Zürcher Hochschule
für Angewandte Wissenschaften

School of Engineering

CAI Centre for Artificial Intelligence

Bachelorarbeit (Informatik)

Ensemble Methods for Speech Recognition

Autoren	Lars Mosimann Ralph Scheu
Hauptbetreuung	Prof. Dr. Mark Cieliebak
Nebenbetreuung	Katsiaryna Mlynchyk
Datum	09.06.2022

Erklärung betreffend das selbstständige Verfassen einer Bachelorarbeit an der School of Engineering

Mit der Abgabe dieser Bachelorarbeit 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 Bachelorarbeit 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:	**Name Studierende:**
Weinfelden, 09.06.2022	Lars Mosimann
Balterswil, 09.06.2022	Ralph Scheu

Zusammenfassung

In den letzten Jahren haben Automatic Speech Recognition (ASR) Systeme grosse Fortschritte beim Transkribieren von Sprache zu Text gemacht. Für vorgelesene Sprache können moderne Systeme mit Word Error Rates (WER) von weniger als 2 % nahezu perfekte Transkripte erstellen. Dieselben Systeme haben jedoch Mühe, für spontane Sprache, wie Meetings oder Telefongespräche, lesbare Transkripte zu erstellen. In dieser Arbeit wird untersucht, ob die WER und somit Transkript-Qualität verbessert werden kann, indem die Hypothesen (Outputs) von mehreren ASR-Systemen kombiniert werden. Es wurden die drei besten Systeme aus der englischen Version vom CEASR-Korpus verwendet, um unterschiedliche Ensemble-Methoden zu implementieren. Diese Ensemble-Methoden wurden mit der WER und SemDist Metrik bewertet. Um die Hypothesen der Systeme vergleichbar zu machen, müssen sie im sogenannten Alignment einander angeglichen werden. Dazu wird eine Primärhypothese ausgewählt, an welche die verbleibenden Hypothesen jeweils in einem phonetischen Alignment-Paar angeglichen werden. In dieser Arbeit wurde der n-align Algorithmus entwickelt, welcher eine beliebige Anzahl von solchen Alignment-Paaren zu einem einzelnen Alignment kombiniert. Des Weiteren wurde das Alignment UI Tool implementiert, mit welchem das Alignment visualisiert und dessen Qualität schnell beurteilt werden kann. Die beste, implementierte Ensemble-Methode verwendet Gradient Boosting, um auf Wortbasis zu lernen, in welcher Situation welche Systeme am ehesten richtig liegen. Dadurch konnte die WER auf dem CEASR-Korpus um 5 % auf spontaner Sprache und um 1–2 % auf vorgelesener Sprache reduziert werden. Schlussendlich demonstriert diese Arbeit, dass Ensemble-Methoden für ASR Potenzial haben. Dennoch ist von einer produktiven Nutzung aus Kosten/Nutzen Gründen abzuraten. Da während dieser Arbeit vieles zum generellen Umfeld von Ensemble-Learning für ASR beobachtet und präsentiert wurde, dienen die Erkenntnisse aus dieser Arbeit als Grundlage für die Entwicklung eines weiteren, besseren Ensemble-Prozesses.

Abstract

In recent years, Automatic Speech Recognition (ASR) Systems have made significant progress for transcribing speech to text. For perfectly clear read-aloud speech, state-of-the-art systems can have Word Error Rates (WER) as low as 2 %. However, the same systems may also struggle to produce readable results for spontaneous speech, such as meetings or phone calls. This thesis explores the possibility of combining the hypotheses (outputs) of multiple ASR systems to improve the WER and thus the quality of the transcribed text. We used the three systems, with the lowest WER from the English version of the CEASR-Corpus, to implement several ensemble methods and evaluated them with WER and SemDist. To make the hypotheses of the systems comparable, they have to be aligned to each other. To achieve this, a primary hypothesis is chosen, to which each of the remaining auxiliary hypotheses is aligned into an alignment pair using phonetic codes. To combine these pairs into a single alignment, we developed the n-align algorithm. Furthermore, we implemented an Alignment UI tool to visualize alignments, intended to assess alignment quality quickly. Of the implemented ensemble methods, the best one uses gradient boosting to learn appropriate system choices on a word-basis. The method decreases the WER on the CEASR-Corpus by 5 % on spontaneous speech and by 1-2 % on read-aloud speech. We conclude that while ensemble methods for ASR show potential, the implemented methods are not suitable for practical usage due to a limited cost/benefit ratio. As we observed and presented numerous circumstances that shape the whole setting of ensemble learning for ASR, we propose using the results of this thesis as a basis for creating a newer and better ensemble process.

Vorwort

Im Rahmen dieser Arbeit konnten wir vieles in den Bereichen Machine Learning, Speech-to-Text und Sprachverarbeitung lernen. Diese anspruchsvolle Arbeit konfrontierte uns mit vielen Herausforderungen, welche nicht immer einfach zu lösen waren. Umso motivierender war somit das Überwinden dieser Hürden. Wir bedanken uns bei unseren Betreuenden, Prof. Dr. Mark Cieliebak und Katsiaryna Mlynchyk, welche uns mit ihren tollen Ideen und Inputs immer wieder neue Sichtweisen aufgezeigt und uns unterstützt haben. Ebenfalls bedanken wir uns bei Freunden und Familie für den Rückhalt und die Korrekturlesungen.

Inhaltsverzeichnis

1. Einleitung

1.1. Ausgangslage und Zielsetzung

Automatic Speech Recognition (ASR) findet heutzutage ein breites Anwendungsgebiet. Sprachassistenten interpretieren Befehle, Business-Kommunikationssoftware transkribiert Meetings oder Telefonanrufe werden automatisch beantwortet. Das Produkt *Interscriber* der Spinning-Bytes AG verwendet verschiedene Speech-to-Text Engines, um Transkripte für Meetings, Interviews und andere Audioaufnahmen zu erstellen [1]. Mit dem automatischen Transkribieren kann im Vergleich zur manuellen Transkription viel Zeit gespart werden. Unter optimalen Voraussetzungen, also klare Aussprache, keine Hintergrundgeräusche, und das Ablesen eines Skriptes, kann Interscriber Transkripte mit einer Fehlerrate von 3-5 % in einem Bruchteil der Zeit eines manuellen Transkribieren erstellen. Ist die Audioaufnahme jedoch durch Hintergrundgeräusche, ein schlechtes Aufnahmegerät, undeutliche Sprache oder andere Störfaktoren beeinträchtigt, so kann die Qualität eines Transkriptes stark darunter leiden oder sogar unleserlich werden. Dies erfordert dann manuelle, potenziell aufwendige Korrekturen im Transkript.
Die Fehlerrate des Transkriptes ist somit nicht in allen Fällen genug tief und soll verbessert werden. In anderen Bereichen der Sprachverarbeitung, etwa Sentiment-Analysis, können einzelne Systeme übertroffen werden, indem die Outputs von mehreren Systemen kombiniert werden. Inspiriert von diesem Ansatz befasst sich diese Arbeit damit, die Transkripte von mehreren Engines zu kombinieren, um ein neues, genaueres Transkript zu erhalten. Eine ZHAW interne Voranalyse [2], fortan STT-Ensemble-Learning - Analyse genannt, hat ergeben, dass die Qualität mit solchen «Ensemble-Methoden» durchaus verbessert werden könnte. Es wurden ausserdem schon mit einigen Ansätzen experimentiert und es konnte eine kleine Verbesserung von 1 % erreicht werden. Diese Arbeit baut auf diesen Erkenntnissen auf.

1.2. Interscriber

Interscriber ist das Flaggschiffprodukt der SpinningBytes AG [3]. Interscriber kann benutzt werden, um Transkripte von beliebigen Audioaufnahmen zu erstellen. Dazu werden diverse kommerzielle und proprietäre ASR-Systeme verwendet. In einem einfachen, aber mächtigen Editor können die Transkripte bearbeitet, verbessert und schlussendlich exportiert werden. Abbildung 1 zeigt den Editor von Interscriber. Eine Audiodatei wurde hochgeladen und transkribiert, bevor dem Benutzer diese Anzeige präsentiert wird. Der Benutzer kann nun die Audiodatei anhören und das Transkript bearbeiten. Die aktuelle Stelle wird ihm im Text in der Hauptansicht angezeigt.

Abbildung 1.: Der Transkripteditor in Interscriber

1.3. Aufgabenstellung

In der Projektausschreibung sind vier Aufgaben formuliert.

- Definieren einer Methode, um mehrere Transkripte auf Wortebene zu vergleichen.
- Unterschiede zwischen Transkripten auf Wortebene untersuchen.
- Entwicklung eines Softwaretools, welches ein optimiertes Transkript aus den Outputs von mehreren ASR-Systemen generiert.
- generieren von optimierten Transkripten für eine vorgegebene Auswahl von Utterances und Berechnung / Ausgabe derer Qualität.

Anfangs war unklar, wo die Arbeit ihre Schwerpunkte setzen soll. Die Aufgabenstellung wurde nach und nach deutlicher. Es stellte sich schnell heraus, dass all diese Punkte bereits in unterschiedlichem Detailgrad in der STT-Ensemble-Learning - Analyse angegangen wurden. Nach und nach zeichneten sich zwei Hauptaufgaben ab. Zum einen ist der Vergleich zwischen den Transkripten (Ziel 1) mit dem sogenannten Alignment ein Hauptaspekt dieser Arbeit. Das Alignment ist aktuell schwer zu beurteilen, was diese Arbeit vereinfachen soll. Ausserdem ist das Alignment in seiner Implementation auf drei Transkripte limitiert, was ebenfalls aufgehoben werden soll. Zum anderen ist das eigentliche Entwickeln von Ensemble-Methoden ein zweiter Hauptaspekt. Messungen mit

Standardmetriken haben ergeben, dass es Potenzial gibt. In dieser Arbeit werden deshalb bekannte und neue Ensemble-Methoden vorgestellt, in einem Softwaretool (Ziel 3 und 4) implementiert und verglichen.

1.4. Übersicht und Aufbau

Im Kapitel 2 werden wichtige Grundlagen zu den Themen Ensemble Learning und ASR erläutert. Ausserdem werden bestehende Lösungen im Bereich Ensemble-Learning für ASR in einer Literaturrecherche abgehandelt. Im Kapitel 3 wird die Ausgangslage und Eingrenzung der Arbeit beschrieben. Als Nächstes werden im Kapitel 4 wichtige Insights und Grundlagen für das Entwickeln von Ensemble-Methoden generiert. Das Alignment wird im Detail analysiert und wo nötig erweitert. Danach folgt eine Analyse über die theoretisch bestmögliche Lösung. Anhand dieser Analysen werden im Kapitel 5 diverse Standard- und eigene Algorithmen verbessert oder neu implementiert, um der theoretisch bestmöglichen Lösung möglichst nahezukommen. Im Kapitel 6 werden die Resultate und Erkenntnisse aus den Kapitel 4 und 5 zusammengefasst. Abschliessend werden im Kapitel 7 die Resultate bewertet und weitere Schritte vorgeschlagen.
Im Kapitel 8 findet sich unter anderem ein Glossar, wo die wichtigsten Begrifflichkeiten erklärt werden. Im Kapitel A folgt ein umfassender Anhang.

2. Grundlagen & Hintergrund

Die nachfolgenden Kapitel geben einen groben Überblick über Ensemble Learning, ASR und Text-Alignment. Dabei wird ein Grundwissen über Machine Learning vorausgesetzt.

2.1. Automated Speech Recognition (ASR)

Unter Automated Speech Recognition (ASR) ist das automatische Erkennen und Verarbeiten von gesprochener Sprache zu verstehen. Es gibt viele Einsatzgebiete wie Sprach-Assistenten oder das Erstellen von Meeting-Transkripten aus einer Aufnahme des Meetings. In dieser Arbeit geht es um letzteres. Mit ASR ist im folgenden immer das Transkribieren von gesprochener Sprache gemeint.

2.1.1. Überblick

An ASR-Technologien wird schon lange geforscht. Erste Versuche wurden bereits in den 1950er durchgeführt [4]. Davis et al.[5] beschreiben in 1952 das automatische Erkennen von gesprochenen Zahlen. Ein wichtiger Meilenstein wurde in 1985 mit der Verwendung von Hidden Markov Models (HMMs) für die Repräsentation eines akustischen Signales erreicht[6]. Seither waren bis vor kurzem die meisten ASR-Systeme fast identisch aufgebaut [7]. Ein akustisches Modell bildet aus dem akustischen Signal eine Phonemsequenz. Ein Aussprachemodell modelliert die Sequenz als Graphemsequenz. Ein Sprachmodell wandelt diese Sequenz schlussendlich in einen sinnvollen Satz um. Alle Modelle werden separat trainiert und weisen unterschiedliche Architekturen auf. Dies macht die Systeme sehr komplex. Deswegen hat sich die Forschung immer mehr mit sogenannten End-to-End Modellen auseinandergesetzt. Diese End-to-End Modelle sollen diese Komplexität reduzieren, indem es nur noch ein Modell gibt. Solche Modelle lernen den Output direkt aus den Audiodaten. Inspiriert von grossen Fortschritten in Deep-Learning in anderen Forschungsgebieten, befassen sich End-to-End Modelle hauptsächlich mit Deep-Learning.

