Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering

Projektarbeit Wirtschaftsingenieurwesen

Visual Analysis of Speech-to-Text Engines

Autoren	Gabriel Hunziker
Hauptbetreuung	Prof. Dr. Mark Cieliebak Katsiaryna Mlynchyk

Datum	22.12.2023

Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering

Erklärung betreffend das selbständige Verfassen einer Projektarbeit an der School of Engineering

Mit der Abgabe dieser Projektarbeit versichert der/die Studierende, dass er/sie die Arbeit selbständig und ohne fremde Hilfe verfasst hat. (Bei Gruppenarbeiten gelten die Leistungen der übrigen Gruppenmitglieder nicht als fremde Hilfe.)

Der/die unterzeichnende Studierende erklärt, dass alle zitierten Quellen (auch Internetseiten) im Text oder Anhang korrekt nachgewiesen sind, d.h. dass die Projektarbeit keine Plagiate enthält, also keine Teile, die teilweise oder vollständig aus einem fremden Text oder einer fremden Arbeit unter Vorgabe der eigenen Urheberschaft bzw. ohne Quellenangabe übernommen worden sind.

Bei Verfehlungen aller Art treten die Paragraphen 39 und 40 (Unredlichkeit und Verfahren bei Unredlichkeit) der ZHAW Prüfungsordnung sowie die Bestimmungen der Disziplinarmassnahmen der Hochschulordnung in Kraft.

Ort, Datum:	Unterschriften:
Winterthur, 22.12.2023	G. Henziker
.....	.....
	.....
	.....

Das Original dieses Formulars ist bei der ZHAW-Version aller abgegebenen Projektarbeiten zu Beginn der Dokumentation nach dem Titelblatt mit Original-Unterschriften und -Datum (keine Kopie) einzufügen.

Zusammenfassung

Die vorliegende Arbeit befasst sich mit der Weiterentwicklung des Webbasierten Text Alignment Tools der ZHAW. Diese Anwendung dient zur Unterstützung bei der Entwicklung von Speech-to-Text Modellen. Sie ermöglicht einen grafischen Vergleich zwischen Referenzen und Hypothesen und soll dadurch allfällige Schwachstellen dieser Modelle identifizieren. Die Anwendung soll ohne Programmierkenntnisse verwendet werden können.
Im Rahmen dieses Projekts lag der Fokus auf dem Frontend. Ziel war die Verbesserung der User Experience und der Zuverlässigkeit der Anwendung. Besonders hervorzuheben sind das durchgängig gestaltete Benutzerkonzept und die optimierten Dialoge zur Texteingabe- und auswahl. Eine weitere Erleichterung stellt die Implementierung einer Import- und Exportfunktion des Ergebnisses dar. Diese ermöglicht dem Nutzer die Einsparung der Wartezeit für spätere Analysen. Zusätzlich wurde die Oberfläche der Anwendung so gestaltet, dass Nutzer mit wenig Vorkenntnissen die Anwendung verwenden können. Die intuitive Bedienung des Tools wird durch dynamische Objekte gefördert. Herausfordernd war die Umsetzung eines intuitiven Dialogs zur Auswahl von Pfaden aus JSON-Dateien, da diese beliebig strukturiert sein können. Ein weiteres Ziel war die Dokumentation des Tools, um zukünftigen Projektteilnehmern den Einstieg zu vereinfachen.
In Zukunft soll die Anwendung mit diversen Funktionen erweitert werden.

Abstract

This thesis concerns the enhancement of ZHAW's web-based text alignment tool, which supports the development of speech-to-text models. The tool allows a graphical comparison between references and hypotheses, identifying any weaknesses in the models. It is designed to be user-friendly, requiring no programming knowledge.
The project focused on improving the front-end to enhance user experience and application reliability. The user concept is consistently designed and the dialogues for text input and selection are optimised. The implementation of an import and export function for the results further simplifies the process and saves waiting time for later analyses. The user interface is designed to be intuitive and user-friendly, even for those with little prior knowledge. Dynamic objects support the intuitive operation of the tool. Implementing an intuitive dialogue for selecting paths from JSON files presented a challenge due to their flexible structure.
Additionally, documenting the tool was a priority to facilitate future project participation.
The application will be expanded with various functions in the future.

Vorwort

Die Anwendung von Text-to-Speech Algorithmen wird in der heutigen Zeit immer wichtiger. Insbesondere meine Faszination wie Computer menschliche Stimmen verstehen können hat mich überzeugt mich diesem Projekt zu widmen. Erstaunlich ist unter anderem, wie gut beispielsweise Siri mittlerweile Schweizerdeutsch versteht. Die Weiterentwicklung einer Webseite in Typscript war für einen Wirtschaftsingenieur herausfordernd. Da ich jedoch sehr am Programmieren interessiert bin war dies eine willkommene Herausforderung.

Ich möchte mich herzlich bei meinen Betreuern Prof. Dr. Mark Cieliebak und Katsiaryna Mlynchyk bedanken, welche sich stets Zeit für meine Fragen genommen haben und mir mit ihren Inputs und Vorschlägen immer hilfreich bei Seite standen.

Disclaimer:
Natürlich wurde für diese Arbeit teilweise moderne KI-Basierte Tools wie DeepL Write oder Chat-GPT verwendet. Diese dienten jedoch nur als Unterstützung bei Formulierungen oder als Denkanstoss. Alle Texte und Inhalte wurden durch mich selbst erfasst und recherchiert.

Inhaltsverzeichnis

1. Einleitung

In einer Welt, die zunehmend durch digitale Technologien geprägt ist, haben sich KI-basierte Anwendungen, insbesondere Speech-to-Text (STT) Algorithmen als integraler Bestandteil in unseren Alltag eingefügt. Diese Algorithmen spielen eine entscheidende Rolle in einer Vielzahl von Anwendungen. Damit Computer effektiv mit der menschlichen Stimme interagieren können, dem intuitivsten und natürlichsten Eingabemedium des Menschen, ist eine exakte und fehlerfreie Verarbeitung dieser Daten unabdingbar. Doch diesem Potential stehen diverse Herausforderungen wie die Dialektvielfalt oder die Qualität der Aufnahmen entgegen. Um diese Probleme zu minimieren arbeiten Forschende weltweit an der Optimierung der Spracherkennungsalgorithmen. Um allfällige Schwachstellen zu identifizieren sind sie auf Tools angewiesen, welche den Textoutput mit einer Referenz oder weiteren Hypothesen vergleichen können. Da hierzu oft grosse Datenmengen verarbeitet werden müssen, stösst ein manueller Vergleich schnell an seine Grenzen und ist nicht mehr effizient durchführbar. Deshalb sind automatisierte Tools für diesen Aufgabenbereich unabdingbar. Bestehende Lösungen sind oft nicht flexibel und kompliziert in der Anwendung.
Hier setzt das Alignment Tool der ZHAW an, welches eine Schlüsselrolle bei der Analyse und Bewertung der Leistungsfähigkeit von STT-Algorithmen spielen soll. Dieses Tool soll die Aufgabe des Textvergleichs für STT-Systeme auf eine einfache Art und Weise ermöglichen und Abweichungen zwischen Referenz- und Hypothesentexten zuverlässig erkennen. Dies ist entscheidend um die Genauigkeit und Zuverlässigkeit von Spracherkennungsanwendungen zu erhöhen.

Im Rahmen dieser Arbeit soll das Frontend des Alignment Tools weiterentwickelt und ausgebaut werden. Ziel ist es eine benutzerfreundliche und effiziente Schnittstelle zu schaffen, die eine intuitive Analyse der STT-Ergebnisse ermöglicht.

1.1. Ausgangslage

1.1.1. Konkurrierende Projekte

Im Bereich des Textvergleichs und der Analyse von Speech-to-Text Ergebnissen existieren bereits mehrere Projekte. Diese variieren in ihrem Umfang und ihrer Komplexität. Viele dieser Lösungen setzten ausserdem Kenntnisse von Programmiersprachen für die Anwendung voraus und sind deshalb nicht leicht zu bedienen. Ausserdem sind die meisten dieser Lösungen für einzelne Texte ausgelegt und es lassen sich keine JSON oder CSV-Dateien verarbeiten.
Ein weiterer limitierender Faktor vieler Lösungen ist die Art, wie die Ergebnisse präsentiert werden. Häufig beschränken sich diese auf die Ausgabe von Kennzahlen wie die Word Error Rate (WER), Word Information Lost (WIL) oder die Character Error Rate (CER). Eine grafische Darstellung und ein visueller Vergleich sind oft nicht möglich.

Eine bekannte Python Library für die Analyse von Referenz- und Hypothesentexten ist beispielsweise JiWER. Diese Library kann diverse Kenngrössen anhand eines Referenz- und Hypothesentextes berechnen. Jedoch fehlen der Library diverse Funktionen, wie die automatische Verarbeitung ganzer Dateien oder das Ersetzten von Abkürzungen diverser Sprachen. Da JiWER eine Python Library ist würden sich diese Funktionen in einem Pythonscript erweitern lassen. Dazu sind jedoch Python Kenntnisse vorausgesetzt [1].

Andere webbasierte Tools wie den «Kensho Scribe Word Error Rate Calculator» bieten eine leicht zu bedienende Benutzeroberfläche, in welcher Texte verglichen werden können und die Word Error Rate berechnet wird. Es besteht jedoch nicht die Option eines Dateiuploads, um hunderte Äusserungen gleichzeitig zu verarbeiten. Ausserdem ist die Visualisierung bei grösseren Texten nicht sehr übersichtlich gestaltet [2].

Abschliessend lässt sich feststellen, dass im Bereich der Text Alignment Tools eine Martlüke besteht. Insbesondere solcher, die in der Lage sind mehrere Texte gleichzeitig zu vergleichen

und diese Ergebnisse optisch übersichtlich sowie benutzerfreundlich darzustellen. Diese Erkenntnis unterstreicht die Relevanz des ZHAW-Alignement Tools.

