstruktur möglich macht.

Die Daten werden zuvor über ein entwickeltes Datenaufbereitungskonzept vor verarbeitet und anschließend mit den initialisierten Vergleichsparametern verglichen. Dabei wurde die Implementierung so Modular wie möglich gehalten. Die Benutzeroberfläche, Datenaufbereitung und der Vergleich wurden als selbstständige Komponenten implementiert, was eine Erweiterung der Funktionalität stark vereinfacht. Die Benutzeroberfläche beinhaltet die Konfiguration und anschließende Visualisierung der Ergebnisse. In der Datenaufbereitung werden die Daten in Sequenzen transformiert, auf einen Wertebereich skaliert und im Vergleich anschließen über die Seqan *T-Coffee* Implementierung verglichen. Die Vergleichsparameter und Daten werden als JSON Datenstruktur bereit gestellt.

Abschließend wurde die Umsetzung auf einen Datensatz angewendet und anschließend ausgewertet. Die verwendeten Daten kommen aus einer Studie, in der die Teilnehmer auf einem Bahnnetz den Weg von einem Startort zu einem Zielort finden mussten. Der Stimulus wurde in ein Raster aufgeteilt und auf die Fixationen untersucht. Dabei haben sich in der Matrix Übereinstimmungen gebildet, die den Scanpath widerspiegeln. Ebenso wurde sichtbar, dass durch diese Analyse und Darstellung die Datenkonzentration und die Abweichungen einzelner Personen sichtbar werden. Mit diesem Analyse verfahren können die Daten effizient in Beziehung gesetzt werden.

**Ausblick** Die Möglichkeiten Eye-Tracking Daten mit einem Sequenzalignment Algorithmus zu vergleichen sind noch nicht ausgeschöpft. Bei der hohen Anzahl an Metriken kann die Datenaufbereitung von EyeMsa noch beliebig erweitert und dabei noch weitere Analyseauswertung durchgeführt werden. Es gibt zudem noch viele weiter Ansätze von Algorithmen die möglicherweise ein optimaleres Ergebnis liefern. Der Vergleichsalgorithmus kann zusätzlich noch mit mehr Vorwissen initialisiert werden. Beispielsweise durch die Konzeption einer Substitutionsmatrix, die den Unterscheidungsgrad analog zur Blosum62 Matrix aus der Bioinformatik bewertet. Die entworfene JSON Struktur bietet noch weitere Möglichkeiten die Daten nachzuarbeiten um weitere Analyse Verfahren auf die verglichenen Daten anzuwenden.

Die implementierte Consensus Matrix ist nur eine Möglichkeit der Visualisierung. Das Tool *iHAT* bietet eine mögliche Darstellung, in der auch die Metadaten mit in die Visualisierung einfließen. Es ist auch vorstellbar die entstandene Vergleichsstruktur anhand des Guide-Trees mit in die Visualisierung einfließen zu lassen.

Die Applikation kann somit noch beliebig weiterentwickelt und das entworfene Konzept für weitere Analysen verwendet werden.

# Literaturverzeichnis


[Cri+10] F. Cristino, S. Mathôt, J. Theeuwes, I. D. Gilchrist. „ScanMatch: A novel method for comparing fixation sequences“. In: *Behavior research methods* 42.3 (2010), S. 692–700 (zitiert auf S. 18).

[D3] *D3- Data-Driven Documents*. https://d3js.org/. Accessed: 2017-08-20 (zitiert auf S. 47).

[Dur+98] R. Durbin, S. R. Eddy, A. Krogh, G. Mitchison. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge university press, 1998 (zitiert auf S. 17, 26).

[Dör+08] A. Döring, D. Weese, T. Rausch, K. Reinert. „SeqAn an efficient, generic C++ library for sequence analysis“. In: *BMC bioinformatics* 9.1 (2008), S. 11 (zitiert auf S. 40, 47).

[GDR09] A. Gogol-Döring, K. Reinert. *Biological sequence analysis using the SeqAn C++ library*. CRC Press, 2009 (zitiert auf S. 47).

[Got82] O. Gotoh. „An improved algorithm for matching biological sequences“. In: *Journal of molecular biology* 162.3 (1982), S. 705–708 (zitiert auf S. 17).

[HH92] S. Henikoff, J. G. Henikoff. „Amino acid substitution matrices from protein blocks“. In: *Proceedings of the National Academy of Sciences* 89.22 (1992), S. 10915–10919 (zitiert auf S. 26).

[Hir75] D. S. Hirschberg. „A linear space algorithm for computing maximal common subsequences“. In: *Communications of the ACM* 18.6 (1975), S. 341–343 (zitiert auf S. 17).

[HM91] X. Huang, W. Miller. „A time-efficient, linear-space local similarity algorithm“. In: *Advances in Applied Mathematics* 12.3 (1991), S. 337–357 (zitiert auf S. 28).

[JH12] F. B.G.J.D.W.K. N. Julian Heinrich Corinna Vehlow. „iHAT: interactive Hierarchical Aggregation Table for Genetic Association Data“. In: (2012) (zitiert auf S. 32, 33, 40).

[JK03] R. Jacob, K. S. Karn. „Eye tracking in human-computer interaction and usability research: Ready to deliver the promises“. In: *Mind* 2.3 (2003), S. 4 (zitiert auf S. 23–25).

[Kec93] J. Kececioglu. „The maximum weight trace problem in multiple sequence alignment“. In: *Combinatorial pattern matching*. Springer. 1993, S. 106–119 (zitiert auf S. 30).

[LAK89] D. J. Lipman, S. F. Altschul, J. D. Kececioglu. „A tool for multiple sequence alignment“. In: *Proceedings of the National Academy of Sciences* 86.12 (1989), S. 4412–4415 (zitiert auf S. 31).

[NHH00] C. Notredame, D. G. Higgins, J. Heringa. „T-Coffee: A novel method for fast and accurate multiple sequence alignment“. In: *Journal of molecular biology* 302.1 (2000), S. 205–217 (zitiert auf S. 17, 31).

[NW70] S. B. Needleman, C. 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 (zitiert auf S. 17, 27).

[Sch07] P. G. Schneider. „Grundlagen der Bioinformatik“. University Lecture. 2007.

[Sie+11] F. Sievers, A. Wilm, D. Dineen, T. J. Gibson, K. Karplus, W. Li, R. Lopez, H. McWilliam, M. Remmert, J. Söding et al. „Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega“. In: *Molecular systems biology* 7.1 (2011), S. 539 (zitiert auf S. 28, 30).

[SMD97] J. Stoye, V. Moulton, A. W. Dress. „DCA: an efficient implementation of the divide-and-conquer approach to simultaneous multiple sequence alignment“. In: *Bioinformatics* 13.6 (1997), S. 625–626 (zitiert auf S. 31).

[SN87] N. Saitou, M. Nei. „The neighbor-joining method: a new method for reconstructing phylogenetic trees.“ In: *Molecular biology and evolution* 4.4 (1987), S. 406–425 (zitiert auf S. 29).

[SW81] T. F. Smith, M. S. Waterman. „Identification of common molecular subsequences“. In: *Journal of molecular biology* 147.1 (1981), S. 195–197 (zitiert auf S. 17, 28).

[Tho94] J. Thompson. „Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice“. In: *Nucleic Acids Res* 22 (1994), S. 4673–4680 (zitiert auf S. 28, 29).

Alle URLs wurden zuletzt am 06 11. 2017 geprüft.

**Erklärung**

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

---

Ort, Datum, Unterschrift