2.1.2. Stand der Technik

Publiziert im Jahr 2020, bildet das *wav2vec 2.0* Modell die Grundlage des Standes der Technik [8]. Inspiriert von wav2vec 2.0 und BERT bildet nach aktuellem Kenntnisstand *w2v-BERT*[9] den Stand der Technik. Beide Modelle lernen in einem ersten Schritt aus unannotierten Daten(self-supervised learning), wie sich Sprache darstellen lässt. Dazu werden mehrere Arten von neuronalen Netzwerken kombiniert. Dieses pre-training erlaubt es, die Menge an annotierten Trainingsdaten drastisch zu reduzieren, ohne dabei Performance einbüssen zu müssen. Gerade für selten gesprochene Sprachen ist dies sehr

interessant, da es für solche Sprachen eher wenige annotierte Testdaten gibt. In einem zweiten Schritt wird dann aus annotierten Trainingsdaten gelernt, indem die Netzwerke um weitere Schichten ergänzt werden. Erst in diesen Schichten findet die eigentliche Umwandlung von Audio zu Text statt.
Wav2vec 2.0 und w2v-BERT können für vorgelesene Sprache nahezu perfekte Transkripte erstellen. Auf dem LibriSpeech-Clean Dataset ist deren Fehlerrate (WER) lediglich 1,8 % respektive 1,4 %.

2.1.3. Grenzen von ASR

Wie wav2vec 2.0 und w2v-BERT demonstrieren, funktioniert ASR auf guten und klaren Audioaufnahmen heutzutage ausgezeichnet [10]. Im Gegensatz dazu ist ASR unter suboptimalen Bedingungen nach wie vor schwer. Sogenannte spontane Sprache kann oft nur mit unleserlichen Fehlerraten transkribiert werden. Mit spontaner Sprache ist dabei meist ein spontaner Dialog gemeint. Gerade das Transkribieren von Meetings gilt als schwierig, weil zu einem die Sprache sehr spontan ist und es zum anderen viele Störfaktoren wie Hintergrundgeräusche, schlechte Aufnahmequalität oder Überschneidungen gibt [11].

2.2. Ensemble Learning

Hinter dem Begriff Ensemble-Learning steht die Idee, mehrere Modelle oder Systeme zu trainieren und deren Outputs zu kombinieren, um dadurch eine bessere Performance [1] zu erhalten. In den meisten Fällen ist ein Ensemble von mehreren schwächeren Modellen besser oder zumindest gleich gut, wie das beste Mitglied des Ensembles alleine [12][13]. Es ist allerdings von zentraler Bedeutung, dass sich die einzelnen Modelle ergänzen, sodass das Ensemble gut generalisieren kann [14][15]. Der Grundgedanke hinter dieser Diversität ist es, dass die Fehler eines Modells durch die anderen Modelle kompensiert werden.
Um ein Ensemble zu bilden, gibt es viele unterschiedliche Methoden und Standardalgorithmen. Es gilt dabei zu beachten, dass das auch ein Ensemble dem 'no free lunch' Theorem unterliegt. Ein einfaches Ensemble kann durchaus bessere Resultate als ein komplexes Ensemble liefern. Im Folgenden werden die für diese Arbeit relevanten Methoden kurz beschrieben.

2.2.1. Voting

Voting ist die naheliegendste Ensemble-Methode. Aus einer Menge von komplementären Modellen löst jedes Modell das Problem zunächst eigenständig. Im Anschluss wird durch eine Mehrheitsentscheidung (Majority Vote) das Resultat bestimmt. Für Klassifikationsprobleme wird die Klasse mit den meisten Stimmen gewählt. Für Regressionsprobleme

[1] Unter Performance ist die Genauigkeit einer Vorhersage eines Modells zu verstehen. (unter anderem tieferer mean squared error oder höhere accuracy)

werden die Outputs je nach Aufgabe unterschiedlich kombiniert. Ein naheliegender Ansatz ist es hier, den Durchschnitt aller Outputs zu bilden.
Eine Variation des Votings ist das gewichtete Voting (Weighted Majority Vote) [16]. Es unterscheidet sich, indessen dass nicht jede Stimme gleich gewichtet ist. Bessere Modelle sollen stärker gewichtet werden als schlechte.

2.2.2. Bagging

Bagging, oder Bootstrap Aggregation [17], versucht die Stabilität und Genauigkeit der Vorhersagen zu verbessern, in dem mehrere unabhängige Modelle zu einem Durchschnitt zusammengefasst werden. Dadurch soll die Varianz in der Vorhersage verkleinert werden. Bagging ist eine der Hauptmethoden im Bereich der Ensembles.
Bagging unterscheidet sich von anderen Ensemble-Methoden in der Sicht, dass das Input Dataset in mehrere Teilsets gemischt wird. Jedes Teilset besteht dabei aus $1 - 1/e = 63,2\%$ einzigartigen Entries [18]. Die Teilsets werden danach manipuliert, sodass sie einen Anteil an Duplikaten beinhalten. Dieses Aufteilen und Manipulieren des Datasets heisst Bootstrapping. Auf jedem dieser Testsets wird dann ein Modell trainiert. Der Output dieser Modelle wird danach entweder über Regression (als Durchschnitt), oder über Voting (bei Klassifikation) kombiniert zur finalen Vorhersage. Für diesen Teilschritt des Verfahrens steht Aggregation. Die Abbildung 2 zeigt den Aufbau von Bagging.
Tests mit realen und simulierten Datensätzen unter Verwendung von Klassifizierungs- und Regressionsbäumen, sowie der Auswahl von Teilmengen bei der linearen Regression zeigen, dass Bagging zu erheblichen Genauigkeitssteigerungen führen kann [17].

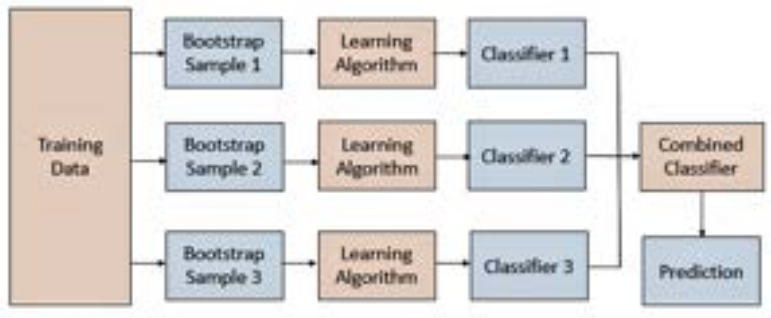


Abbildung 2.: Bagging Aufbau [19]

2.2.3. Random Forest

Bei Random Forests handelt es sich um einen verbreiteten Standard-Algorithmus, welcher oft gut performt [20]. Ein Random Forest besteht aus einer grossen Menge von Decision Trees. Jeder Baum wird dabei unterschiedlich trainiert, indem nur eine zufällige Teilmenge der Features beachtet werden. Gleich wie bei Bagging (Kapitel 2.2.2) werden für das Training aus den Daten Teilmengen gebildet und für jeden Baum unterschiedlich gezogen. Schlussendlich werden alle trainierten Decision Trees mittels Voting

kombiniert. Typische, justierbare Parameter für Random Forests sind die Anzahl Bäume, deren maximale Tiefe sowie die Mindestgrösse des Trainingssamples.

2.2.4. Stacking

Stacking [21] beschreibt eine Gruppe von Ensemble-Methoden, bei welchen mehrere Basismodelle trainiert werden, die dann durch ein Metamodell mit verbesserter Vorhersagekraft zusammengefasst werden. Das Metamodell lernt dabei die Schwächen und Stärken der einzelnen Basismodelle und kombiniert ihre Vorhersagen, um das endgültige Ergebnis zu erreichen. Die Abbildung 3 zeigt einen Blueprint eines Stacking Ensemble. Stacking kann einen positiven Einfluss auf den Generalization error haben, im Vergleich zur Anwendung von nur einem Learning-Algorithmus. Der Generalization error misst die Vorhersagekraft auf noch nicht gesehenen Daten. Stacking hat im Vergleich zu anderen Methoden den Vorteil, dass sowohl Bagging wie auch Boosting Algorithmen als Basismodelle im Ensemble verwendet werden können. Die grösste Herausforderung dabei ist es, die beste Kombination von Modellen und Algorithmen auszuwählen, wenn man das Stacking Ensemble definiert [22]. Wenn es gelingt, gute Algorithmen auszuwählen, können diese ihre gegenseitigen Schwächen ausbügeln und so bessere Vorhersagen erreichen.

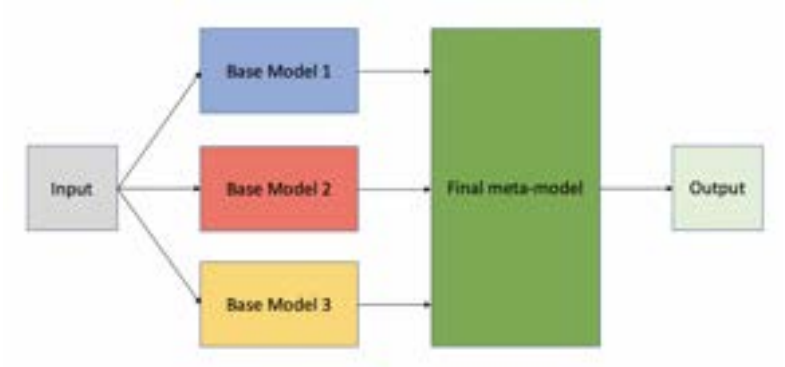


Abbildung 3.: Stacking Blueprint [23]

2.2.5. Boosting

Boosting Algorithmen kombinieren viele Weak Learner iterativ zu einem «Strong-Learner». Mit einem Basis-Algorithmus, etwa einem Decision Tree, wird in jeder Iteration ein Weak Learner trainiert, welcher sich um die Schwächen des Vorgängers kümmert. Zentral ist dabei, dass Boosting Algorithmen auf gewichteten Daten arbeiten. Dabei werden Fehler höher gewichtet, sodass diese in der nächsten Iteration eher korrekt vorhergesagt werden [24].
Es gibt viele Boosting Alorithmen. Unter anderem sind AdaBoost [24][25] und Gradientboost [26] bekannte Algorithmen.

Gradient Boosting

In dieser Arbeit wird von den Standardalgorithmen hauptsächlich Gradient Boosting verwendet. Gradient Boosting verwendet Gradient Descent, um eine beliebige, differenzierbare Loss-Funktion zu minimieren. Natekin et al.[27] erklären den Algorithmus von Friedman[26] wie folgt.

Algorithm 1 Friedman's Gradient Boost algorithm, übernommen aus Natekin et al.[27]

inputs:

- input data $(x, y)_{i=1}^{N}$
- number of iterations M
- choice of the loss-function $\Psi(y, f)$
- choice of the base-learner model $h(x, \theta)$

Algorithm:

1: initialize $\widehat{f_0}$ with a constant
2: **for** t = 1 to M **do**
3: compute the negative gradient -$g_t(x)$
4: fit a new base learner function $h(x, \theta_t)$
5: find the best gradient descent step-size ρ_t :
$\rho_t = argmin_\rho \sum_{i=1}^{N} \Psi \left[y_i, \widehat{f}_{t-1}(x_i) + \rho h(x_i, \theta_t) \right]$
6: upate the function estimate:
$\widehat{f}_t \leftarrow \widehat{f}_{t-1} + \rho_t h(x, \theta_t)$
7: **end for**

Der Algorithmus geht von einer Basisfunktion $\widehat{f_0}$ aus und wird für M Iterationen ausgeführt. Im Schritt 3 wird der negative Gradient $g_t(x)$ (Residuen) der Loss-Funktion aus den Testdaten und dem existierenden Ensemble aus der vorherigen Iteration berechnet. Der negative Gradient entspricht dabei intuitiv dem Fehler des Ensembles aus der vorherigen Iteration.

$$g_t(x) = -\frac{\partial \Psi(y, f_{t-1}(x))}{\partial f(x)} \tag{2.1}$$

Im Schritt 4 wird nun ein neues Modell trainiert, welches diese Fehler approximiert. Anschliessend wird im Schritt 5 ermittelt, wie stark das neue Modell ins Gewicht fallen soll. Oftmals wird dafür auch einfach eine Konstante a zwischen 0 und 1, die Learning-Rate, verwendet. Abschliessend wird das soeben neu gelernte Modell dem Ensemble hinzugefügt. Der beschriebene Algorithmus lässt sich von Regressionsproblemen auf Klassifikationsprobleme übertragen, indem Klassenzugehörigkeiten als Wahrscheinlichkeiten aufgefasst werden.

Gradient Boosting : Parameter

Gradientboosting hat einige Hyperparameter, welche auf ein spezifisches Problem angepasst werden können.