1.1.2. Bestehendes Tool der ZHAW

Das bestehende Alignment Tool wurde durch ZHAW-Angehörige entwickelt. Das Ziel des Alignment Tools ist es, die Ergebnisse von Speech-to-Text (STT) Modellen zu vergleichen. Diese Modelle wandeln gesprochene Sprache in Text um. Das Tool dient zur Bestimmung der Genauigkeit dieser Umwandlung. Es vergleicht die Referenztexte mit den entsprechenden Hypothesen aus den Modellen. Dies ist besonders nützlich, um Abweichungen und Fehler der Spracherkennung zu identifizieren und die Leistung der STT-Modelle zu bewerten. Der Dateiupload, die Selektion und Darstellung des Ergebnisses sind bereits möglich. Ein funktionierendes Backend, welches die Daten berechnet ist bereits entwickelt. Das Hauptproblem ist jedoch, dass das Tool nicht selbsterklärend ist. Nutzer, welche mit der Anwendung nicht vertraut sind, stossen häufig auf Schwierigkeiten. Ein spezifisches Vorwissen oder Erfahrung wird vorausgesetzt, um das Alignment Tool effektiv zu nutzen.
Darüber hinaus ist die Benutzeroberfläche zurzeit noch inkonsistent und nicht sehr intuitiv gestaltet. Dies führt zu einer erschwerten Navigation und kann die Effizienz der Nutzer bei der Anwendung beeinträchtigen. Ein weiteres Problem ist, dass dem Nutzer bei der Auswahl der Textreferenzen und Hypothesen nicht immer ganz klar ist, was genau ausgewählt werden muss. Hierzu fehlt momentan auch noch der Platz für Hilfetexte, welche dem Anwender erklären, was genau er tun soll.
In Anbetracht dieser Herausforderungen besteht ein Bedarf, das Tool zu überarbeiten und das Nutzererlebnis zu verbessern.

1.2. Zielsetzung

Das Ziel dieser Arbeit ist es, dass Frontend des bereits bestehenden Alignment Tools weiterzuentwickeln. Dies umfasst die Gestaltung eines intuitiven User Interface für die Selektion der Daten nach dem Upload, wobei der Fokus auf dem Import von JSON-Dateien liegt. Ausserdem soll die Applikation ein gesamtheitliches Erscheinungsbild erhalten, um die Benutzererfahrung auf ein bestmögliches Niveau zu bringen.
Im Rahmen dieser Weiterentwicklung sollen auch neue Features und Verbesserungen implementiert werden, die das Tool funktionaler und zugänglicher für ein breites Publikum machen. Dazu gehört unter anderem eine Import- und Exportfunktion, damit ein Ergebnis zu einem späteren Zeitpunkt wieder angezeigt werden kann. Das Endziel ist eine robuste und benutzerorientierte Webanwendung zu erschaffen, welche den Anwendern zur Verfügung gestellt werden kann. Die Optimierung des Backends ist nicht Teil dieser Projektarbeit, da diese parallel durch andere ZHAW-Angehörige weiterentwickelt wird.

Die Struktur der Arbeit umfasst in der Einleitung eine Analyse bestehender Tools für dieses Aufgabengebiet, eine Übersicht über den aktuellen Stand des ZHAW Alignment Tools sowie die konkrete Umsetzung der Verbesserung und die Darstellung der daraus folgenden Resultate. Abschliessend erfolgt eine Diskussion und ein Ausblick auf mögliche zukünftige Entwicklungsmöglichkeiten des Alignment Tools.

Für einige Teile dieser Arbeit werden gewisse Grundkenntnisse der Funktion und Aufbau von Webanwendungen vorausgesetzt.

2. Übersicht der bestehenden Applikation

2.1. Überblick

Das in dieser Arbeit weiterentwickelte Alignment Tool wurde als Proof of Concept von Studierenden der ZHAW entwickelt und ist bereits auf einem öffentlich zugänglichen Server publiziert. Die Applikation besteht aus zwei Hauptkomponenten: das Frontend und das Backend, welche jeweils unterschiedliche Funktionen erfüllen. Das Frontend der Applikation wurde mit React, einer verbreiteten JavaScript Bibliothek entwickelt [3]. Als Skriptsprache wird TypeScript verwendet. TypeScript ist ursprünglich aus JavaScript entstanden, bietet jedoch den Vorteil, dass sie typisiert ist. Dadurch entstehen weniger Laufzeitfehler im Vergleich mit JavaScript. Jedoch ist der Programmieraufwand deswegen bei Typescript leicht erhöht .

Die primäre Aufgabe des Frontends besteht darin, dem Benutzer eine intuitive Schnittstelle für die Auswahl und Übermittlung der Daten zu bieten. Ausserdem wird im Frontend das Ergebnis zur Auswertung grafisch ansprechend dargestellt.
Eine grosse Herausforderung ist die Heterogenität der Daten. Die Anwendung soll mit möglichst vielen Dateiformaten und Strukturen arbeiten können.

Das Backend welches mit Python und dem Webframework Flask entwickelt wurde, ist für die Verarbeitung der vom Frontend empfangen Daten zuständig. Durch die Programmiersprache Python und der Kompaktheit von Flask ist dieses Webframework gut geeignet für Backend Anwendungen [4], [5]. Im Backend werden die empfangenen Texte aufbereitet und es vergleicht die Hypothesen Wort für Wort mit der Referenz. Anschliessend wird das Ergebnis zurück an das Frontend übermittelt und visuell dargestellt. Der Quellcode des Projekts ist auf Github veröffentlicht und beinhaltet eine Beschreibung der Grundfunktionen und des Projektaufbaus [6].

Da im Projekt Front- und Backend strikt getrennt sind, kann jeder der zwei Teile separat oder als ein zusammenhängendes Paket verwendet werden. Dies erleichtert die Wartung und Weiterentwicklung der einzelnen Komponenten. Der Quellcode für die Frontend-Applikation ist im Verzeichnis **alignment_tool_app** zu finden, während sich das Flask Backend im Pfad **backend** des Projekts befindet.

Für die Entwicklung und Bereitstellung des Alignment Tools kann ein Docker Stack verwendet werden. Dieses Stack beinhaltet einen separaten Docker Container für das Frontend und das Backend. Hierzu befindet sich im Stammverzeichnis eine Docker-compose Datei. Um das Docker Stack Lokal auf dem Rechner zu Starten sind jedoch einige Änderungen nötig, da das Projekt die Variablen für das Produktiv Setup beinhaltet. Die entsprechenden Schritte sind in der Github Dokumentation beschrieben, welche unter anderem im Anhang dieser Arbeit angefügt ist.
Wenn beispielsweise nur das Frontend weiterentwickelt wird, ist es möglich nur den Backend Container in der Docker Umgebung zu starten, um diesen für die Anfragen aus dem Frontend zu verwenden.

Im nächsten Abschnitt werden die wichtigsten Grundbegriffe erklärt, welche elementar für das Verständnis der Funktionen sind.
Anschliessend bieten die nachfolgenden Kapitel einen Überblick über die verwendeten Bibliotheken und Frameworks im Projekt. Ausserdem wird das bestehende Frontend analysiert und die Funktion des Backends beschrieben.

2.2. Erklärung der wichtigsten Begriffe

Für das Verständnis der Applikation ist eine Erklärung einiger Begriffe erforderlich.

Corpus
Im Bereich des Natural Language Processing (NLP) bezeichnet ein Corpus eine Sammlung von Textdaten. Diese Texte können aus verschiedenen Quellen stammen. Spech-to-Text Modelle werden anhand dieser Corpus Daten trainiert. Im Alignment Tool wird der Corpus verwendet, um die Leistungsfähigkeit verschiedener STT-Modellen zu ermitteln.

Utterance
Eine Utterance (Äusserung) ist Teil des Corpus und beschreibt einen einzelnen gesprochenen Textabschnitt. In der Anwendung dieser Applikation besitzt jede Utterance mehrere Hypothesen, welche mit einem Referenztext verglichen werden.

Word Error Rate (WER)
Die Word Error Rate ist ein verbreitetes Mass im Bereich des NLP, welches dazu dient, die Genauigkeit von Speech-to-Text Modellen zu bewerten. Sie misst den Prozentsatz der Fehler im transkribierten Text im Vergleich zur Referenz. Die Berechnung geschieht durch die Bestimmung der Anzahl der Substitutionen (S), Einsetzungen (I) und Entfernung (D) von Wörtern in der Hypothese. Anschliessend wird dieser Wert durch die Anzahl Wörter geteilt, um den Prozentsatz zu erhalten [7].
Die Word Error Rate ist die bekannteste Kennzahl, um die Qualität von Speech-to-Text Modelle zu beurteilen. Da sie jedoch sehr fehleranfällig ist, ist sie nicht unumstritten. Natürlich gibt es weitere Kennzahlen wie beispielsweise die Character Error Rate (CER). In dem vorliegenden Projekt ist zurzeit jedoch nur die Berechnung der WER implementiert.

2.3. Frontend

2.3.1. Verwendete Bibliotheken und Frameworks

React
React wurde durch Facebook entwickelt und wird nun durch die Meta Open Source Initiative betreut. Der Fokus von React liegt auf einer effizienten und flexiblen JavaScript Bibliothek für das Erstellen von Benutzeroberflächen. Mithilfe von React Native können Webseiten als auch Smartphone Applikationen entwickelt werden. Das Kernkonzept von React basiert auf der Verwendung modularer Codeeinheiten, den sogenannten Komponenten. Diese tragen dazu bei, komplexe Applikationen durch den Einsatz kleiner, wiederverwendbarer Teile aufzubauen [3], [8].

Vite
Vite ist ein moderner Frontend Entwicklungsserver, welcher darauf ausgerichtet ist, den Entwicklungsprozess durch die Implementierung des Hot-Module-Replacement (HMR) zu beschleunigen. Diese Technologie ermöglicht das sofortige Aktualisieren der Vorschau direkt nach dem Speichern der Codeänderungen. Dadurch wird die Produktivität und die Effizienz im Entwicklungsprozess erheblich gesteigert [9].

MUI
MUI ist eine beliebte React Komponentenbibliothek. Sie bietet eine umfangreiche Sammlung an vorbereiteten und anpassbaren Komponenten. Diese erleichtern das Erstellen moderner und konsistenten Benutzeroberflächen. MUI orientiert sich am Material Design Konzept von Google [10].

2.3.2. Erscheinungsbild der aktuellen Applikation

Startseite und Dateiupload
Das Frontend der bestehenden Applikation wurde für das erste Konzept schlicht und einfach gehalten und konzentriert sich auf die Demonstration der Kernfunktionalität. Das Hauptmenu (siehe Abbildung 2) besteht aktuell aus zwei Tabs («Drag & Drop» und «Manual Insert»).
Der «Drag & Drop» Dialog (siehe Abbildung 2) dient gleichzeitig als Startseite. Dateien lassen sich entweder auswählen oder können per Drag & Drop in das Feld gezogen werden.

Abbildung 1: Dateiupload der bestehenden Applikation

Abbildung 2: Manuelles erfassen von Äusserungen der bestehenden Applikation

Hinter dem Reiter «Manual Insert» befindet sich die manuelle Eingabemaske (siehe Abbildung 2). Diese ist übersichtlich gestaltet. Der Nutzer kann zusätzliche Hypothesen hinzufügen oder eine weiter Eingabemaske hinzufügen. Wenn eine Box abgeschlossen ist, wird diese mit dem «Save this Utterance» Button gespeichert. Schlussendlich werden die Texte mit «Submit & Align» mithilfe der HTTP POST-Methode an das Backend zur Verarbeitung gesendet.

Auswahl der Referenz- und Hypothesentexte nach dem Dateiupload

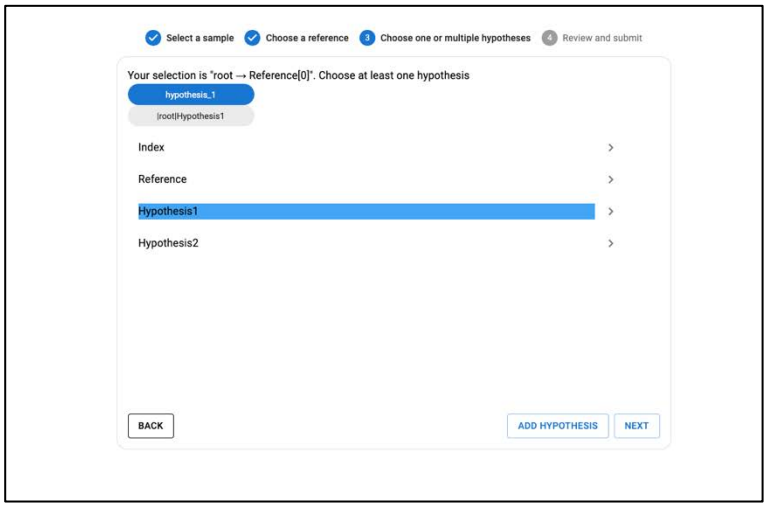

Abbildung 3: Hypothesen Selektion der bestehenden Applikation

Nachdem eine JSON oder CSV-Datei hochgeladen wurde, erscheint ein Dialog (siehe Abbildung 3), welcher in vier Teilschritte aufgeteilt ist. Darin muss der Anwender als erstes den Pfad der Samples auswählen. Anschliessend folgt die Auswahl der Textreferenz und der Hypothesen. Schlussendlich kann die Auswahl überprüft und an das Backend zur Auswertung gesendet werden.

Auswertung der Ergebnisse

Abbildung 4: Layout des Resultats der bestehenden Applikation

Sobald das Backend die Auswertung abgeschlossen hat, werden die aufbereiteten Daten zurück an das Frontend gesendet und visuell dargestellt. Im Dropdown-Menü wird dem Benutzer eine Liste aller verarbeiteten Texte angezeigt. Daneben ist die Word Error Rate (WER) ersichtlich. In der Alignment Map ist ersichtlich, ob Wörter in der Hypothesis korrekt erkannt wurden oder ob ein Fehler vorliegt. Verschiedenen Fehlerarten sind dabei farblich markiert, wodurch eine intuitive und schnelle Erkennung von Fehlern ermöglicht wird. Im Abschnitt «Detail» sind die einzelnen Wörter ersichtlich. Dadurch lässt sich optimal identifizieren, ob der Algorithmus im Backend richtig gearbeitet hat.

2.4. Backend

In diesem Kapitel sollen die wichtigsten Merkmale des Backends erläutert werden. Der Fokus liegt jedoch auf der Verwendung der Schnittstelle zwischen Backend und Frontend, da die Backend Entwicklung nicht Teil dieser Arbeit ist.

2.4.1. Verwendete Bibliotheken und Frameworks

Das Backend des Alignment Tools ist in der Programmiersprache Python geschrieben. Als Webframework wird Flask eingesetzt. Flask ist ein minimal gehaltenes Webframework, welches nur die wichtigsten Bibliotheken enthält. Deshalb eignet es sich bestens für kleinere Webseiten oder als Backendframework.
Zusätzlich kommen weiter Python Bibliotheken wie numpy, pandas und weitere zum Einsatz.
In der Produktivversion des Alignment Tools wird das Backend mittels einem Gunicorn Webserver bereitgestellt.

2.4.2. Funktionalitäten des Backends

Das Backend des Alignment Tools empfängt Daten zur Auswertung vom Frontend. Für diesen Datenaustausch wird vom Backend eine REST-Schnittstelle mit zwei verschiedenen Endpunkten bereitgestellt.
In einem ersten Schritt werden die empfangenen Daten aufbereitet. Dies umfasst das Entfernen von Abkürzungen, Sonderzeichen und das Ersetzten von Grossbuchstaben, um eine konsistente Analyse sicherzustellen.
Anschliessend wird jedes Wort der verschiedenen Hypothesen mit den Wörtern aus den Referenztexten verglichen. Bei einer Abweichung wird untersucht, ob es sich um eine Ersetzung, Entfernung oder Hinzufügung eines Wortes handelt.
Ein weiterer wichtiger Schritt im Backend ist die Berechnung verschiedener Kennzahlen. Momentan wird in erster Linie die Word Error Rate berechnet.
Schlussendlich wird das Ergebnis inklusive aller Details zu jedem Wort der Hypothesen zurück an das Frontend gesendet. Dadurch wird eine detaillierte Darstellung der Ergebnisse im Frontend ermöglicht.

2.4.3. Backend Endpunkte

Das Backend stellt zwei Endpunkte zur Verfügung an welche Daten zur Auswertung gesendet werden können.

/api/upload_file

Dieser Endpunkt wird verwendet, wenn eine Auswertung einer hochgeladenen Datei durchgeführt werden soll. Zusätzlich zur Datei muss das Frontend den Pfad zu dem Referenztext, den Hypothesen, sowie den sample Pfad im Body übertragen. In der folgenden Tabelle 1 sind die benötigten Parameter inklusive eines Beispiels aufgelistet.

Tabelle 1: Parameterübersicht für den Endpunkt "/api/upload_file"

Schlüssel	Datentyp und Details	Beispielwert
file	Binärdatei (.json oder .csv)	(Binärdatei)
selection	JSON, Pfad der Referenz und der Hypothesen	`{ "reference":"|root|example|reference", "hypothesis1":"|root|example|hypothesis_one", "hypothesis2":"|root|example|hypothesis_two" }`
samples	String, Pfad der Samples	`'|root'`

Als Trennzeichen im Pfad muss dabei zwingend «|» verwendet werden.

/api/send_utterances
Der Endpunkt «send_utterances» wird für das Senden des manuellen Inputs verwendet. Hierzu wird im Body nur ein Schlüssel «utterances» gesendet, welcher alle Referenzen und Hypothesen enthält. In der folgenden Tabelle 2 sind die benötigten Parameter inklusive eines Beispiels aufgelistet.

Tabelle 2: Parameterübersicht für den Endpunkt "/api/send_utterances"

Schlüssel	Datentyp und Details	Beispielwert

selection	JSON, Enthält alle Referenzen und Hypothesen, jede utterance ist ein eigener Schlüssel	[{
		"reference":"this is a test sentence",
		"hypothesis_1":"this is test sentence",
		"hypothesis_2":"this is a test sentence"
		},{
		"reference":"next utterance",
		"hypothesis_1":"next utterance",
		"hypothesis_2":"next utterances"
		}]

2.4.4. Backend Antwort nach der Verarbeitung
Die Antwort des Servers wird im JSON-Format geliefert. Ein Beispiel dazu ist im nachfolgenden Codeausschnitt ersichtlich. Auf der obersten Ebene des Objekts befinden sich zwei Schlüssel: «alignment» und «systems».
Unter «alignment» befindet sich der Corpus, welcher alle gesendeten Texte enthält. Die Word Error Rate (WER) wird standartmässig mit dem Wert «null» übertragen, da diese Kalkulation im Frontend stattfindet. Innerhalb des Schlüssels «words» erfolgt die Auswertung der einzelnen Wörter aus der Referenz. Das System führt hier einen Vergleich für jedes Wort aus den Hypothesen mit der Referenz durch. Die Ergebnisse für die Wörter aus den Hypothesen sind wiederum im Element «items» aufgelistet.

Innerhalb des Schlüssels «systems» befinden sich aktuell noch keine Resultate. Dieser Schlüssel ist für die Sammlung verschiedener Kennzahlen zu den Hypothesen vorgesehen.

```
{
    "alignment": {
        "corpus": {
            "0": {
                "id": 0,
                "reference": "word1 reference",
                "utterance_id": 0,
                "wer": null,
                "words": [
                    {
                        "items": [
                            {
                                "text": "word1",
                                "type": "Correct"
                            }
                        ],
                        "text": "word1"
                    },
                    {
                        "items": [
                            {
```

```
                                "text": "hypothesis1",
                                "type": "Substitution"
                            }
                        ],
                        "text": "reference"
                    }
                ]
            }
        }
    },
    "systems": [
        {
            "alignment_scores": [],
            "name": "hypothesis_1"
        }
    ]
}
```


3. Weiterentwicklung der Frontendapplikation

Wie bereits in Kapitel 1.1 erläutert, besteht das primäre Ziel dieser Arbeit in der Weiterentwicklung und Optimierung der Benutzeroberfläche des bestehenden Alignment Tools. Ziel ist es allfällige Schwachstellen in der Bedienungsfreundlichkeit und Zuverlässigkeit der Applikation zu beseitigen. Der Nutzer soll durch die Applikation geführt werden und das Tool ohne zusätzliche Anleitung bedienen können. Um dies zu erreichen ist ein einheitliches Erscheinungsbild der Applikation inklusive Erklärungen zu den Funktionen nötig. Ausserdem soll die Auswahlmaske für die Referenzen und Hypothesen durch ein neues Userinterface ersetzt werden, bei welchem die Texte effizienter ausgewählt werden können. Des Weiteren wird die Applikation um eine Import- und Exportfunktion für das Resultat erweitert, damit dieselbe Datei nicht mehrmals verarbeitet werden muss. Schlussendlich werden diverse kleinere Anpassungen wie das Hinzufügen unterstützender Seiten und das Strukturieren des bestehenden Codes vorgenommen. In diesem Kapitel wird der Prozess der Weiterentwicklung detailliert beschrieben.