- Das Basis-Modell $h(x, \theta)$, welches in jeder Iteration den Fehler approximiert. Oftmals ist h ein Decision Tree.
- Die Loss-Funktion $\Psi(y, f)$ wird nach Machine Learning Task bestimmt. Für Regression könnte etwa der squared error loss verwendet werden.
- Die Anzahl Iterationen M entspricht der Anzahl der Modelle, welche trainiert werden. M wird typischerweise so gesetzt, dass der Algorithmus bei Konvergenz abbricht.
- Die Learning-Rate a beschreibt, wie sehr ein neues Modell ins Gewicht fällt. Bei einer Learning-Rate von 1 konvergiert der Algorithmus schneller ist dafür aber anfälliger für Overfitting. Bei einer tiefen Learning-Rate sind zwar mehr Iterationen nötig, dafür kann das Ensemble tendenziell besser generalisieren.

2.3. Word-/Sentence-Embeddings

Word- oder Sentence-Embeddings sind Vektoren, welche die Bedeutung eines Wortes beziehungsweise eines Satzes darstellen. Dabei sollen Wörter oder Sätze mit ähnlicher Bedeutung auch ähnliche Vektoren haben. Embeddings machen natürliche Sprache also in gewisser Weise mathematisch vergleichbar.
Embeddings werden typischerweise mit neuronalen Netzwerken gelernt. Es gibt viele vortrainierte Modelle, welche Word- oder Sentence-Embeddings berechnen können. Für Word-Embeddings sind *GloVe* [28] oder *fasttext* [29] bekannte, vortrainierte Modelle. Sentence-Embeddings werden heutzutage meist mit transformerbasierten[1] Modelle wie *roBERTa* erstellt [30].
Embeddingvektoren haben typischerweise zwischen 300 und 1000 Komponenten. Ein Beispiel mit fünf Komponenten könnte wie folgt aussehen (Word-Embedding):

$$\begin{bmatrix} w & K1 & K2 & K3 & K4 & K5 \\ cat & 0 & 0.8 & 0.8 & 1 & 1 \\ dog & 0.4 & 0.55 & 0.8 & 1 & 0.99 \\ teacher & 1 & 0.0 & 0.7 & 0.43 & 0.24 \end{bmatrix}$$

$$cos\_sim(w_{cat}, w_{dog}) = 0.9734$$
$$cos\_sim(w_{cat}, w_{teacher}) = 0.5395$$

Wie im Beispiel zu sehen ist, sind die Word-Embeddings für *cat* und *dog* sehr ähnlich, da beide Worte beliebte Haustiere sind. [2] Vergleicht man jedoch *cat* oder *dog* mit *teacher*, so ist der Unterschied grösser.

[1] Die Transformer-Architektur beschreibt eine neuartige Architektur von neuronalen Netzen. Die Architektur wird in Kapitel 5.4.1 noch weiter erklärt

[2] Die hohe *cos_sim* von 0.9734 dient zur Veranschaulichung. Real wäre cat and dog weiter auseinander

2.4. Substitution, Insertion, Deletion

In vielen Bereichen dieser Arbeit, wie zum Beispiel Metriken oder Alignment, ist von Substitutions, Insertions und Deletions die Rede. Damit sind Transformationen gemeint, um einen Satz s_1 in einen anderen Satz s_2 zu transformieren. Beispiel:

$$s_1 = \text{Hello World}$$

$$\downarrow$$

$$\text{substitute(Hello, Bye) \& insert(see you tomorrow)}$$

$$\downarrow$$

$$s_2 = \text{Bye World, see you tomorrow}$$

2.5. ASR-Metriken

Für die Evaluation von ASR-Systemen gibt es verschiedene Metriken. Diese liefern einen numerischen Vergleich zwischen der bekannten, menschlich annotierten Referenz einer Audioaufnahme und der Hypothese eines ASR-System.

2.5.1. Word Error Rate (WER)

Die Word Error Rate [31], kurz WER, hat sich als gängigste Metrik zur Messung von ASR-Systemen etabliert. Dabei wird ein Referenztext mit einer Hypothese verglichen. Die Summe der Abweichungen ergibt einen prozentualen Score. Die Word Error Rate ist definiert durch die Formel

$$\text{WER} = \frac{\text{S} + \text{D} + \text{I}}{\text{H} + \text{S} + \text{D}} \tag{2.2}$$

wobei S = Anzahl von ausgewechselten, D = Anzahl von gelöschten, I = Anzahl der eingefügten und H = Anzahl von übereinstimmenden Wörtern ist.
Die WER ist allerdings keine perfekte Evaluationsmetrik. Die Kritik geht in mehrere Richtungen. Zum einen sind in der WER alle Fehler gleich stark gewichtet. Es haben aber nicht alle Fehler denselben Einfluss auf die Lesbarkeit. Zum anderen ist die WER anfällig auf unterschiedliche Hypothesenlängen und nicht immer repräsentativ. Auf kürzeren Hypothesen fallen Fehler mehr ins Gewicht. Andersherum braucht es bei einer langen Hypothese viele Fehler, bis die WER signifikant sinkt. Trotz dieser Kritik ist die WER auch in dieser Arbeit eine der Hauptmetriken.

2.5.2. SemDist

SemDist beschreibt die semantische Distanz zwischen zwei Texten [32]. Dabei bedeutet eine SemDist von 0, dass die Texte exakt dasselbe bedeuten und von 1, dass die Texte eine komplett unterschiedliche Bedeutung haben. Die Berechnung von SemDist ist aufgeteilt

in zwei Teile. Mit trainierten Sprachmodellen, wie roBERTa, wird eine Utterance und deren Referenz in ein Sentence-Embedding umgewandelt. Aus diesen Sentence-Embeddings ist SemDist wie folgt definiert:

$$\text{SemDist}(e_{ref}, e_{hyp}) = 1 - \frac{(e_{ref})^T * e_{hyp}}{\|e_{ref}\| * \|e_{hyp}\|} \tag{2.3}$$

Anders ausgedrückt kann SemDist also als Cosinus-Distanz angesehen werden:

$$\text{SemDist}(e_{ref}, e_{hyp}) = 1 - cos\_sim(e_{ref}, e_{hyp}) \tag{2.4}$$

SemDist korreliert mit der WER. Die SemDist wird automatisch besser, wenn die WER verbessert wird. Allerdings kann SemDist in Vergleichen von zwei Systemen mit gleicher WER zeigen, welches System besser ist. Dazu ein Beispiel aus dem SemDist-Paper [32]:

ref:	*This*	*is*	*a*	*cat*	WER	SemDist
h1:	*This*	*is*	*the*	*cat*	25 %	0.0077
h2:	*This*	*is*	*a*	*cap*	25 %	0.0157

Im Beispiel ist für beide Hypothesen die WER identisch. Die SemDist zeigt jedoch einen wichtigen Unterschied und es kann gesagt werden, dass das erste Transkript besser ist als das zweite.

2.5.3. Match Error Rate

Die Match Error Rate [33], kurz MER, gibt annähernd die Wahrscheinlichkeit an, ob ein Wort richtig transkribiert wird. Die Match Error Rate ist definiert durch die Formel

$$\text{MER} = \frac{\text{S} + \text{D} + \text{I}}{\text{H} + \text{S} + \text{D} + \text{I}} \tag{2.5}$$

wobei S = Anzahl von ausgewechselten, D = Anzahl von gelöschten, I = Anzahl der eingefügten und H = Anzahl von übereinstimmenden Wörtern ist.

2.5.4. Word Information Lost

Die Word Information Lost [33], kurz WIL, ist eine Annäherung an die relative Information, welche durch das Transkribieren verloren geht. Die Word Information Lost ist definiert durch die Formel

$$\text{WIL} = 1 - \frac{\text{H}^2}{(\text{H} + \text{S} + \text{D})(\text{H} + \text{S} + \text{I})} \tag{2.6}$$

2.5.5. Word Information Preserved

Die Word Information Preserved [33], kurz WIP, ist das Gegenstück von WIL.

$$\text{WIP} = 1 - WIL \tag{2.7}$$

2.6. CEASR Korpus

Das CEASR Korpus [34] ist eine Sammlung von mehreren öffentlichen Sprachkorporas. Für jeden Korpus sind Metadaten sowie die Transkripte von 14 ASR Systemen enthalten. Der Output der ASR Systeme basiert in dieser Version auf dem Stand der Technik von 2019. Für diese Arbeit werden die sechs englischen Sprackkorpora AMI, TedLium, ST, LibriSpeech, VoxForge und CommonVoice berücksichtigt. Die Korpora lassen sich in spontane Sprache (AMI), halb-spontaner Monolog (TedLium) und vorgelesener Sprache (ST, LibriSpeech, VoxForge und CommonVoice) aufteilen. Das CEASR Korpus enthält 25'094 Äusserungen mit Metadaten zum Sprecher und der Aufnahme. Die Äusserungen basieren auf über 45 Stunden Audioaufnahmen. Die Daten sind in einer einheitlichen Form aufgearbeitet und strukturiert als JSON abgelegt. Die ASR Systeme, die zum Transkribieren verwendet wurden, sind sowohl von kommerziellen Anbietern, als auch Open Source Lösungen. Die kommerziellen Lösungen sind als Systeme B, C und D anonymisiert worden. Unterschiedliche Konfigurationen [1] der Systeme werden mit Zahlencodes anonymisiert. Folglich ist zum Beispiel das Korpus «B1» das System 1 von Hersteller B. Das Korpus liegt zum Start der Arbeit in der Version 1,0 vor.

2.7. Alignment

Das Alignment im Kontext dieser Arbeit beschreibt das Problem, mehrere ähnliche Hypothesen vergleichbar zu machen. Die Hypothesen verschiedener Systeme sind grundsätzlich ähnlich, jedoch nicht identisch. Gewisse Wörter könnten in der einen Hypothesen fehlen oder anders transkribiert worden sein. Das Ziel vom Alignment ist es, dass verschiedene Hypothesen auf Wortbasis miteinander verglichen werden können. Beispiel:

I	*love*	*cats*	*and*	*dogs*	*the*	*same*
I	*like*	*cats*	*people*	*and*	*dogs*	*same*
love	*cats*	*and*	*ducks*	*the*	*same*	

↓ Alignment ↓

I	*love*	*cats*		*and*	*dogs*	*the*	*same*
I	*like*	*cats*	*people*	*and*	*dogs*		*same*
	love	*cats*		*and*	*ducks*	*the*	*same*

2.7.1. ROVER

Fiscus[35] beschreibt mit Recognizer Output Voting Error Reduction (ROVER) unter anderem ein Alignment von beliebig vielen Hypothesen an eine Primärhypothese. Die vorgeschlagene Lösung basiert auf sogenannten Word Transition Networks (WTN).

[1] Viele Systeme können je nach Anwendungsfall konfiguriert werden, z. B. für Videoaufnahmen, Dialoge, Interviews etc.

Word Transition Network

Ein Word Transition Network kann man als endlichen Automat (FSM) auffassen. Diese FSM stellt die Wörter und deren Übergänge in einem Text dar. Ohne Alignment ist ein WTN einfach eine sequentielle Auflistung. Die Übergänge in Abbildung 4 stellen Wörter dar.

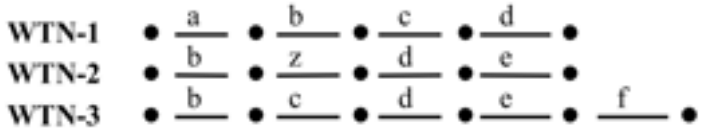


Abbildung 4.: Darstellung von drei WTNs, übernommen aus [35]

Alignment von zwei WTN

Mittels Dynamic Programming (DP) werden rekursiv die Zustände und deren Übergänge kombiniert. Dabei wird die Anzahl Insertions, Deletions und Substitions minimiert [35][36]. Dies entspricht der Levensthein-Distanz auf Wort- statt Buchstabenbasis.

Abbildung 5.: Alignment von WTN-1 und WTN-2 (siehe Fig. 4), übernommen aus [35]

Alignment von mehreren WTN

Die Alignments von mehreren Hypothesen an die Primärhypothese wird heuristisch-iterativ durchgeführt. Schritt für Schritt wird jede Hypothese in das resultierende WTN anhand einiger Regeln übernommen. Zusammengefasst lauten diese Regeln:

1. Substitutionen und korrekte Wörter werden als Zustandsübergang übernommen
2. Deletions werden als leerer Zustandsübergang eingetragen
3. Für Insertions wird ein Sub-WTN erstellt und eingefügt. Das Sub-WTN hat einen neuen Zustand, welcher die Insertion als Übergang zum nächsten Wort hat. Ausserdem wird für jedes bereits übernommene Alignment ein leerer Zustandsübergang hinzugefügt

Verwendet man die obigen Regeln, um die Alignments WTN1 - WTN2 und WTN1 - WTN3 (siehe Abbildung 4) zu kombinieren, so erhält man die resultierende WTN-Base" (siehe Darstellung 6). Substituiert man nun die Zustandsübergänge mit den korrespondierenden Wörtern und @ mit Leerzeichen, so erhält man ein Alignment der drei Hypothesen.

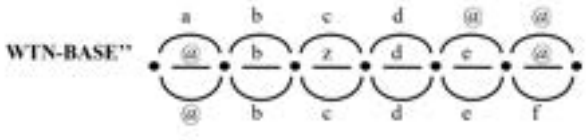


Abbildung 6.: Alignment von WTN-1, WTN-2, WTN-3 (siehe Fig. 4), übernommen aus [35]

2.7.2. Existierender Alignment Algorithmus - Triple Alignment

Ein Alignment-Algorithmus der ZHAW liegt bereits vor [2]. Dieser Alignment-Algorithmus unterscheidet sich allerdings von ROVER. Zum einen werden keine Word Transition Networks verwendet. Zum anderen können nur maximal drei Hypothesen angeglichen werden. Ansonsten basiert die Implementation auf der Kombination von phonetischen Codes und der Levensthein-Distanz. Ähnlich wie bei ROVER wird auch hier eine Primärhypothese festgelegt, an welche jeweils eine andere Hypothese angeglichen wird. Im Folgenden wird der vorhandene Algorithmus im Detail erklärt. Zuerst wird das Alignment zwischen zwei Hypothesen betrachtet. Im Anschluss wird auf das Vereinen von zwei Alignments zum «Triple-Alignment» eingegangen.

Schritt I: Preprocessing

Als erster Schritt wird auf beiden Hypothesen ein Preprocessing durchgeführt. Es werden Satz- und Sonderzeichen aus den Hypothesen entfernt. Ausserdem werden bekannte Eigenheiten von CEASR bereinigt. Manche Systeme schreiben zum Beispiel [*noise*] oder *[???]*, falls etwas nicht als Wort transkribiert werden konnte. Ausserdem werden "ß" zu "ss" gemacht.

Schritt II: Delete Correct

Als Nächstes werden als Vereinfachung mit einer Heuristik alle grossen Übereinstimmungen der beiden Transkripte "gelöscht" beziehungsweise vorgemerkt. Ein rekursiver Algorithmus bildet n-Gramme für die zwei Texte. Dabei ist anfangs n gross (bis zu 50 Worte) und wird in jedem Schritt kleiner, bis $n = 3$ ist. Falls ein n-gram in beiden Texten gleich oft vorkommt und die Texte gleich lang sind, wird das n-gram mit dem Platzhalter $[n]$ substituiert. In einer weiteren Heuristik werden kleinere Übereinstimmungen mit *gestalt pattern matching*[37][38] gesucht. Zum Schluss werden im resultierenden Alignment für alle Substitutionen $[n]$ jeweils n Wörter als korrekt gekennzeichnet.
Beispiel:

I	*love*	*cats*	*and*	*dogs*	*the*	*same*
I	*like*	*cats*	*people*	*and*	*dogs*	*same*

wird zu

$$\begin{array}{cccccc} I & love & cats & [2] & the & same \\ I & like & cats & people & [2] & same \end{array}$$

und schlussendlich zu

$$\begin{array}{lccccccc} hyp1: & I & love & cats & and & dogs & the & same \\ & ? & ? & ? & correct & correct & ? & ? \end{array}$$

$$\begin{array}{lccccccc} hyp2: & I & like & cats & people & and & dogs & same \\ & ? & ? & ? & ? & correct & correct & ? \end{array}$$

Schritt III: Alignment

Als Letztes werden für alle ?-Einträge der Mutationstyp (Substitution, Insertion, Deletion) bestimmt und miteinander vereinigt. Dazu werden zuerst alle Einträge zwischen korrekten Wörtern aggregiert.

$$hyp1: [\text{I love cats, the same}]$$

$$hyp2: [\text{I like cats people, same}]$$

Anschliessend wird über jeden Eintrag in diesen Listen iteriert. Falls beide Einträge genau ein Wort beinhalten, handelt es sich entweder um eine Substitution oder um ein korrektes Wort und es kann direkt verglichen werden. Ansonsten werden bi- oder trigramme gebildet. Falls der Eintrag eine Zahl beinhaltet, werden 5-Gramme gebildet und Zahlen als Worte ausgeschrieben. N-Gramme helfen hier, um ähnliche Teilsätze besser miteinander zu matchen. Aus den gebildeten n-Grammen wird nun jeweils ein phonetischer Code gebildet. Für alle Paare von phonetischen Codes wird mittels Levensthein-Distanz das Paar mit der höchsten Ähnlichkeit (tiefste Levensthein Distanz) als Match ausgewählt und aus der Liste entfernt. Durch das Verwenden von phonetischen Codes und n-Grammen können auch Wortfolgen wie «before» und «be for» erkannt werden. Falls es keinen Match gibt, handelt es sich um eine Deletion oder Insertion, je nachdem, ob die Primärhypothese oder die Sekundärhypothese länger ist.
Beispiel für den ersten Eintrag aus der vorherigen Auflistung:

$$3-grams1: [\text{'i', 'love', 'cats', 'i love', 'love cats', 'i love cats'}]$$

$3-grams2:$ ['i', 'like', 'cats', 'people', 'i like', 'like cats', 'cats people', 'i like cats', ...]

In diesem Fall ist das erste Paar mit der höchsten Übereinstimmung 'i' (100 %). Im Resultat wird nun ein also an der Position von 'i' ein correct eingefügt und 'i' aus der abzuarbeitenden Liste entfernt. In der nächsten Iteration verbleibt

$$3-grams1: [\text{'love', 'cats', 'love cats'}]$$

$$3-ngrams2: [\text{'like', 'cats', 'people', 'like cats', 'cats people', 'like cats people'}]$$

In diesem Fall ist 'cats' eine 100% Übereinstimmung, also wird an der Position von 'cats' in beiden Hypothesen ein correct hinzugefügt. In der folgenden Iteration würde like und love gematcht, weil diese am ähnlichsten sind. In der letzten Iteration verbleibt nur noch 'people', was dann zu einer Insertion führt. Anschliessend wird dasselbe für den zweiten Eintrag mit 'the same' und 'same' durchgeführt.
Nachdem alle Kombinationen abgearbeitet wurden, sind alle ?-Einträge ersetzt. Es verbleiben zwei Listen mit den Mutationstypen, welche nun vereint werden müssen.

$hyp1:$	*I*	*love*	*cats*	*and*	*dogs*	*the*	*same*
	correct	*substitution*	*correct*	*correct*	*correct*	*deletion*	*correct*

$hyp2:$	*I*	*like*	*cats*	*people*	*and*	*dogs*	*same*
	correct	*substitution*	*correct*	*insertion*	*correct*	*correct*	*correct*

Dazu wird über die erste Liste iteriert. Einträge am selben Index werden übernommen, wenn sie vom Typ Substitution oder Correct sind. Eine Deletion in der Primärhypothese wird in das Resultat übernommen. Eine Insertion in der Sekundärhypothese wird ebenfalls übernommen. Schlussendlich resultiert also das Alignment:

I	*love*	*cats*		*and*	*dogs*	*the*	*same*
I	*like*	*cats*	*people*	*and*	*dogs*		*same*
correct	*substitution*	*correct*	*insertion*	*correct*	*correct*	*deletion*	*correct*

Triple Alignment

Es soll nun eine dritte Hypothese angeglichen werden:

I	*love*	*cats*	*and*	*dogs*	*the*	*same*
I	*like*	*cats*	*people*	*and*	*dogs*	*same*
love	*cats*	*and*	*ducks*	*the*	*same*	

Als Erstes wird gemäss Schritt I-III ein Alignment zwischen der Ersten und der Dritten Hypothese erstellt:

I	*love*	*cats*	*and*	*dogs*	*the*	*same*
	love	*cats*	*and*	*ducks*	*the*	*same*
deletion	*correct*	*correct*	*correct*	*substitution*	*correct*	*correct*

Nun werden die zwei Alignment-Paare ähnlich wie bei ROVER anhand einiger Regeln kombiniert: Es wird über die beiden Listen (Alignment-Paare) iteriert und sichergestellt, dass immer dieselben Elemente der Primärhypothese verglichen werden. Deletions, Substitutions und Correct werden übernommen. Für Insertions werden neue Einträge erstellt. Das finale Alignment lautet:

I	*love*	*cats*		*and*	*dogs*	*the*	*same*
I	*like*	*cats*	*people*	*and*	*dogs*		*same*
	love	*cats*		*and*	*ducks*	*the*	*same*

2.8. Literaturübersicht - Ensemble im Bereich ASR

Dass über mehrere ASR Systeme ein Ensemble gebaut wird, ist in der wissenschaftlichen Literatur eher selten erwähnt. Die meisten Referenzen sind schon einige Jahre alt [35][39][40][41]. Ein Grund für dieses Desinteresse in den letzten Jahren könnten neue Modelle und Architekturen wie wav2vec 2.0 sein, welche den Stand der Technik so weit vorangetrieben haben, dass sich Ensemble-Learning nicht mehr lohnt.
Die Idee, mehrere Outputs zu einem besseren Output zu kombinieren, ist naheliegend. Bereits im Jahr 2000 wurde mit ROVER[35] ein Mechanismus beschrieben, um die Outputs von verschiedenen Systemen aneinander auszurichten (= Alignment, siehe Kapitel 2.7) und mittels Majority Voting eine bessere Hypothese zu erreichen.
Siohan et al. [40] versuchten im Jahr 2005 durch einen weiteren Voting-Ansatz, die WER zu verbessern. Mit Decision Trees wurden mehrere komplementäre ASR-Systeme aufgebaut. Der Versuch nutzte Datensets des MALACH Projekts sowie vom DARPA EARS Projekt. Die Hypothesen der Systeme wurden ebenfalls mit ROVER kombiniert und ausgewertet. Die erreichte Verbesserung lag bei 1,1 %, wenn drei Systeme kombiniert wurden.
Dimitrakakis et al.[39] befassen sich mit Boosting und Bagging von Hidden Markov Models (HMMs) auf der Stufe von Phonemen und gesamten Utterances. Mittels Boosting konnte die WER zwar nicht signifikant verbessert werden, aber mittels Bagging wurde die WER auf dem OGI Numbers 95 Dataset von 7,52 auf 5,97 gesenkt.
Etwas neuer ist der Ansatz von Deng et al. [41]. Es handelt sich um eine eher klassische Umsetzung von Ensemble-Learning. Mittels Stacking werden verschiedene neuronale Netzwerke (DNN, CNN, RNN) trainiert und kombiniert. Auf dem TIMIT-Datenset konnte so die Fehlerquote der Phonemerkennung des akustischen Modells von 22,1 % (DNN) bzw. 19,3 % (CNN und RNN) auf 18,3 % reduziert werden. Eine Aussage über die Auswirkungen auf WER wurde jedoch nicht getroffen.
Für ASR-Architekturen auf dem Stand der Technik, wie wav2vec 2.0 [8] oder w2v-BERT [9] (siehe auch Kapitel 2.1), sind die Ansätze von Deng et al.[41] oder Dimitrakakis et al.[39] nicht mehr direkt anwendbar, weil sich die Architektur von ASR-Systemen stark verändert hat. Wie in Kapitel 2.1 schon erwähnt wurde, bringen diese Architekturen aber von sich aus bereits eine bessere Leistung als die erwähnten Ensemble-Methoden.

3. Vorgehen und Methoden

3.1. Zielsetzung

Diese Arbeit verfolgt mehrere Ziele. Als Erstes soll ein Alignment-Algorithmus implementiert werden, welcher eine beliebige Anzahl von Transkripten verarbeiten kann. Generierte Alignments sollen ausserdem automatisch bewertet werden können, damit die Qualität eines Alignments beurteilt werden kann. Als Zweites sollen Ensemble-Methoden gefunden werden, welche eine möglichst grosse Verbesserung der WER und der SemDist erzielen. Als Drittes sollen die Grundlagen aufgearbeitet beziehungsweise geschaffen werden, sodass nachfolgende Arbeiten darauf aufbauen können.

3.2. Einschränkungen der Arbeit

Diese Arbeit ist bewusst eingeschränkt auf das Kombinieren von Outputs (Transkripten) von mehreren ASR-Systemen. Ein anderer, klassischer Ansatz wäre etwa das Training und Erstellen eines modernen ASR-Systems mit Ensemble-Methoden. Von einem solchen Ansatz wurde bewusst abgesehen, da die Lösung schlussendlich für Interscriber gedacht ist und Interscriber kommerzielle Systeme verwendet, welche nicht mit Ensemble-Methoden beeinflusst werden können. Deshalb werden ASR-Systeme in dieser Arbeit auch weitgehend als Blackbox angeschaut.
Ferner gilt ebenfalls eine Einschränkung auf drei Systeme. Dies hat vor allem Kostengründe. Falls die Lösung in Interscriber zum Einsatz kommen würde, würde jedes verwendete System Kosten verursachen, was irgendwann nicht mehr wirtschaftlich ist. Deswegen wird die Option, mehr als drei Systeme zu verwenden, nur kurz erforscht, danach aber weitgehend weggelassen. Dennoch werden alle Algorithmen für eine beliebige Anzahl von Systemen entworfen.
Als letztes beschränkt sich die Arbeit auf die englische Version von CEASR. Die Algorithmen sollten jedoch ohne grosse Modifikationen mit der deutschen Version von CEASR zurechtkommen.