3.1. Konzeption der Anpassungen

Der Prozess der Weiterentwicklung des User Interfaces begann mit einer Analyse des bestehenden Designs und der Funktionalität. Im nächsten Schritt wurden verschiedene Mockups erstellt. Diese dienten als visuelle Grundlage für die geplanten Änderungen der Benutzeroberfläche und sind im Anhang dieser Arbeit angefügt.
Während des Entwicklungsprozesses wurden die Funktionen der Applikation Regelmässig analysiert und diskutiert. Dadurch weichen die Mockups in diversen Punkten von der finalen Applikation ab. Die grundlegenden Ideen und Anordnungen wurden jedoch in angepasster Form übernommen.

3.2. Integration und Anpassung der Features

In diesem Unterkapitel werden die Schritte zur Implementierung neuer Funktionen sowie die Anpassungen und Optimierung bestehender Elemente des Alignment Tools beschrieben. Die folgenden Abschnitte geben einen Einblick in die technischen Details und die Überlegungen hinter den Verbesserungen.

3.2.1. Implementierung der Navigationsleiste

Um die Benutzerfreundlichkeit zu erhöhen, wurde der Applikation eine Navigationsbar hinzugefügt. Diese ermöglicht dem Nutzer den Wechsel zwischen den einzelnen Seiten des Alignment Tools. Da die Navigationsleiste auf allen Seiten sichtbar ist, wird ein einheitliches Erscheinungsbild gefördert.
Weil die Navigationsleiste ein separates Element innerhalb der Applikation darstellt, wurde sie als neue Komponente angelegt. Diese befindet sich im Ordner /**src/features/Navbar**.
Als Vorlage wurde die AppBar Komponente von MUI verwendet [11]. Für die Navigation zwischen den verschiedenen Bereichen kommt die Funktion «**useNavigate**» aus dem react-router-dom zum Einsatz. Diese Funktion ermöglicht ein nahtloses und intuitives Wechseln der Ansichten innerhalb der Applikation. Für den Link der Startseite wurde bewusst ein direkter HTML-Link hinterlegt. Dadurch wird die komplette Seite neu geladen und alle Variablen zurückgesetzt.
Um die Navigationsleiste konstant auf allen Seiten und Komponenten hinweg sichtbar zu halten, wurde sie direkt in die Hauptdatei der React Applikation (**main.tsx**) integriert. Dadurch ist es nicht mehr erforderlich, die Navigationsleiste einer einzelnen Komponente zuzuweisen.

3.2.2. Überarbeitung der Startseite

Ein wichtiger Bestandteil dieser Projektarbeit ist die Überarbeitung der Startseite. Das Ziel ist es, eine intuitive und informative erste Anlaufstelle für den Nutzer zu schaffen, welche eine klare Übersicht über die wichtigsten Funktionen des Tools bietet.
Die Startseite soll als direkter Einstiegspunkt zu den verschiedenen Funktionen dienen, damit der Nutzer nach der Einleitung nicht auf eine andere Seite wechseln muss. Die Funktionen sollen übersichtlich und einheitlich dargestellt werden.

3.2.3. Textselektion für JSON-Dateien

Da der aktuelle Dialog zur Auswahl von Textreferenzen schwer verständlich ist, wurde beschlossen, ihn durch eine neue Lösung zu ersetzen. Dazu soll die Navigation durch die JSON-Datei mithilfe einer Baumstruktur erfolgen. Während der Auswahl von Referenzen und Hypothesen soll eine Anleitung eingeblendet werden, die sich je nach Benutzerschritt variabel anpasst. Ausserdem sollten sich die Schaltflächen auf der Seite der entsprechenden Funktion dynamisch anpassen oder gesperrt und freigegeben werden.

Als Grundbaustein für die Baumstruktur wurde die TreeView Komponente von MUI verwendet [12]. Mithilfe dieses Bausteins wurde eine neue Komponente «j**sonTreeView.tsx**» innerhalb der Hauptkomponente «**fileUpload**» erstellt. Die technische Herausforderung der Umsetzung liegt jedoch bei dem Inputformat der JSON-Datei. Dieses darf beliebig strukturiert sein und unendlich viele Ebenen besitzen. Deshalb musste eine Funktion entwickelt werden, welche durch alle Elemente des JSON iteriert und bei jedem Element überprüft, ob es sich um ein Objekt, Schlüssel oder Wert handelt. Dazu wurde folgende Funktion implementiert:

```
const renderTree = (jsonObject: any, path: string, parentCount = 0) => {
    if (!jsonObject || parentCount >= maxTreeItems) return null;

    let currentLevelCount = 0;

    if (typeof jsonObject === 'object' && !Array.isArray(jsonObject)) {
      return (
        <TreeItem key={path} nodeId={path} label={getLabel(path)}
          onClick={() => handleItemClick('-', path)}>
          {Object.entries(jsonObject)
            .filter(([subValue]) => subValue !== null && subValue !== undefined)
            .map(([subKey, subValue]) => {
              const newPath = `${path}.${subKey}`;
              const renderedSubtree = renderTree(subValue,
```

```
                                                newPath,
                                                currentLevelCount);
            currentLevelCount += 1;
            return renderedSubtree;
          })}
      </TreeItem>
    );
  } else if (Array.isArray(jsonObject)) {
    return (
      <TreeItem key={path} nodeId={path} label={getLabel(path)}
        onClick={() => handleItemClick('-', path)}>
        {jsonObject.slice(0, maxTreeItems).map((item, index) => {
          const newPath = `${path}[${index}]`;
          const renderedSubtree = renderTree(item, newPath, currentLevelCount);
          currentLevelCount += 1;
          return renderedSubtree;
        })}
      </TreeItem>
    );
  } else {
    return (
      <TreeItem key={path} nodeId={path} label={getLabel(path)}
        onClick={() => handleItemClick(jsonObject, path)}>
      </TreeItem>
    );
  }
};
```

Diese Funktion ruft sich selbst so lange auf, bis das Element kein Objekt oder Array enthält und fügt die Elemente der Baumstruktur hinzu. Während der Entwicklung wurde festgestellt, dass die Baumstruktur bei sehr grossen JSON-Dateien (ca. ab 5 MB) langsam wird. Um dieses Problem zu beseitigen wurde ein Zähler integriert, welcher die Elemente pro Ebene zählt und nach einer bestimmten Anzahl die Funktion abbricht.

3.2.4. Import und Export Funktion

Aktuell bietet das Alignment Tool keine Möglichkeit das Ergebnis zu exportieren, um es zu einem späteren Zeitpunkt wieder zu verwenden. Da das Verarbeiten von grossen Datensätzen mehrere Minuten in Anspruch nehmen kann, ist die Integration einer solchen Funktion für ein effizientes Arbeiten mit der Applikation entscheidend.

Export

Die Exportfunktion hat zum Ziel, die vom Backendserver verarbeiteten Daten in eine Datei zu konvertieren, welche innerhalb des Tools heruntergeladen werden kann.
Da das Backend das Ergebnis im JSON-Format an die Applikation übermittelt, soll dieser JSON-String in Form einer Datei zur Verfügung gestellt werden. Um eine Verwechslung zwischen den ursprünglichen Rohdaten und den verarbeiteten Ergebnissen zu vermeiden, wird die Dateiendung für die exportierbare Datei auf STT geändert. Durch diese spezifische Dateiendung wird es der Frontendapplikation ermöglicht, automatisch zu erkennen, ob es sich um Rohdaten oder um ein Ergebnis handelt.
Die Implementierung dieser Funktionalität wurde direkt in die **AlignmentComponent** Komponente integriert, welche sich in der Datei **/alignment/index.tsx** befindet.

Import
Der Import der STT-Datei geschieht direkt auf der Startseite. Der geparste Inhalt der Datei wird direkt der Funktion **getAlignment** aus der create_context Datei weitergereicht. Dadurch wird dieselbe Aktion ausgeführt wie beim Empfangen der Resultate vom Backendserver.

3.2.5. Information- und Hilfeseiten

Für Hilfe und Informationsseiten wurde dem Projekt eine neue Komponente «info» hinzugefügt. Diese unterstützenden Seiten sind relativ einfach aufgebaut und enthalten hauptsächlich HTML-Elemente. Deshalb wurden diese in derselben **index.tsx** Datei als separate Funktionen integriert. Anschliessend wurden die Pfade in der Haupt React Datei referenziert und der Navigationsbar hinzugefügt.

Um zusätzliche Hilfeleistungen und Informationen innerhalb des Alignment Tools zur Verfügung zu stellen, wurde dem Projekt eine neue Komponente mit dem Namen «info» hinzugefügt. Diese Komponente ist für die Bereitstellung von Hilfe- und Informationsseiten gedacht. Sie beinhaltet eine «Help» Seite, auf welcher Fragen zur Applikation beantwortet werden. Die zweite Seite ist die «About» Seite. Diese dient als Impressum. Diese Informationsseiten bestehen hauptsächlich aus HTML-Elementen. Pro Hilfe -oder Informationsseite wurde eine Funktion innerhalb der entsprechenden **index.tsx** Datei angelegt.
Nach der Erstellung dieser Funktionen wurden sie in der **main.tsx** Datei referenziert und dem React Router hinzugefügt. Zudem wurden sie innerhalb der Navigationsbar verlinkt.

3.3. Eingesetzte Tools

Für die Versionskontrolle des Quellcodes und die Zusammenarbeit am Projekt wird Github eingesetzt. Dies bietet den Vorteil, dass jeder Projektbeteiligte einen eigenen Fork des Mastercodes erstellen kann und die Änderungen nach der Finalisierung zurück an den Master senden kann. Als Entwicklungsumgebung wurde Visual Studio Code genutzt. Dieses Tool eignet sich besonders für das Programmieren in diversen Sprachen, da es durch Extensions erweitert werden kann, welche den Arbeitsprozess effizierter gestalten. Während der Entwicklung wurde das Backend in einem Docker Container gestartet, damit die Funktionalitäten der React Applikation laufend getestet werden konnten.

4. Resultate

In diesem Kapitel werden die Resultate der Weiterentwicklung des Alignment Tools vorgestellt. Das Hauptziel dieser Projektarbeit lag in der Verbesserung der User Experience der Applikation. Zur Veranschaulichung der erreichten Verbesserungen wird eine Reihe von Screenshots der finalen Applikation präsentiert. Diese Darstellungen dienen dazu, die wesentlichen Neuerungen und Anpassungen zu illustrieren.