3.3. Auswahl des Korpus

Diese Arbeit verwendet CEASR in der Version 1.0, was dem Stand der Technik von 2019 entspricht. Es ist daher anzunehmen, dass möglicherweise durch Ensemble-Learning auf den Daten von 2019 der Stand der Technik von 2022 gar nicht geschlagen werden kann. Eine spätere Analyse (siehe Kapitel 4.5) bestätigt diesen Verdacht. Trotzdem wird mit

CEASR fortgefahren, mit der Erwartung, dass die Erkenntnisse und Resultate allgemeingültig sind. CEASR wurde aufgrund dessen Einfachheit gewählt. Alle Daten sind in einem einheitlichen Format. Ausserdem basiert die existierende Pipeline aus der STT-Ensemble-Learning - Analyse ebenfalls auf den Erkenntnissen aus CEASR. Diese Arbeit verwendet die öffentliche Version von CEASR. Die private, ZHAW-interne Version, welche für die bestehenden Resultate in Abbildung 8 verwendet wurde, beinhaltet mit RT, Switchboard und TIMIT noch weitere Korpora, welche in der öffentlichen Version nicht vorhanden sind. Ferner ist das Korpus *TEDLIUM-unsegmented* in der öffentlichen Version vorhanden, wird jedoch in Abbildung 8 nicht erwähnt. Aus diesen Gründen verwendet diese Arbeit schlussendlich die folgenden 8 Korpora aus CEASR: st, librispeech(clean & other), AMI (Headset und Mix), commonvoice, TEDLIUM-segmented und voxforge.

3.4. Ausgangslage und Ensemble-Prozess

Grundlage dieser Arbeit ist die Ensemble-Pipeline aus der STT-Ensemble-Learning - Analyse [2]. Die Pipeline aus dem Jahr 2020 bietet auf der Projektseite eine kurze Dokumentation. Darin wird die grobe Funktionsweise der Pipeline erläutert. Anfangs bot die Pipeline die Möglichkeit, Ensemble-Methoden für drei Systeme zu evaluieren. Dazu werden die Daten von den drei besten Systemen aus dem CEASR-Korpus verwendet. Für jede Utterance werden die Hypothesen der drei Systeme benutzt, um eine neue Hypothese zu erzeugen. Danach werden die Hypothesen mit der WER bewertet. Die Pipeline kann in drei Komponenten aufgeteilt werden: Alignment, Ensemble-Methoden und Evaluation.
In dieser Arbeit wurde diese Evaluationspipeline übernommen und erweitert. Abbildung 7 zeigt die Pipeline, so wie sie am Ende der Arbeit vorliegt und veranschaulicht die Abhängigkeiten im Ensemble Prozess. Insbesondere sei an dieser Stelle nochmals die Rolle des Alignments erwähnt. Das Alignment bildet die absolute Grundlage für die Ensemble-Methoden in dieser Arbeit. Jede Ensemble-Methode operiert auf einem Alignment.

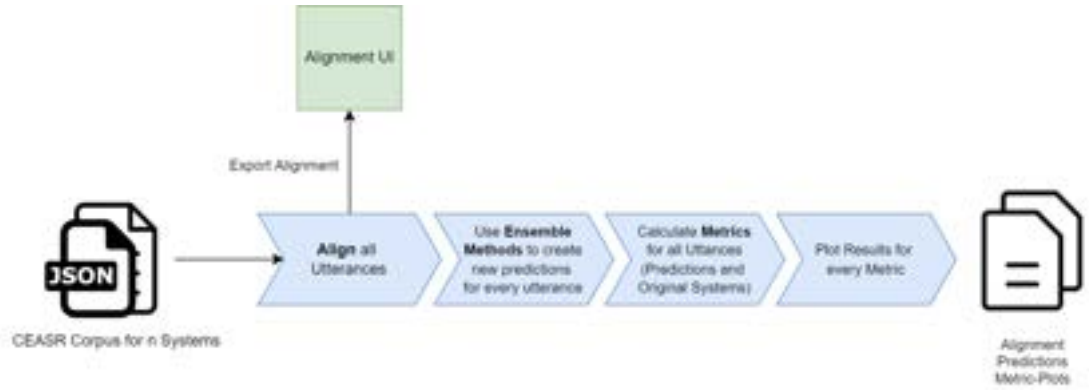


Abbildung 7.: Komponenten der Evaluationspipeline – Der Ensemble Prozess

Wie im Kapitel 2.7 detailliert beschrieben wurde, kann die bestehende Lösung ein Ali-

gnment für genau drei Hypothesen erzeugen. Dazu wird eine Primärhypothese ausgewählt. Die anderen zwei Hypothesen werden dann einzeln an die Primärhypothese angeglichen. Erst am Schluss werden die Alignments kombiniert.

Die existierende Pipeline implementiert folgende Ensemble-Methoden:

- Voting – Kapitel 5.2
- WordSimilarity Voting – Kapitel 5.3
- Perplexities (GPT) – Kapitel 5.4
- Hybrid (Voting + Perplexities) – Kapitel 5.5

Ausserdem wurde ein anderer Best-Possible (siehe Kapitel 4.5) Ansatz implementiert, welcher jedoch nicht mehr lauffähig war. Das Scoring der Pipeline funktioniert über eine eigene Implementierung der WER. Andere Metriken wurden nicht implementiert. Abbildung 8 zeigt die erreichten WER Verbesserungen. Unter anderem ist ersichtlich, dass die erzielten Verbesserungen im Rahmen von 1 bis 2 % WER-Reduktion liegen.
Vergleiche zu Lösungen in dieser Arbeit sind mit Vorsicht zu ziehen. Als Erstes wird im Kapitel 4.1 beschrieben, dass die WER - Implementation in dieser Arbeit unterschiedlich ist und es Messunterschiede gibt. Als Zweites wird in Abbildung 8 nur eine einzelne Originalhypothese, Google, betrachtet. Es ist unklar, welches der Systeme im öffentlichen Korpus Google ist. Als Drittes werden die Resultate in dieser Arbeit stets mit allen Originalhypothesen verglichen und es zeigt sich des Öfteren, dass nicht immer dasselbe System das Beste ist. Weil Abbildung 8 nur mit einer Originalhypothese vergleicht, ist unklar, ob diese wirklich immer die beste ist. Als Letztes ist unklar, ob die STT-Ensemble-Learning - Analyse wirklich denselben Korpus wie diese Arbeit verwendet. So beinhaltet Abbildung 8 Aussagen über die Korpora rt, switchboard und timit, welche in der öffentlichen CEASR Version nicht vorhanden sind.

	The very best possible. Word wise	Best possible	Voting + Best possible Semi-artificial	Google w/o Ensembling	Voting	Perplexities GPT*	Voting + GPT*	Embeddings Similarity Voting
ami_segmented_h	0.2792	0.3147	0.323	0.403	0.3891	0.4262	0.3916	**0.3832**
ami_segmented_mix	0.5822	0.7093	0.7227	**0.7788**	0.7912	0.8094	0.8038	0.8251
commonvoice	0.0354	0.0435	0.0559	0.099	0.0705	0.0795	**0.066**	0.0692
librispeech_clean	0.0201	0.0251	0.0429	0.0749	0.053	0.0582	**0.0497**	0.0515
librispeech_other	0.0547	0.068	0.0925	0.1549	0.1193	0.1309	**0.1133**	0.1185
rt_segmented_h	0.2639	0.2899	0.2984	0.3618	0.3497	0.3823	**0.3482**	0.3505
rt_segmented_hsum	0.5605	0.6542	0.6678	**0.7324**	0.7332	0.7543	0.7406	0.7642
st	0.0246	0.0268	0.0357	0.0443	0.0407	0.0479	0.0408	**0.0405**
switchboard_segmented	0.1778	0.2079	0.2192	0.3503	0.343	0.352	**0.3405**	0.3466
tedlium_segmented	0.0291	0.0371	0.046	0.0787	0.0597	0.0871	**0.0579**	0.0583
timit	0.02	0.0259	0.0412	0.0628	0.0511	0.0629	**0.0482**	0.0509
voxforge	0.0352	0.0422	0.0551	0.0868	0.0651	0.0777	**0.0633**	0.0651
AVERAGE	0.1736	0.2036	0.2167	0.2690	0.2554	0.2724	**0.2553**	0.2603

* ≤ 3 context words before and after the excerpt of interest taken into account
# Spaces not counted as match

Abbildung 8.: Resultate der originalen Ensembling Implementation. Übernommen aus [2]. Das beste Model pro Korpus ist jeweils markiert.

3.5. Evaluationen mit WER und SemDist

Wie im Kapitel 2.5 aufgezeigt wurde, gibt es viele Metriken, welche zur Bewertung eines ASR Systems verwendet werden können. Diese Arbeit beschränkt sich allerdings auf die WER und SemDist. Die WER wurde gewählt, weil es die gängigste Metrik im ASR Umfeld ist und die originale Pipeline ebenfalls mit der WER evaluierte. So können Vergleiche gemacht werden. SemDist wurde aufgrund der semantischen Komponente gewählt. Ausserdem ist SemDist für Situationen gedacht, in denen ein WER Vergleich keinen Unterschied zeigt. Wie in Kapitel 2.5.2 erklärt, kann so auch bei identischer WER eine Aussage über das bessere System getroffen werden. Die Evaluationen werden in dieser Arbeit mit der Evaluationspipeline (Anhang A.5) durchgeführt.

4. Datenanalyse

In den folgenden Kapiteln werden Analysen über diverse Themen durchgeführt, um das Umfeld für die zu implementierenden Ensemble-Methoden festzulegen. Es muss festgestellt werden, unter welchen Umständen die Ensemble-Methoden agieren.

4.1. Implementation der Evaluationsmetriken

Um die Effektivität des Ensembles zu evaluieren, wurden die in Kapitel 2.5 erwähnten Evaluationsmethoden in der existierenden Pipeline implementiert. Diese Metriken bilden die Grundlage für spätere Auswertungen und erlauben den Vergleich zwischen verschiedenen Ideen und Ansätzen.

4.1.1. WER, MER, WIL, WIP

Die Berechnung der WER, MER, WIL und WIP wird von der Library *jiwer* [42] übernommen. Die WER Implementation aus der originalen Pipeline wurde angesichts schlechter Performance (Laufzeit) verworfen. Es gibt Unterschiede der WER von *jiwer* und der eigenen Implementation. Die Gründe dafür sind unbekannt. Es könnten Fehler in einer oder beiden WER-Implementationen existieren. Da diese Arbeit ohnehin alle Experimente von der STT-Ensemble-Learning - Analyse wiederholt, wurde diesen Unterschieden nicht weiter nachgegangen.
Obwohl die Metriken MER, WIL und WIP in dieser Arbeit keine weitere Anwendung finden, sind sie in der Pipeline integriert.

4.1.2. SemDist

Für SemDist existiert keine Python-Library. Daher wurde diese Metrik neu implementiert. Als einzig wichtige Voraussetzung für SemDist wird Mean-Pooling auf den Embedding-Vektoren gennant. Wie in Kapitel 3.5 erwähnt, dient die SemDist zur semantischen Bewertung.

Mean Pooling

Die Dimensionen der Sentence-Embeddings, welche mit einem transformerbasierten Modell erzeugt wurden, sind typischerweise abhängig von der Länge der Eingabe. Damit solche Vektoren vergleichbar sind, sollen diese jedoch dieselben Dimensionen aufweisen. Typisch sind Ziel-Vektoren der Länge 768 oder 384. *Mean Pooling* beschreibt eine einfache Art, die Dimension eines Vektors zu reduzieren. Ein Sliding-Window der Grösse

n wird über den Vektor gelegt. In jedem Schritt wird der Durchschnitt der n-Werte im Window gebildet und in den Zielvektor übertragen. Dann wird das Window um n-Stellen verschoben und der Vorgang wiederholt. n wird anhand der Zieldimension bestimmt.

Implementation

SemDist schreibt nicht explizit vor, wie die Sentence-Embeddings zu bilden sind. Die originale Implementation verwendet eine modifizierte und selbst trainierte Version von *roBERTa*. Für die Implementation wurde das vortrainierte *Distilroberta* Modell gewählt. Dieses Modell basiert auf roBERTa und kann in Python mit dem *SentenceTransformers*-Package mühelos verwendet werden. Ausserdem verwendet die Kombination von *Distilroberta* und *SentenceTransformers* bereits Mean-Pooling. Mit den Sentence-Embeddings und der Formel 2.3 konnte SemDist problemlos implementiert werden.

Resultat

SemDist wurde zum einen in einem Single- und zum anderen in einem Bulk-Mode implementiert. Der Single-Mode hat einen grossen Overhead und hat deshalb eine erhöhte Laufzeit. Der Bulk-Mode unterstützt das Berechnen von Embeddings auf Nvidia-GPUs und erlaubt ein zügiges Berechnen von vielen SemDist-Werten.