4.1. User Interface

4.1.1. Startseite

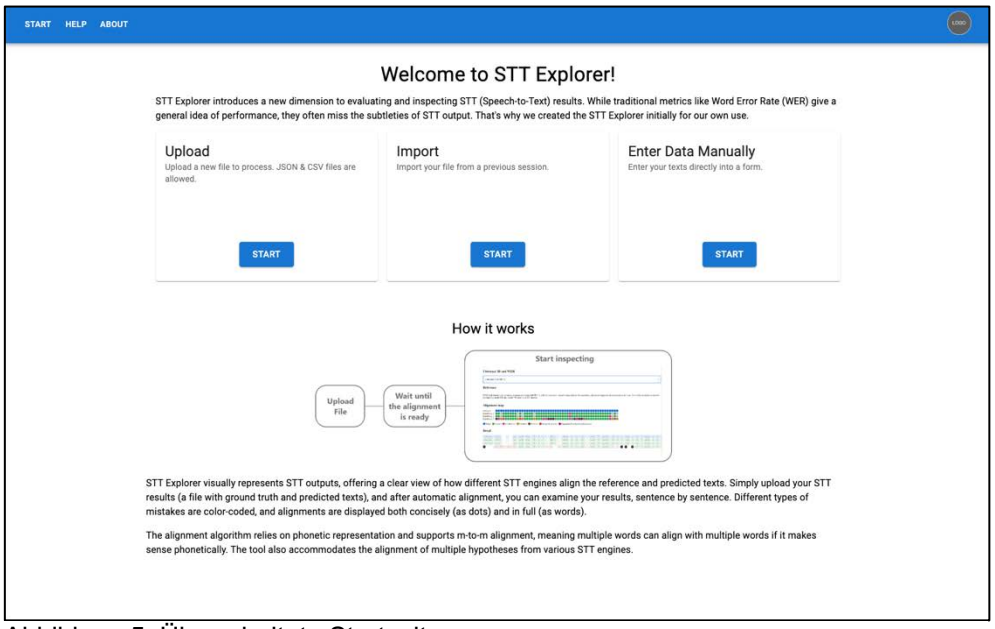

Abbildung 5: Überarbeitete Startseite

Der Startseite und allen weiteren Seiten wurde eine Menüleiste hinzugefügt (siehe Abbildung 5). Diese Leiste beinhaltet derzeit drei Hauptoptionen: «Start», «Help» und «About». In der rechten oberen Ecke ist Platz für ein zukünftiges Logo der Applikation. Bei der Gestaltung wurde bewusst klares Design geachtet, um den Nutzern eine schnelle Orientierung zu ermöglichen. Ausserdem bietet die Menüleiste ausreichend Raum für zukünftige Funktionen.

Ein wichtiger Aspekt der Startseite ist die Integration einer kurzen Projektbeschreibung sowie einer Beschreibung der Hauptfunktionen. Durch diese Informationen sind der Zweck und Nutzen der Seite unmittelbar nach dem Aufrufen ersichtlich. Standardmässig findet der Anwender die Optionen zum Dateiupload, Import eins existierenden Resultates oder der manuellen Eingabe neuer Texte. Diese drei Funktionen wurden zentral auf der Seite platziert. Ausserdem bieten diese drei einheitlichen Boxen ausreichend Platz für eine Kurzbeschreibung.

Sobald die gewünschte Startfunktion ausgewählt ist, erscheint ein Fenster, in dem die benötigten Daten abgefragt werden. Beim Upload und Import ist das Heraufladen einer Datei erforderlich (siehe Abbildung 6). Wenn die Option «Enter Data Manually» ausgewählt wurde, kann direkt die Anzahl der zu verarbeitenden Äusserungen und die Anzahl der Hypothesen definiert werden (siehe Abbildung 7).

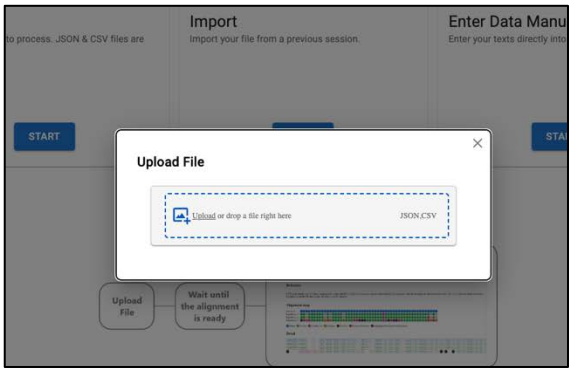

Abbildung 6: Dateiupload Dialog für neue Dateien

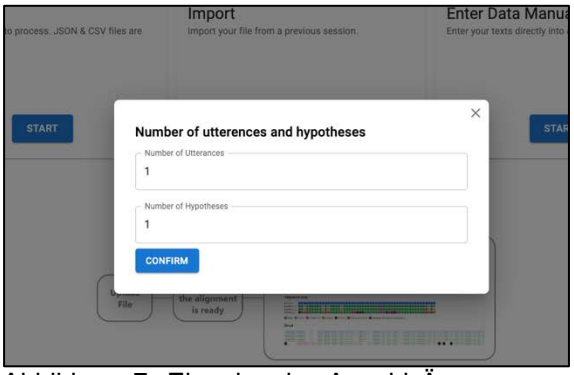

Abbildung 7: Eingabe der Anzahl Äusserungen und Hypothesen

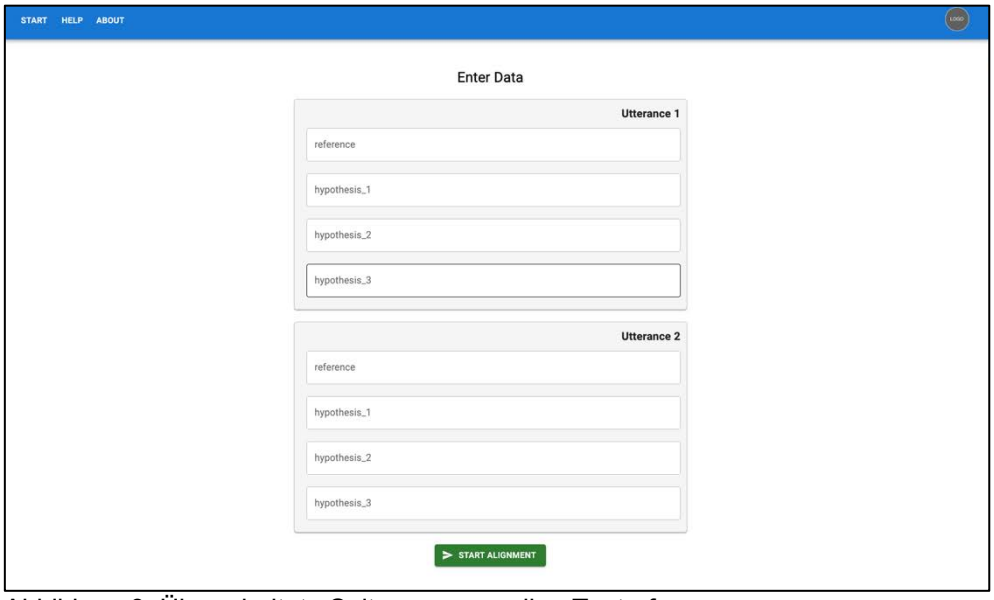

Abbildung 8: Überarbeitete Seite zur manuellen Texterfassung

Durch die Definierung der Text- und Hypothesenanzahl auf der Startseite konnte auf der Eingabeseite die Anzahl Buttons stark reduziert werden (siehe Abbildung 8). Dadurch sinkt das Risiko einer falschen Eingabe durch den Nutzer.

4.1.2. Hilfeseite

Die Hilfeseite der Applikation (siehe Abbildung 9) ist als Q&A Bereich (Questions & Answers) gestaltet und bietet den Nutzern eine Sammlung von häufig gestellten Fragen. Diese sind in einer Liste organisiert, wobei jede Frage ausklappbar ist, um die dazugehörige Antwort anzuzeigen. Die ausklappbare Struktur sorgt für eine übersichtliche Darstellung und vermeiden gleichzeitig, dass zu viele Informationen auf einmal angezeigt werden.
Die Integration dieses Q&A Bereichs zielt darauf ab, den Nutzern eine Selbsthilfe Ressource zur Verfügung zu stellen, in der sie Antworten auf die Funktionalitäten des Tools finden können. Des Weiteren können Beispieldateien heruntergeladen werden, um das Tool zu testen.

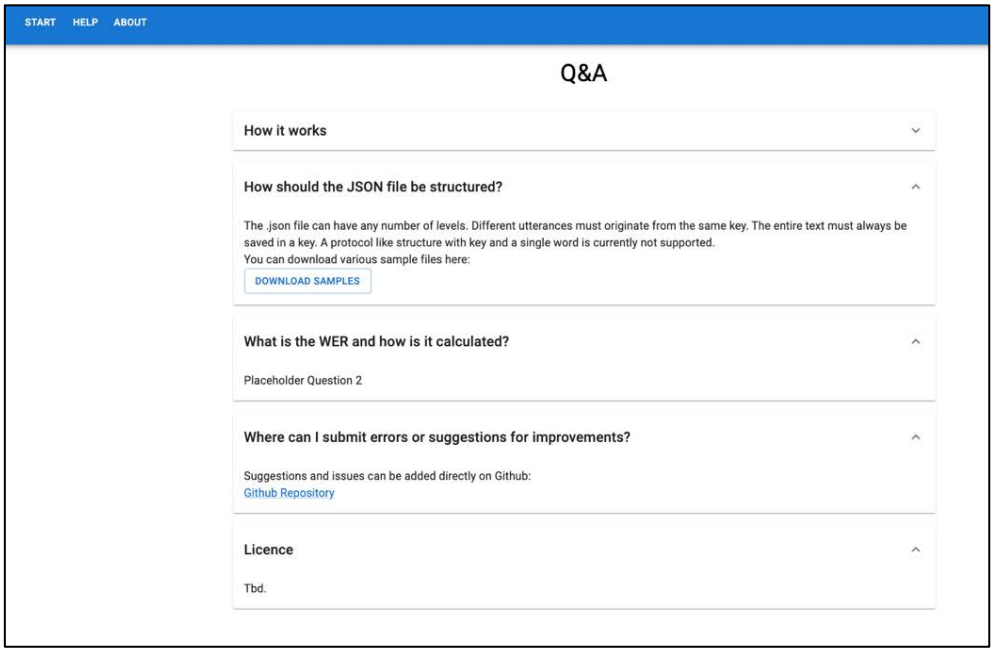

Abbildung 9: neue Q&A Seite

4.1.3. Auswahl der Texte nach dem Dateiupload

Die Seite zur Auswahl der Referenz und Hypothesenpfaden für die Verarbeitung der Dateien wurde komplett neugestaltet (siehe Abbildung 10). Durch den Einsatz eines zwei Spalten Layouts mit der Baumstruktur auf der linken Seite und den Informationen auf der rechten Seite sind immer alle Informationen auf ersichtlich. Dadurch wird ein unnötiges hin- und herspringen zwischen den Arbeitsschritten verhindert.
Im oberen Bereich befindet sich eine kurze Anweisung, in welcher dem Nutzer erklärt wird, was er auswählen muss. Diese passt sich dynamisch dem aktuellen Arbeitsschritt an. Auch der Button (blau) zur Bestätigung der Auswahl ändert seinen Text, damit dem Anwender immer ersichtlich ist, welche Aktion ausgeführt wird. Der «Start Alignment Button ist bis zum letzten Schritt nicht auswählbar. Dadurch wird das Absenden verhindert, bis alle Informationen korrekt ausgewählt sind. Durch einen Klick auf das Stiftsymbol kann die gespeicherte Eingabe bearbeitet werden und mithilfe des Mülleimersymbols können falsch ausgewählte Hypothesen wieder entfernt werden.