4.2. Alignment UI

Ein Alignment auf dem ganzen CEASR Korpus umfasst eine grosse, unübersichtliche Datenmenge von über 25'000 Utterances. Das Alignment UI, welches in dieser Arbeit entwickelt wird, visualisiert das Alignment zwischen einer beliebigen Anzahl von Systemen im Bezug auf eine Basishypothese. Das Tool hilft dadurch, in Alignments problematische Stellen mithilfe von Farbkodierungen zu finden. Die problematischen Stellen des Alignments können in einer detaillierten Sicht, in der auch die Wörter dargestellt sind, beurteilt werden. Das Hochladen und Betrachten von ganzen Korpora soll einen guten Überblick über die Qualität des Alignments liefern. Durch das Alignment der Vorhersage mit der Referenz kann das Tool auch bei der Analyse von Fehlern in Vorhersagen eingesetzt werden.

4.2.1. Funktionsumfang

Das Alignment UI soll den folgenden Funktionsumfang ausweisen:

- Upload und Anzeige von Alignment File
- Darstellung einer Utterance in Punktemap und im Detail auf Wortebene
- Anzeige und allfällige Berechnung der WER auf Utterance Ebene
- Anzeige von Statistiken zu den Systemen

Als erster Schritt wurde ein Mockup (Abbildung 9) des Tools erstellt, welches die wesentlichen Features grafisch aufzeigt. Das Mockup dient als Basis für die Implementierung.

Abbildung 9.: Alignment UI Mockup

4.2.2. Implementation

Als zweiter Schritt wird das Alignment UI auf Basis des Mockups 9 implementiert. Für die Implementierung wird das Frontend Framework Angular [43] verwendet. Die Anwendung ist als Single Page Application entwickelt. Für die Verwaltung der grossen Datenmengen der Alignments setzt das Tool auf NgRx [44].

4.2.3. Resultat

Die Abbildung 10 zeigt das fertige Alignment UI. Ein Alignment JSON File, welches vom n-Align Algorithmus aus Kapitel 4.3 herausgegeben wurde, kann leicht modifiziert, über eine Drag&Drop Zone hochgeladen werden. Die Modifikation am Alignment File betrifft nur die Datenstruktur der Systeme. Ein Beispiel der genauen JSON Struktur kann dem Readme im Github Projekts A.3 entnommen werden. Das Alignment UI kann mit einer beliebigen Anzahl an angeglichenen Systemen umgehen. Zu jeder Utterance kann die WER im Vergleich zur Primärhypothese mitgegeben werden. Falls dieses Feld nicht abgefüllt ist, wird es nach dem Hochladen im Alignment UI berechnet. Die WER hilft, als Sortierkriterium fehlerhafte Alignments in einer grossen Datenmenge einfacher zu finden. Das Alignment UI kann bei Alignments, welche drei Systeme von dem ganzen CEASR Korpus enthalten, langsam reagieren, aufgrund der Datenmenge in diesen Dateien.

Abbildung 10.: Alignment UI

4.2.4. Fazit

Das Alignment UI hilft sowohl in der Analyse von Alignments, wie auch in der Auswertung von Vorhersagen mit einer übersichtlichen grafischen Oberfläche. Das Tool konnte in dieser Arbeit bei mehreren Aufgaben zu diesen Zwecken eingesetzt werden. Zum einen wurde das Tool für die Analyse im Kapitel 4.4 zur Alignmentqualität verwendet. Zum anderen wurde das Tool beim Entwickeln vom n-Align Algorithmus in Kapitel 4.3 zur Fehlerdiagnose eingesetzt.

4.3. n-Align Algorithmus

Die bestehende Lösung kann nur drei Hypothesen beziehungsweise Texte aneinander angleichen. Diese Limitierung soll in Hinsicht zweierlei Probleme aufgehoben werden. Zum einen könnte es in der Zukunft auch möglich sein, mehr als drei Systeme zu verwenden, zum Beispiel, wenn ASR-Systeme günstiger zu verwenden werden. Zum anderen bietet ein Alignment von n-Hypothesen mehr Möglichkeiten für die Datenanalyse und Auswertung. Hypothesen könnten zum Beispiel an die Ground Truth (Referenzalignment) oder an die Vorhersage einer Ensemble-Methode angeglichen werden. Im Folgenden wird ein Algorithmus vorgeschlagen und implementiert, welcher beliebig viele Alignment-Paare kombinieren kann. Diese Alignment-Paare werden identisch zum vorhandenen Triple Ali-

gnment aus Kapitel 2.7.2 gebildet. Auf eine Implementierung nach ROVER wurde aus Zeit- und Komplexitätsgründen verzichtet.

4.3.1. Implementation

Inspiriert von den ROVER-Regeln [35], kombiniert der n-Align Algorithmus Paare von Alignments anhand vordefinierter Regeln. Weil die bestehende Implementation nicht mit WTN arbeitet, können die Regeln nicht 1:1 angewendet werden. Es lassen sich aber ähnliche Regeln formulieren:

1. Korrekte Werte und Substitutionen verhalten sich gleich. Sie können übernommen werden.
2. Insertions werden separat behandelt. Falls an einer Position mehrere Hypothesen eine Insertion aufweisen, so werden diese jeweils kombiniert.
3. Löschungen ergeben einen leeren Eintrag

4.3.2. Resultat

Abbildung 11 zeigt schematisch die Kombination von $n-1$ Alignment-Paaren. Ein wichtiger Punkt des Algorithmus bildet Schritt drei in Abbildung 11. In diesem Schritt wird sichergestellt, dass von jeder Hypothese das korrekte Gegenstück in der Primärhypothese verglichen wird.

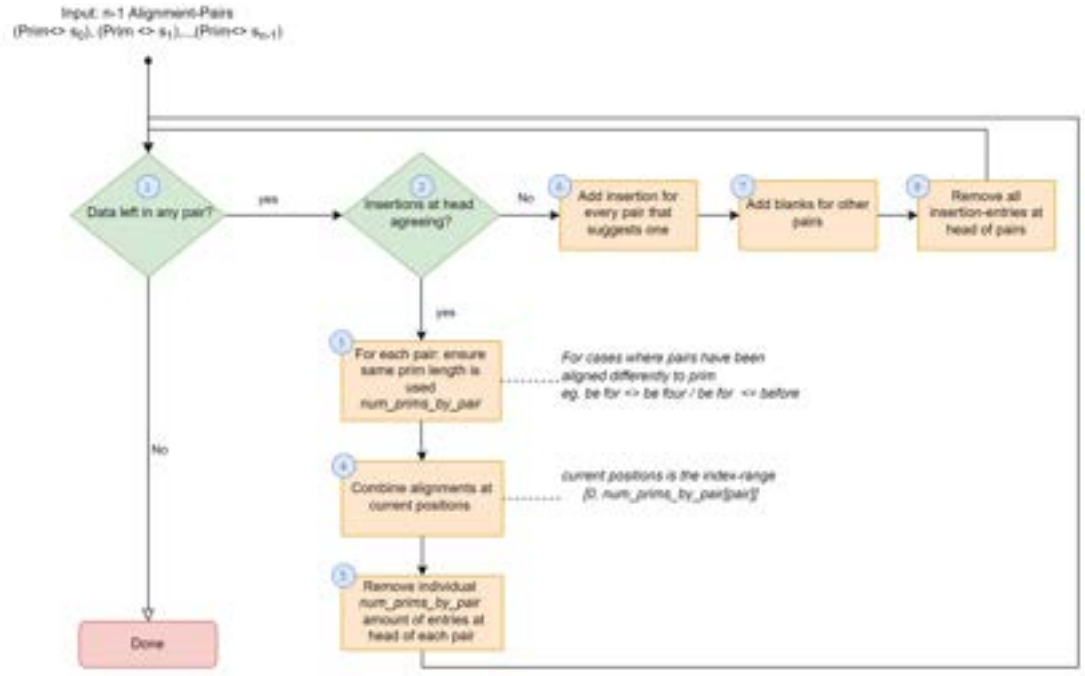


Abbildung 11.: n-Align Algorithmus

Der neue n-Align Algorithmus liefert für $n = 3$ das exakt selbe Alignment wie der ursprüngliche Triple-Alignment Algorithmus. Der neue Algorithmus schreibt das Alignment ausserdem in ein neues, flexibleres JSON Format, welches vom Alignment UI (siehe Kapitel 4.2) verarbeitet werden kann. Als Folge von diesem neuen Format wurde auch für den Rest der Pipeline ein Refactoring auf das neue Format durchgeführt.

Beispiel

Im Folgenden wird das Alignment von drei Hypothesen mit n-Align an einem Beispiel veranschaulicht. Ausgangslage sind die zwei Alignment-Paare

I	*saw*	*them*		be for
I	*saw*	*them*	*and*	*before*
correct	*correct*	*correct*	*insertion*	*substitution*

und

I	*saw*	*them*	*be*	*for*
	saw	*them*	*be*	*for*
deletion	*correct*	*correct*	*correct*	*correct*

Gemäss Abbildung 11 wird nun über die Paare iteriert, solange Elemente vorhanden sind. Die erste Iteration verläuft wie folgt:

Paar 1 @ Head	**Paar 2 @ Head**
I	*I*
I	(leer)
correct	*deletion*

↓

Schritt 2: Yes (keine Insertion)

↓

Schritt 3: Beide Paare haben «I» als Primärhypothese. Somit ist für beide Paare $\text{num\_prims}_{p1} = \text{num\_prims}_{p2} = 1$

↓

Schritt 4: kombinieren:

I
{ text: ”I”, type: "Correct"}
{ text: ””, type: "Deletion"}

↓

Schritt 5: num_prims (also 1) Einträge aus den originalen Paaren löschen. Neue Listen:

saw	*them*		be for		*saw*	*them*	*be*	*for*
saw	*them*	*and*	*before*	**und**	*saw*	*them*	*be*	*for*
correct	*correct*	*insertion*	*substitution*		*correct*	*correct*	*correct*	*correct*

Die nächsten zwei Iterationen verlaufen identisch. Anschliessend ist jedoch die Insertion von «and» an erster Stelle. In Schritt 3 sind sich so die Insertions nicht mehr einig:

Paar 1 @ Head	**Paar 2 @ Head**
(leer)	be
and	*be*
Insertion	*correct*

Folglich muss für die Insertion zuerst gemäss Schritt 6 bis 8 eine Insertion für Paar 1 hinzugefügt und der Insertion-Eintrag aus Paar 1 entfernt werden. Die nächste Iteration ist die letzte. Diese läuft fast identisch wie die ersten Iterationen ab.

Paar 1 @ Head	**Paar 2 @ Head**
be for	*be*
before	*be*
substitution	*correct*

Wie ersichtlich ist, werden jedoch nicht mehr dieselben Anteile von der Primärhypothese verglichen. Paar 2 vergleicht nur «be», Paar 1 allerdings «be for». Somit muss in Schritt 3 $num\_prims_{p2} = 2$ gesetzt werden, bevor kombiniert wird. Schlussendlich werden also die folgenden Alignments kombiniert:

Paar 1	**Paar 2**	
be for	*be*	*for*
before	*be*	*for*
substitution	*correct*	*correct*

Weil $num\_prims_{p2} = 2$ für gesetzt wurde, werden im Schritt 5 nun zwei Einträge aus der Liste von Paar 2 gelöscht, womit nun beide Listen leer sind und somit der Algorithmus fertig ist. Das Endresultat aus diesem Beispiel ist im Anhang A.2 zu finden.

4.3.3. Fazit

Mit n-Align können nun beliebig viele Hypothesen angeglichen werden. Unter anderem ermöglicht dies eine Implementierung des Best-Possible (siehe Kapitel 4.5), da es nun möglich ist, drei oder mehrere Systeme jeweils an die originale Referenz anzugleichen. So wird die Referenz mit den Hypothesen der verschiedenen Systeme vergleichbar gemacht, was Rückschlüsse auf die bestmögliche Performance zulässt.

4.4. Qualitätsbeurteilung des Alignments

Die Qualität des Alignment kann die Ensemble-Methoden massgeblich beeinflussen. Da das Alignment die Grundlage der Ensemble-Methoden bildet, soll festgestellt werden, ob

das bestehende Alignment gut funktioniert. Die Qualität eines Alignment ist schwierig zu beurteilen. Es sind keine Metriken oder Datensets bekannt, mit welchen ein Alignment-Algorithmus evaluiert werden kann. Theoretisch lässt sich die Qualität ebenfalls schwer bewerten [35]. Für ein Alignment zwischen zwei Sequenzen ist bekannt, dass das Alignment optimal ist. Allerdings ist ein iterativ erhaltenes Alignment, so wie es vorliegt, kein optimales Alignment. Die Algorithmen für ein optimales Alignment zwischen mehreren Sequenzen existieren zwar, jedoch handelt es sich um ineffiziente Lösungen mit der Komplexität $\mathcal{O}(c^n)$.
Zur Überprüfung des Alignment wurde auf der n-Align Ausgabe von den Systemen B7, C2 und D2 eine Auswertung gemacht. Das Alignment funktioniert nicht, falls aus einem Wort einer Hypothese eine Substitution oder Insertion gemacht wurde, statt dass das Wort als Correct mit dem gleichen Wort vorher oder nachher in der Primärhypothese angeglichen wurde.
Auf dem gesamten CEASR Korpus (25094 Utterances) wurden 666 Utterances gefunden (2,6 %), bei welchen diese Konstellation vorhanden ist. Die Fälle wurden von Hand mit dem Alignment UI beurteilt. Dabei konnten zwei Fälle klassifiziert werden:

1. Gleiches Wort aufeinanderfolgend in einer Hypothese
2. fehlerhaftes Alignment von drei Hypothesen

97 % sind Fälle vom Typ 1 wie in der Abbildung 12. Da dabei das Wort mehrmals vorkommt, aber auch einmal richtig angeglichen wurde, ist dies kein Fehler. Bei 3 % ist jedoch ein Alignment herausgekommen, dass als fehlerhaft klassifiziert werden kann. In der Abbildung 13 ist beim zweiten und dritten System «it's» mehrmals vorhanden, die drei «it's» aus der zweiten Hypothese sollten mit denen aus der dritten Hypothese angeglichen sein. Da dieses Problem bei weniger als 0,1 % der Utterances auftritt, wurde im Rahmen dieser Arbeit nichts weiter am n-Align Algorithmus aus Kapitel 4.3 angepasst.