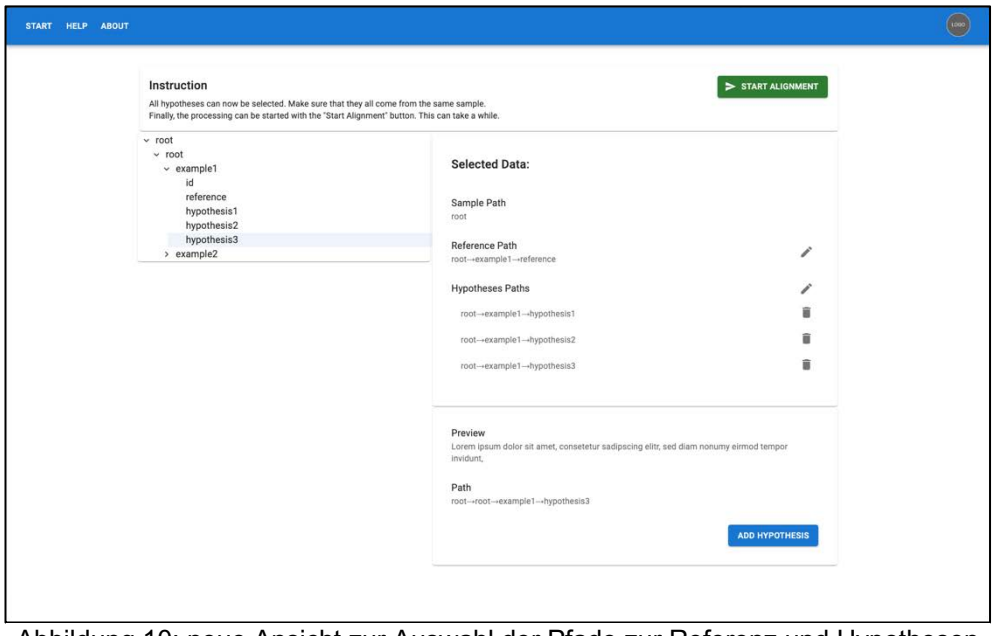

Abbildung 10: neue Ansicht zur Auswahl der Pfade zur Referenz und Hypothesen

4.1.4. Visualisierung der Resultate

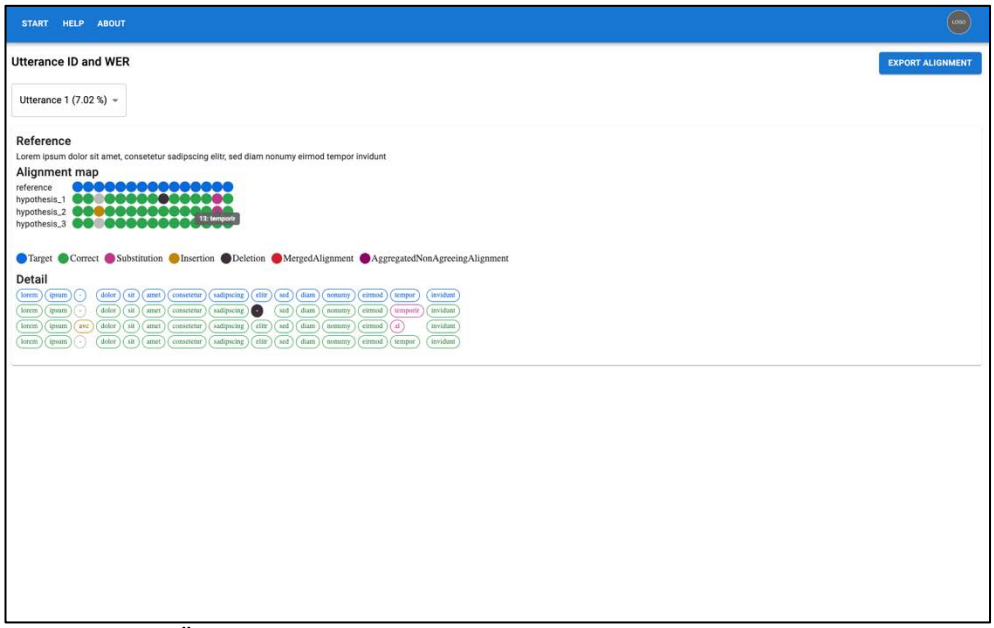

Abbildung 11: Überarbeitetes Layout des Resultats inkl. Export Button

Der Visualisierungsseite wurde ein Export Button hinzugefügt (siehe Abbildung 11). Dadurch kann der Nutzer das Ergebnis durch einen Klick herunterladen und später wieder auf der Startseite importieren. Dadurch kann die Referenz- und Hypothesenauswahl übersprungen werden. Durch das hinzufügen einer Box um den unteren Teil, ist das Resultat eindeutiger vom Dropdown abgegrenzt. Des Weiteren wurden diverse Fehler auf dieser Seite entfernt.
Die Alignment Map und die Detailansicht wird nun in einer Box angezeigt, welche horizontal scrollbar ist. Dadurch wird verhindert, dass die Seite bei langen Texten zu breit wird. Durch das Hinzufügen eines Tooltips in der Alignment Map, wird dem Benutzer das entsprechende Wort inklusive des Index angezeigt. Durch diese kleine Optimierung muss das fehlerhafte Wort nicht mehr in der Detailansicht gesucht werden.

4.2. Github Dokumentation

Im Rahmen dieser Projektarbeit wurde eine umfassende Dokumentation des Alignment Tools erarbeitet. Diese Dokumentation bietet einen Einblick in die verwendeten Frameworks und Bibliotheken.
Ein wesentlicher Bestandteil ist ausserdem eine umfangreiche Anleitung, welche Schritt für Schritt den Prozess der Einrichtung einer lokalen Entwicklungsumgebung beschreibt. Diese Anleitung ist besonders für neue Projektteilnehmer nützlich, da sie den Einstieg in die Projektmitarbeit erleichtern soll.
Zusätzlich zur Anleitung der Einrichtung umfasst die Dokumentation eine detaillierte Darstellung der Ordnerstruktur des Projekts. Zudem enthält die GitHub Dokumentation Beispiele und Codeausschnitte, welche bei der Weiterentwicklung von Nutzen sein können.
Es ist elementar, dass diese Dokumentation kontinuierlich aktualisiert und erweitert wird.

5. Diskussion und Ausblick

Das Ziel dieser Arbeit war die Analyse der bestehenden Applikation und die Optimierung des Frontends. Dadurch sollte die User Experience merkbar erhöht und allfällige Fehler der Applikation entfernt werden. Dadurch soll eine zuverlässige Anwendung geschaffen werden.

Anhand der dargelegten Resultate ist eine deutliche Verbesserung der Benutzerfreundlichkeit und eine Erweiterung des Funktionsumfangs zu erkennen. Durch die übersichtlich gestaltete Startseite, der hinzugefügten Navigationsbar und den überarbeiteten Dialogen zur Eingabe und Textauswahl wird dem Nutzer ein verständliches Tool zur Verfügung gestellt.
Durch die Neugestaltung des Dialogs zur Auswahl der Referenz- und Hypothesenpfade lassen sich diese effizienter auswählen und falsche interaktionsanreize wurden weitgehend entfernt.
Mithilfe der Import- und Exportfunktionalität wird dem Nutzer ein mehrfaches Untersuchen der Ergebnisse ohne ein erneutes Ausführen der Berechnungen ermöglicht. Dadurch wird die Belastung des Servers reduziert und die Wartezeit für den Anwender beim Import eliminiert.

Die Funktionsbeschreibungen und die Q&A Sektion sollten laufend nach einem allfälligen Feedback der Nutzer aktualisiert werden, um die letzten Unklarheiten bei der Bedienung zu beseitigen.

Durch die Github Dokumentation wird zukünftigen Projektteilnehmern der Einstieg in die Entwicklung des Alignment Tools vereinfacht. In Zukunft sollte in Github das Issue Tracking verwendet werden, damit auch die Behebung kleinerer Fehler nicht vergessen geht.

Trotz dieses Fortschritts sollte die Applikation im Rahmen zukünftiger Projekte um zusätzliche Funktionen erweitert werden. Dadurch kann der Anwendungsbereich vergrössert und der Nutzen für die Anwender erhöht werden.
Die Ergebnisseite würde sich beispielsweise um zusätzliche Grafiken und Kennzahlen erweitern lassen. Momentan muss jede einzelne Utterance separat untersucht werden. Für den Fall, dass dieselbe Hypothesennummern innerhalb eines Corpus immer aus demselben Modell stammen, wäre eine Grafik, welche die durchschnittliche WER berechnet, hilfreich. Dadurch könnte das beste Modell schneller selektiert werden. Um die Zuverlässigkeit der WER zu erhöhen, sollte das Tool mit Daten aus verschiedenen Quellen getestet werden, um allfällige Schwachstellen zu identifizieren und zu beseitigen.
Ein weiterer Verbesserungsvorschlag wäre die Implementierung weiterer Kennzahlen zusätzlich zur Word Error Rate.
Wünschenswert wäre ausserdem die automatische Erkennung der Referenz- und Hypothesenpfade innerhalb der Applikation. Dies ist jedoch durch die freie Struktur der Dateien nicht leicht umzusetzen. Ausserdem wäre es wünschenswert, dass auch Transkriptionen mit Zeitstempel verarbeitet werden könnten.