Abbildung 12.: Alignment doppelte Wörter

Abbildung 13.: Alignment Fehler

4.5. Möglichkeiten im CEASR Korpus - Best-Possible und Systemauswahl

Ensemble-Methoden können grundsätzlich keine besseren Resultate erreichen, als was die zugrunde liegenden Daten hergeben. Eine Analyse soll zeigen, was für eine Verbesserung eine Ensemble-Methode auf dem CEASR Korpus theoretisch erreichen kann. Dieser Ansatz wird Best-Possible genannt.

4.5.1. CEASR-Heatmap

Als erster Schritt muss evaluiert werden, welche Systeme aus CEASR überhaupt am besten funktionieren. Die Informationen aus dem CEASR-Paper reichen nicht aus, um die korrespondierenden Daten im CEASR Korpus zu identifizieren. Die im Paper erwähnten Abkürzungen stimmen nicht mit den Dateinamen in CEASR überein. Eine Übersetzungstabelle existiert ebenfalls nicht. Daher wurden alle Metriken aus Kapitel 4.1 auf allen Korpora und Systemen aus CEASR berechnet, gemittelt und in einer Heatmap zusammengefasst.

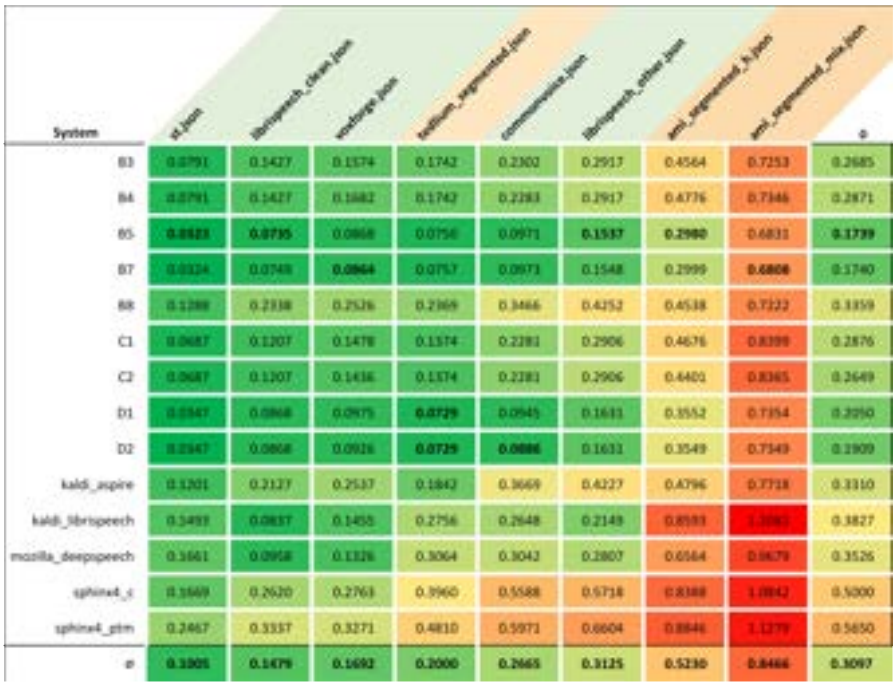


Abbildung 14.: Heatmap WER

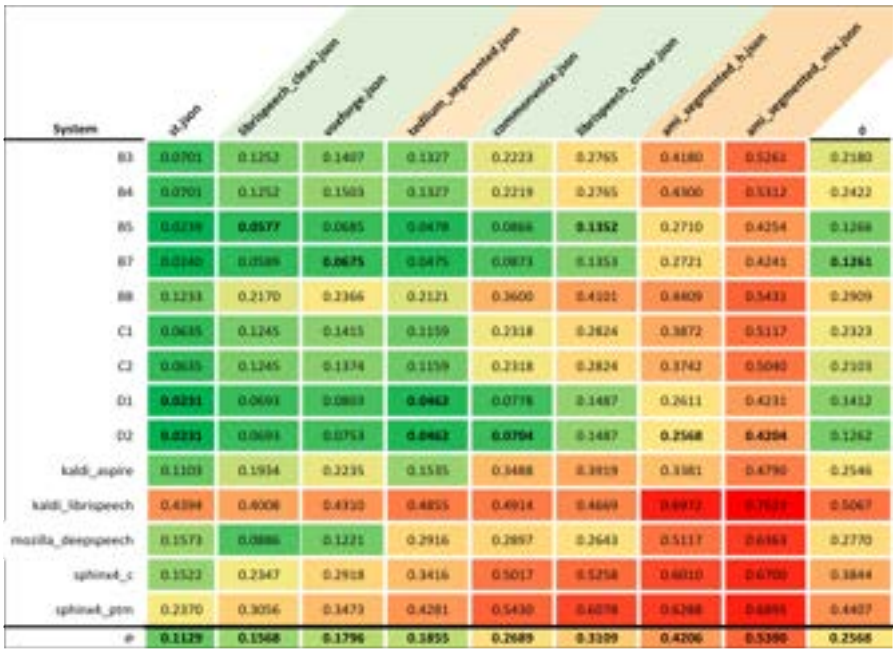


Abbildung 15.: Heatmap SemDist

In den Heatmaps ist ersichtlich, dass die drei Systeme B5, B7 und D2 dominieren und

immer eines dieser drei Systeme den besten Wert erreicht. Das System B ist in der WER etwas besser als System D, dafür ist System D semantisch leicht besser als System B. Schlussendlich müssen drei Systeme für das Best-Possible ausgewählt werden. Am sinnvollsten erschien es, die Systeme B7, C2 und D2 zu verwenden. Diese Auswahl wurde aus zwei Gründen getroffen. Zum einen stammen Systeme mit dem gleichen Buchstaben vom gleichen Hersteller. Diese können tendenziell schlechter kombiniert werden, weil sie sich kaum unterscheiden und so die Diversität des Ensembles nicht gegeben ist. Zum anderen wurde B5 zugunsten von B7 weggelassen. B7 ist auf spontaner Sprache im Durchschnitt leicht besser als B5. Da der Stand der Technik auf nicht spontaner Sprache auch ohne Ensemble Learning schon hervorragend ist, wird so der Fokus korrekterweise auf spontane Sprache gelegt.

4.5.2. Berechnung des Best-Possible

Das Best-Possible kann als eine Ensemble-Methode angesehen werden, welche die Referenz verwendet, um die bestmögliche Hypothese zu erzeugen. Für die Berechnung des Best-Possible wird vom n-Align Algorithmus Gebrauch gemacht. Die Referenz wird als Basishypothese angeschaut. Alle CEASR-Hypothesen der Systeme B7,D2 und C2 werden an die Referenz angeglichen. Das resultierende Vierer-Alignment kann dann verwendet werden, um die bestmögliche Hypothese für jede Utterance zu bilden. Dazu wird Wort für Wort geprüft, ob in einer der drei Hypothesen das geforderte Wort der Referenz enthalten ist. Falls ja, wird dieses in die neue Hypothese übernommen. Ansonsten wird das aktuelle Wort ignoriert und weiter gemacht. Da dies auf «string equality» basiert, wird streng genommen nur die WER minimiert. Schlussendlich werden für die Referenz und die neue Hypothese die Metriken aus Kapitel 4.1 berechnet und der Durchschnitt über den Korpus gebildet. Daraus resultiert Abbildung 16 (WER) respektive Abbildung 17 (SemDist).

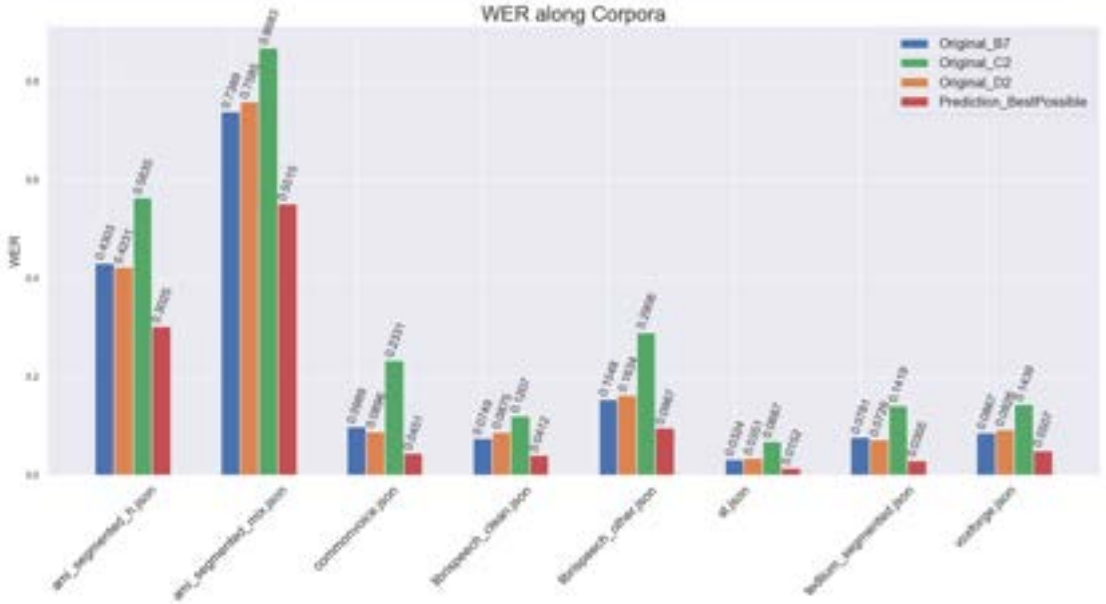


Abbildung 16.: Best-Possible WER

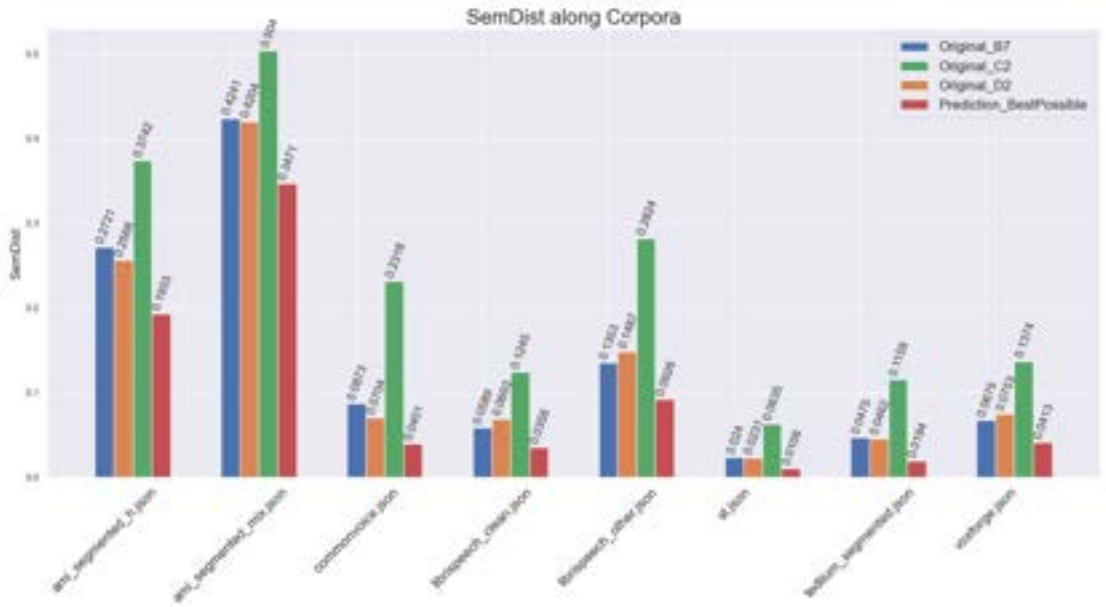


Abbildung 17.: Best-Possible SemDist

4.5.3. Resultat

Die Abbildungen 16 bzw. 17 zeigen, dass Ensemble Learning durchaus Potenzial hat. Gerade bei spontaner Sprache könnte die WER und SemDist deutlich verbessert werden.