Ein Ausbau der Applikation zu einem Portal, auf welchem sich die Nutzer anmelden, ihre Resultate abspeichern und wiederverwenden können würde ein weiterer Funktionsausbau ermöglichen. Dadurch könnte man beispielsweise anhand eines vordefinierten Audiodatensatzes die Modelle aller Nutzer untereinander vergleichen und schlussendlich das beste Modell identifizieren.

Die Optimierung der Applikation brachte viele Herausforderungen mit sich. Da der Quellcode bereits zum Projektbeginn sehr umfangreich und die Dokumentation verbesserungsfähig war, gestaltete sich der Einstieg in den Programmcode schwierig.
Durch den Unterschied zwischen den Mockups und der realisierten Applikation ist ersichtlich, wie sich das Tool während der Zeit der Projektarbeit verändert hat. Selbst kleinere Anpassungen am Userinterface wurden immer wieder hinterfragt und überarbeitet. Dadurch wurde sicherlich das bestmögliche Ergebnis der Benutzeroberfläche erreicht. Dies führte jedoch zu grösseren Zeitverlusten während der Entwicklung. Im Nachhinein wäre es wahrscheinlich

sinnvoller gewesen, wenn das eine oder andere Detail nicht im Rahmen der Projektarbeit umgesetzt worden wäre. Stattdessen hätte etwas mehr Zeit in die Erweiterung der Funktionen investiert werden können. Dadurch wäre für Aussenstehende der Fortschritt besser ersichtlich. Insgesamt hat sich der Aufwand für die übersichtliche Benutzeroberfläche jedoch gelohnt und es konnte eine solide Grundlage für zukünftige Entwicklungen geschaffen werden.

6. Verzeichnisse

6.1. Literaturverzeichnis

[1] «jitsi/jiwer: Evaluate your speech-to-text system with similarity measures such as word error rate (WER)». Zugegriffen: 22. Dezember 2023. [Online]. Verfügbar unter: https://github.com/jitsi/jiwer

[2] «Kensho Word Error Rate Calculator | Scribe». Zugegriffen: 21. Dezember 2023. [Online]. Verfügbar unter: https://kensho.com/scribe/wer-calculator

[3] «React». Zugegriffen: 22. Dezember 2023. [Online]. Verfügbar unter: https://react.dev/

[4] B. Krebs und M. Juan Cruz, «Developing RESTful APIs with Python and Flask». Zugegriffen: 22. Dezember 2023. [Online]. Verfügbar unter: https://auth0.com/blog/developing-restful-apis-with-python-and-flask/

[5] N. Rajvanshi, «TypeScript vs JavaScript: Which Is Worthier? (Simplified Guide)», Ace Infoway. Zugegriffen: 22. Dezember 2023. [Online]. Verfügbar unter: https://www.aceinfoway.com/blog/typescript-vs-javascript

[6] «mlynckat/alignment-app». Zugegriffen: 22. Dezember 2023. [Online]. Verfügbar unter: https://github.com/mlynckat/alignment-app

[7] F. Chiusano, «Two minutes NLP — Intro to Word Error Rate (WER) for Speech-to-Text», NLPlanet. Zugegriffen: 22. Dezember 2023. [Online]. Verfügbar unter: https://medium.com/nlplanet/two-minutes-nlp-intro-to-word-error-rate-wer-for-speech-to-text-fc17a98003ea

[8] «React | Meta Open Source». Zugegriffen: 22. Dezember 2023. [Online]. Verfügbar unter: https://opensource.fb.com/projects/react/

[9] «Vite». Zugegriffen: 22. Dezember 2023. [Online]. Verfügbar unter: https://vitejs.dev

[10] «MUI: Move faster with intuitive React UI tools». Zugegriffen: 22. Dezember 2023. [Online]. Verfügbar unter: https://mui.com/

[11] «App Bar React component - Material UI». Zugegriffen: 22. Dezember 2023. [Online]. Verfügbar unter: https://mui.com/material-ui/react-app-bar/

[12] «Tree View React component - MUI X». Zugegriffen: 22. Dezember 2023. [Online]. Verfügbar unter: https://mui.com/x/react-tree-view/

6.2. Abbildungsverzeichnis

6.3. Tabellenverzeichnis

7. Anhang

Phonetic Alignment and Visualization App

This is a web application built with React TypeScript on the frontend and Python on the backend. The goal of the app is to phonetically align and visualize the alignment between a reference text and one or more hypotheses texts, which represent Speech-to-Text transcriptions from different systems. The alignment algorithm used in this project is taken as it is from ZHAW CAI, and the visualization of the alignment has been translated and rewritten from a previous Angular project.

The app is currently running on https://srv-lab-t-568.zhaw.ch/

Table of Contents

Features

- Supports JSON and CSV files with different structures for input data.
- Allows manual copy & paste input of the hypotheses.
- Supports one or more hypotheses for each reference text.
- Supports Export & Import of Alignment Results
- Proper error handling is currently lacking and will be addressed in future updates.
- Currently supports only JSON and CSV formats; other formats are not supported yet.
- Sometimes, the alignment may crash, but it will be fixed in the next update.

Used Frameworks & Libraries

name	where	for what
React	Frontend	main javascript library --> code is written in Typescript
Vite	Frontend	Frontend-Build-Tool

name	where	for what
MUI	Frontend	React component library -> used for UI components
nginx	Frontend	Webserver for production
Flask	Backend	Python Webframework
gunicorn	Backend	WSGI HTTP Server --> used for production

Requirements

- JSON files should have a path that represents all samples, under which reference and hypotheses texts can be found.
- When manually entering data, multiple utterances should have the same number of hypotheses.
- When selecting the path to reference and/or hypotheses' texts, the full path to the actual text should be given (i.e., it is a path to the corresponding text of one (any) sample).
- When uploading CSV files, the path to the samples should always be set as "root."

Production

The app is currently running on https://srv-lab-t-568.zhaw.ch/ in production. In order for it to run on https://, nginx was introduced. The approximate schema of the connections looks like follows: nginx

Development

In order to run it in development, the two services frontend and backend should be started separately:

Start Backend without Docker

First you have to navigate to the backend folder and install all required Python packages inside a virtual enviroment.

Create virtual envrionment and install packages:
Make sure that you are running python 3.8.8 on your computer

```
python3.8 -m venv backend/.flask-env
source backend/.flask-env/bin/activate
pip install -r  backend/requirements.txt
```

Start flask backend server:

```
python3.8 backend/app.py
```

After that the server should be started on localhost with port 8000

Start Frontend without Docker

Make sure you have npm installed on your computer. This application is tested with npm V10.2.0

Check npm version

```
npm -v
```

Install all dependencies:

```
cd alignment_tool_app
npm install
```

Start Frontend Application with live preview

```
cd alignment_tool_app
npm run dev
```

If you want to use the local Backend Server you have to adjust some urls in the project:
Replace the https://xxx url with http://localhost:8000

- package.json
- src/main.tsx

Build local Docker Container

Since the nginx server configuration in this project is configured for the production server with SSL encryption, we must first adjust some files.

1. Open /aligment_tool_app/nginx.conf file and replace the code with the following:

```
# Set the number of worker processes to match the number of CPU cores
worker_processes 4;

# Define global event settings
events {
    worker_connections 1024;  # Maximum number of simultaneous connections
per worker process
}

# Define HTTP server configuration
http {
    # Set the maximum client request body size to 100MB
    client_max_body_size 100M;

    access_log /var/log/nginx/access.log;      # Access log file
    error_log /var/log/nginx/error.log;        # Error log file
```

```
    # Define the first server block (HTTP to HTTPS redirection)
    server {
        listen 80;  # Listen on port 80 (HTTP)
        server_name _;  # Catch-all server_name (matches any hostname)

        # Define the root directory for serving static content
        location / {
            root /usr/share/nginx/html;  # Root directory for HTML files
            index index.html;  # Default index file
            client_max_body_size 100M;

            # Proxy headers for the application
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        }

        # Define a location for proxying requests to the /api endpoint
        location /api {
            proxy_pass http://localhost:8000;  # Proxy requests to the
specified backend server
            proxy_read_timeout 3600s;  # Set the proxy read timeout to
3600 seconds (1 hour)
        }
        # Enable gzip compression for text-based content (additional
optimization)
        gzip on;
        gzip_types text/plain text/css application/json
application/javascript text/xml application/xml application/xml+rss
text/javascript;

        # Include MIME types for content negotiation
        include /etc/nginx/mime.types;

        server_name _;  # Catch-all server_name (matches any hostname)
    }
}
```

This will deactivate the https redirection and replace the public proxy with a local.

2. Replace https://xxx urls with http://localhost:8000 if you want to use a local Backend Server. For this you have to edit the following files:

- alignment_tool_app/package.json
- alignment_tool_app/src/main.tsx

3. Remove the following lines from the docker-compose.yml file:

```
version: '3'

services:
  backend:
```

```
    restart: always
    build: backend
    command: [ "gunicorn","-w","8", "--timeout", "0", "--access-logfile",
"/usr/src/app/backend/logfile.log", "--error-logfile",
"/usr/src/app/backend/errorfile.log", "-b","0.0.0.0:8000","app:app"]
    volumes:
      - ./backend:/backend
    environment:
      - FLASK_ENV=production
      - FLASK_APP=app.py
    ports:
      - 8000:8000
    env_file:
      - backend/secrets.env

  frontend-nginx:
    container_name: ngixreactapp
    build: alignment_tool_app
    ports:
      - 80:80
      #- 443:443                                          <-- Remove this
line
    environment:
      NODE_ENV: production

    volumes:
      - ./alignment_tool_app:/alignment_tool_app
      - node-modules:/alignment_tool_app/node_modules
      #- /etc/letsencrypt:/etc/nginx/certs                <-- Remove this
line

    depends_on:
      - backend

volumes:
  node-modules:
```

4. Build container:

First, navigate to the project root dir

```
docker compose-build --no-cache
```

5. Start docker container:

First, navigate to the project root dir

```
docker compose up
```

Now, the React frontend Application should be available on http://localhost:80 and the backend is running on http://localhost:8000

React Frontend

All Frontend related files are localed in **/alignment_tool_app**

```
. alignment_tool_app
├─ node_modules
│  └─ ...
├─ src
│  ├─ features                    <-- root for all components
│  │  ├─ assets
│  │  │  └─ ...                   <-- icons, pictures for website
│  │  ├─ alignment
│  │  │  ├─ create_content.tsx
│  │  │  ├─ index.style.ts
│  │  │  ├─ index.tsx             <-- index.tsx includes final export
│  │  │  └─ model.ts
│  │  ├─ fileUpload               <-- one folder per location / function
│  │  │  ├─ index.style.ts
│  │  │  ├─ index.tsx
│  │  │  ├─ inputForm.tsx
│  │  │  ├─ jsonTreeView.tsx
│  │  │  ├─ parse_json.tsx
│  │  │  └─ wizzard.tsx
│  │  ├─ info
│  │  │  └─ index.tsx
│  │  ├─ navbar
│  │  │  └─ index.tsx
│  │  └─ selectedColumn
│  │     └─ index.tsx
│  ├─ main.tsx                    <-- main react file
│  └─ vite-env.d.ts
├─ Dockerfile
├─ index.html
├─ nginx.conf
├─ package-lock.json
├─ package.json
├─ tsconfig.json
├─ tsconfig.node.json
└─ vite.config.ts
```

The individual pages such as "Upload" or the results page are a single React component, just like the navbar. To prevent a component from becoming too large, parts of it can be outsourced to separate files and then imported back into the main "index.tsx" file. A single function of the tool always has its own folder. However, the index.tsx file should contain the code that is assigned to the page in the router.

For the interaction within a component, such as the file upload, conditional state rendering was usually used. This prevents intermediate loading of the components while working.

In order to guarantee a uniform appearance, components from the MUI library are always used whenever possible.

main.tsx - add new commponent

The main.tsx is the main file of the React project. This is where the router is initialised, which assigns the corresponding component to the various url locations. If an additional function is added, the React component must be imported here from the corresponding index.tsx file and assigned to a path.

The navigation bar is always fixed at the top of the page and does not need to be added to each component.

```
import React from "react";
import ReactDOM from "react-dom/client";
import { BrowserRouter, Route, Routes } from "react-router-dom";
import { FileUpload } from "@/features/fileUpload";
import { FileReaderContextProvider } from
"@/features/alignment/create_context";
import { AlignmentComponent } from "@/features/alignment";
import { AboutComponent, HelpComponent } from "@/features/info";
import Navbar from "@/features/navbar";
import "style.css";

import { NewComponent } from "@/features/example";

export const BACKEND_URL = "https://srv-lab-t-568.zhaw.ch"


const App: React.FC = () => {
  return (
    <React.StrictMode>
      <BrowserRouter>
        <Navbar />
        <FileReaderContextProvider>
          <Routes>
            <Route path="*" element ={<div>Page not found</div>} />
            <Route path="/" element={<FileUpload />} />
            <Route path="/visualize" element={<AlignmentComponent />} />
            <Route path="/about" element={<AboutComponent />} />
            <Route path="/help" element={<HelpComponent />} />
            <Route path="/new_example_component_path" element=
{<NewComponent />} />
          </Routes>
        </FileReaderContextProvider>
      </BrowserRouter>
    </React.StrictMode>
  );
};

ReactDOM.createRoot(document.getElementById("root") as
HTMLElement).render(
```

```
    <App />
);
```

Navbar

The navbar is used to navigate between the different locations. It has a very simple structure with buttons that link to the corresponding locations of the router from the main.tsx. The logo is currently used as a placeholder and will be replaced by a real one in the future.

add new Button to the navbar
To add a new page to the navbar, the source code of the index.tsx from the navbar order must be extended by a button with the corresponding path (from the React router).

```
<Button color="inherit" onClick={() => handleNavigation("/help")}>
    Help
</Button>
```

info

The "info" component contains smaller static pages, such as the About or Help page.

fileUpload

The fileUpload component contains the start page with the instructions and the selection of the input. The index.html contains the basic content of the start page and the upload dialogues. After a file has been uploaded, it is assigned to the corresponding component. Further processing then takes place in the sub-component (e.g. jsonTreeView for .json files).

There are basically four different input options, which are located behind three options.

1. Upload of new files (.json or .csv) ***Files:*** for JSON: index.tsx, jsonTreeView.tsx, for CSV: index.tsx, parse_json.tsx, wizzard.tsx
2. Import of an exported file from a previous processing. ***Files:*** index.tsx
3. Manual text input ***Files:*** index.tsx, inputForm.tsx

1. Upload of new files
If a .json file is uploaded, the jsonTreeView is called for further processing. If a .csv file is uploaded, further processing is carried out with the HorizontalLinearStepper from the wizzard.tsx. This was deliberately split up so that a different dialogue can be used to select the data from .csv.

The user must then select the path of the samples, the reference text and the hypotheses.

Finally, the file and the paths are sent to the server in the following format:

```
const sendData = async () => {
  const samplesPath = confirmedData.samplePath;
  const content = new FormData();
  content.append("file", fileToSend)
```

```
  content.append("selection", JSON.stringify(selection));
  content.append("samples", JSON.stringify(smaples_path));
  setIsLoading(true);
  return await axios({
    method: "post",
    url: `${BACKEND_URL}/api/upload_file`,
    data: content,
    headers: {
      Accept: "application/json",
      "Content-Type": "multipart/form-data",
    },
  })
  .then((response) => {
    setIsLoading(false);
    return response.data;
  })
  .catch((error) => {
    setIsLoading(false);
    console.log(error);
  });
};
```

The backend location for the request is /api/upload_file
"file" contains the binary file, "selection" contains all paths to the references and hypotheses and "samples" contains the path to the samples:

```
file: (Binärdaten)
selection:
{"reference":"|root|example1|reference","hypothesis_1":"|root|example1|hyp
othesis1","hypothesis_2":"|root|example1|hypothesis2","hypothesis_3":"|roo
t|example1|hypothesis3"}
samples: "|root"
```

The paths must start and separated with "|".

2. Import of an exported file from a previous processing.
The result of the alignment can be exported. This is a .json file, which however has the file extension .STT. This has been adapted so that the React application can differ between a result and a new file.

As only the server response received after processing is saved during export, the same function can be called during import to display the result again.

```
const reader = new FileReader();
reader.onload = async () => {
  try {
    const importedData = JSON.parse(reader.result as string);
    getAlignment?.(importedData);
  } catch (error) {
```

```
      console.error("Error parse imported data:", error);
      setFileToSend(undefined);
    }
  };
  reader.readAsText(file);
```

Where "reader.result" is the content of the .STT file.

3. Manual text input

If the text is entered manually, it is sent to the backend server as follows:

```
const handleCopyPasteSubmit = async () => {
  const formData = new FormData();
  formData.append("utterances", JSON.stringify(utteranceData));
  return await axios({
    method: "post",
    url: `${BACKEND_URL}/api/send_utterances`,
    data: formData,
    headers: {
      Accept: "application/json",
      "Content-Type": "multipart/form-data",
    },
  })
    .then((response) => {
      getAlignment?.(response.data);
      return response.data;
    })
    .catch((error) => {
      console.log(error);
    });
};
```

The backend location for the texts is: /api/send_utterances

In addition, the entire content is sent as json formatted with the key "utterances":

```
utterances:[{"reference":"Lorem ipsum dolor sit amet, consetetur
sadipscing elitr, sed diam nonumy eirmod tempor
invidunt","hypothesis_1":"Lorem ipsum   dolor sit amet, consetetur
sadipscing, sed diam nonumy eirmod temporir
invidunt","hypothesis_2":"Lorem Ipsum [avc] dolor sit amet, consetetur
sadipscing elitr, sed diam nonumy eirmod al invidunt"},
{"reference":"Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed
diam nonumy eirmod tempor invidunt","hypothesis_1":"Lorem ipsum dolor sit
amet, consetetur sadipscing, sed diam nonumy eirmod temporir
invidunt","hypothesis_2":"Lorem ipsum dolor sit amet,    sadipscing elitr,
sed diam nonumy eirmod tempor lund invidunt,"}]
```

Import

Info | **Import** | Upload File | Manual Input | LOGO

Select or Drag & Drop your exported file

Upload or drop a file right here

Process

Import

Info | **Import** | Upload File | Manual Input | LOGO

Result

Upload File

Info | Import | **Upload File** | Manual Input | LOGO

Select or Drag & Drop your file

Upload or drop a file right here JSON,CSV

Start

Upload File

Info | Import | **Upload File** | Manual Input | LOGO

Select your Reference

Tree View

- execution
- samples
  - sample1
  - sample2
    - metadata
      - **sample text**
      - language
      - id
    - transcriptions

Ex consequat commodo adipisicing exercitation aute excepteur occaecat ullamco duis aliqua id magna ullamco eu. Do aute ipsum ipsum ullamco cillum consectetur ut et aute consectetur labore. Fugiat laborum incididunt tempor eu consequat enim dolore proident. Qui laborum do non excepteur nulla magna eiusmod consectetur in. Aliqua et aliqua officia quis et incididunt voluptate non anim reprehenderit adipisicing dolore ut consequat deserunt mollit dolore. Aliquip nulla enim veniam non fugiat id cupidatat nulla elit cupidatat commodo velit ut eiusmod cupidatat elit dolore.

Next

Upload File

Info | Import | **Upload File** | Manual Input | LOGO

Select your Hypothesis

Tree View Multi Select

sample2

metadata

transcriptions

Nuance

metrics

Text

Amazon

metrics

Text

Hypothesis 1:
Ex consequat commodo adipisicing exercitation aute excepteur occaecat ullamco duis aliqua id magna ullamco eu...
Hypothesis 2
Ex consequat commodo adipisicing exercitation aute excepteur occaecat ullamco duis aliqua id magna ullamco eu...

Back

Process

alignment

The alignment folder with the main component "AlignmentComponent" contains the functions for displaying the data received from the server.

index.tsx
The index.tsx file contains the AlignmentComponent component, which is responsible for displaying the stored data. The export function is also included in this file.

model.ts
Here the types and structure are defined for the different variables (e.g. words are contained in sentences). The color codes for color mapping are also defined in this file.

create_context.tsx
The createContent.tsx uses React's Context API to create the global context "FileReaderContext". This means that the data can directly be used in other components. The word error rate is also calculated here.

Flask Backend

```
tbd
```