So kann die WER auf den AMI-Korpora um 13 % resp. 18 % verbessert werden. Auf den anderen Korpora ist die mögliche Verbesserung kleiner. Allerdings sind auch dort WER-Verbesserungen von bis zu 5 % und SemDist Verbesserungen von 2 % bis 5 % möglich.

4.5.4. Fazit

Mit dem Best-Possible Ansatz konnte gezeigt werden, wie gut die beste Lösung sein könnte. Da «string equality matching» verwendet wurde, wird die WER minimiert, nicht aber zwingend die SemDist. Allerdings ist das Potenzial der SemDist auch ohne semantisches Matching gross, wie Abbildung 17 zeigt, weshalb kein Best-Possible für SemDist erstellt wurde. Abschliessend muss jedoch gesagt werden, dass der aktuelle Stand der Technik schon besser als das Best-Possible von CEASR aus 2019 ist. Für LibriSpeech-Clean ist bekannt, dass w2v-BERT eine WER von 1,4 % erreicht. Nachforschungen von Kanda et al. [11] legen ausserdem nahe, dass der Stand der Technik auf dem AMI Korpus auch schon unter 30 % WER liegt.

4.6. Auswahl der Primärhypothese

Für das Alignment muss eine Primärhypothese ausgewählt werden. Dieses Kapitel untersucht, ob sich die Auswahl der Primärhypothese markant auf die WER oder SemDist einer Ensemble-Methode auswirkt. Insbesondere soll festgestellt werden, ob durch eine schlechtere Primärhypothese die Ensemble-Methode markant schlechter wird. Ein Einfluss auf das Best-Possible kann ausgeschlossen werden, da bei Best-Possible immer die Referenz als Primärhypothese angesehen wird.
Für das Experiment wird einmal B7 und einmal C2 als Primärhypothese ausgewählt. Anders als bei den meisten anderen Analysen wird hier der gesamte CEASR Korpus verwendet. Für jede Utterance wurden die Alignments erstellt. Auf diesen Alignments wird schlussendlich die Voting-Methode (siehe Kapitel 5.2) angewendet, um den Einfluss der Primärhypothese auf das Resultat abschätzen zu können.

4.6.1. Resultat

Ein diff-Vergleich zwischen den zwei Alignments zeigt, dass die Alignments deutlich unterschiedlich sind. Ein genauerer Blick zeigt, dass die meisten Unterschiede aus der Reihenfolge in der Datenstruktur entstehen. Es gibt aber auch gänzlich unterschiedliche Einträge. Vor allem betroffen sind Insertions und Deletions, welche nun vertauscht wurden. Nach diesen Erkenntnissen überrascht es nicht, dass die Resultate nicht identisch sind. Die WER mit C2 als Primärhypothese ist ein wenig höher als mit B7 als Primärhypothese ist. Der Unterschied ist jedoch klein und auf einigen Korpora (st, voxforge) ist das C2-Alignment sogar marginal besser. Die SemDist vom B7-Alignment ist ebenfalls fast durchgehend besser als C2. Allerdings sind die Unterschiede auch hier nur bescheiden.

Corpus	Original B7	Original C2	Original D2	Prediction C2	Prediction B7	Delta B7-C2
Ami_segmented_h	0,4303	0,5635	0,4231	0,43	0,4234	-0,66 %
Ami_segmented_mix	0,7389	0,8683	0,7585	0,7242	0,7171	-0,71 %
Commonvoice	0,0989	0,2331	0,0896	0,1017	0,0943	-0,74 %
Librispeech_clean	0,0749	0,1207	0,0875	0,0714	0,0711	-0,03 %
Librispeech_other	0,1548	0,2906	0,1634	0,1594	0,1538	-0,56 %
St	0,0324	0,0687	0,0351	0,0315	0,0321	0,06 %
Tedlium_segmented	0,0781	0,1419	0,0729	0,0676	0,0621	-0,55 %
Voxforge	0,0867	0,1439	0,0926	0,0799	0,0807	0,08 %

Tabelle 1.: Vergleich WER mit Voting. Rot = C2 als Prim ist besser als B7

Corpus	Original B7	Original C2	Original D2	Prediction C2	Prediction B7	Delta B7-C2
Ami_segmented_h	0,2721	0,3742	0,2568	0,2751	0,2708	-0,43%
Ami_segmented_mix	0,4241	0,504	0,4204	0,4278	0,4198	-0,8%
Commonvoice	0,0873	0,2318	0,0704	0,0888	0,0831	-0,57%
Librispeech_clean	0,0589	0,1245	0,0693	0,0598	0,0597	-0,01%
Librispeech_other	0,1353	0,2824	0,1487	0,1459	0,1414	-0,45%
St	0,024	0,0635	0,0231	0,022	0,0225	0,05%
Tedlium_segmented	0,0475	0,1159	0,0462	0,0421	0,0385	-0,36%
Voxforge	0,0675	0,1374	0,0753	0,0658	0,0657	-0,01 %

Tabelle 2.: Vergleich SemDist mit Voting. Rot = C2 als Prim ist besser als B7

4.6.2. Fazit

Es zeigt sich, dass die Wahl der Primärhypothese keine entscheidende Rolle für die relative Verbesserung spielt. Schaut man sich die absolute Auswirkung an, so führt die Wahl von der schlechteren Primärhypothese jedoch oft zu keiner Verbesserung mehr. Es ist somit zu empfehlen, das beste System als Primärhypothese festzulegen, da dies zu einer leicht tieferen WER führt. Ausserdem können Ensemble-Methoden so als Fallback immer die Primärhypothese auswählen und erhalten so tendenziell die bessere Lösung.

4.7. Anzahl Systeme

Grundsätzlich gilt gemäss Kapitel 3.2 eine Einschränkung auf drei Systeme. Dieses Kapitel soll untersuchen, ob von mehr als drei Systemen profitiert werden könnte. Dazu wurden zusätzlich zu den Systemen B7,D2 und C2 auch noch B3 und C1 verwendet. Mit dem n-Align Algorithmus wurde ein Fünfer Alignment mit B7 als Primärhypothese erstellt. Ausserdem wurde ein Sechser Alignment mit der Referenz als Primärhypothese erstellt. Anschliessend wurden die Alignments nach der Methode im Kapitel 5.1 50:50 aufgeteilt, damit die Utterances in allen Vergleichen identisch sind. Aus dem Referenz-Alignment wird ein Best-Possible (siehe Kapitel 4.5) für fünf Systeme erstellt. Auf das normale Alignment wird die Voting-Methode aus Kapitel 5.2 angewendet. So soll zum einen die Theorie und zum anderen die Praxis erforscht werden.

	Original					Best-Possible	
corpus	**B7**	**C2**	**D2**	**B3**	**C1**	**3 Sys**	**5 Sys**
ami_segmented_h	0,4319	0,5718	0,4287	0,6403	0,585	0,3044	0,2981 (+0,63 %)
ami_segmented_mix	0,7589	0,8821	0,763	0,8319	0,8877	0,5673	0,5271 (+4,02 %)
commonvoice	0,1007	0,2373	0,0917	0,2528	0,2373	0,0461	0,0439 (+0,22 %)
librispeech_clean	0,0739	0,122	0,0856	0,1449	0,122	0,0393	0,0364 (+0,29 %)
librispeech_other	0,1566	0,2884	0,1643	0,2962	0,2884	0,0991	0,0942 (+0,49 %)
st	0,0349	0,071	0,0392	0,0829	0,071	0,0181	0,0165 (+0,16 %)
tedlium_segmented	0,0776	0,1429	0,0719	0,1807	0,1429	0,0296	0,0273 (+0,23 %)
voxforge	0,0796	0,137	0,0888	0,1485	0,1399	0,0451	0,0413 (+0,38 %)

Tabelle 3.: Vergleich Best-Possible WER für 3 und 5 Systeme

	Original					WER Voting		Improvements	
corpus	**B7**	**C2**	**D2**	**B3**	**C1**	**3 Sys**	**5 Sys**	**3 Sys**	**5 Sys**
ami_segmented_h	0,4319	0,5718	0,4287	0,6403	0,585	0,4234	0,4755	0,53 %	-4,68 %
ami_segmented_mix	0,7589	0,8821	0,763	0,8319	0,8877	0,7171	0,7686	4,18 %	-0,97 %
commonvoice	0,1007	0,2373	0,0917	0,2528	0,2373	0,0943	0,1398	-0,26 %	-4,81 %
librispeech_clean	0,0739	0,122	0,0856	0,1449	0,122	0,0711	0,0802	0,28 %	-0,63 %
librispeech_other	0,1566	0,2884	0,1643	0,2962	0,2884	0,1538	0,1941	0,28 %	-3,75 %
st	0,0349	0,071	0,0392	0,0829	0,071	0,0321	0,0422	0,28 %	-0,73 %
tedlium_segmented	0,0776	0,1429	0,0719	0,1807	0,1429	0,0621	0,0865	0,98 %	-1,46 %
voxforge	0,0796	0,137	0,0888	0,1485	0,1399	0,0807	0,0867	0,6 %	-0,71 %

Tabelle 4.: Vergleich der Voting WER für 3 und 5 Systeme

Tabelle 3 zeigt den Vergleich des Best-Possible für die fünf Systeme B7, D2, C2, B3 und C1, Im Vergleich zu nur drei Systemen sieht man, dass die WER des Best-Possible mit den fünf Systemen durchgehend leicht besser ist. In Tabelle 4 wird die WER für die Voting-Methode verglichen. Es zeigt sich, dass die WER bei Voting für 5 Systeme durchgehend höher als für 3 Systeme ist.

Fazit

Theoretisch zeigt Tabelle 3, dass mit 5 Systemen bessere Resultate als mit drei Systemen erzielt werden könnten. Es handelt sich allerdings mit Ausnahme von ami_segmented_mix nur um Verbesserungen von unter 1 %. Die geringe theoretische Verbesserung deutet darauf hin, dass die Systeme B3 und C1 nur in den seltensten Fällen einen Beitrag leisten können. Schlussendlich heisst das, dass in den meisten Fällen, wenn C1 und B3 richtig liegen, die anderen Systeme auch richtig liegen. Die Diversität des Ensembles mit 5 Systemen ist somit nicht ausreichend. Die Praxis in Tabelle 4 zeigt, dass die WER deutlich schlechter wird, sobald schlechtere Systeme in das Ensemble aufgenommen werden. Die schlechteren Systeme verfälschen die Abstimmung im Voting-Algorithmus. Der naive Ansatz, noch andere Systeme aus CEASR zu verwenden, lohnt sich somit nicht. Möglicherweise könnten die Informationen aus B3 und C1 als Tiebreaker oder mit tieferer Gewichtung verwendet werden. Dieser Ansatz wurde allerdings nicht mehr weiter verfolgt.

4.8. Feature Extraction

Ein möglicher Verbesserungsansatz könnte Machine Learning sein. Es könnte auf Wortbasis gelernt werden, in welchen Situationen welches System ausgewählt werden soll. Dazu braucht es ein Datenset, um die Systeme gegeben ihrer Outputs zu klassifizieren. Diese Idee resultiert direkt aus dem Best-Possible Ansatz. In diesem Kapitel wird beschrieben, wie die Features für ein solches Datenset für die besten Systeme aus CEASR (B7, D2, C2) aussehen und extrahiert werden könnten. Dafür wurde ein FeatureExtractor-Tool geschrieben, welches in Anhang A.5 dokumentiert ist. Dieses FeatureExtractor-Tool wurde hinsichtlich der bestehenden Pipeline kompatibel implementiert, sodass in allen Analysen Features extrahiert werden könnten.

4.8.1. Labels

Die grundlegende Idee des Labelings ist identisch zum Best-Possible Ansatz. Der neue n-Align Algorithmus wird ausgenutzt, um ein Referenz-Alignment von B7,D2,C2 und der Ground Truth aus CEASR zu erhalten. Durch das Referenz-Alignment ist bekannt, welche Systeme richtig liegen und welche nicht. Dies entspricht genau dem Label im Datenset. Man könnte also sagen, dass mit diesem Datenset ein «Best-Possible-Estimator» trainiert werden kann. Für die Systeme B7,D2,C2 werden alle Kombinationen als Label definiert.

- none
- B7
- D2
- C2
- B7_D2
- B7_C2
- D2_C2
- B7_D2_C2

4.8.2. Features

Aus den bereits implementierten Ensemble-Methoden, insbesondere dem Voting, geht hervor, dass die gegenseitige Übereinstimmung der Systeme einen signifikanten Einfluss haben kann. Im CEASR-Korpus sind ausserdem diverse Metadaten wie Alter, Geschlecht, Sprechgeschwindigkeit oder Dialekt verfügbar. Gerade der Dialekte oder Akzente können einen grossen Einfluss auf die Qualität eines Transkripts haben [45][46]. Eine weitere Hypothese war es ausserdem, dass Wort- und Satzlänge einen Einfluss haben könnten. Längere Wörter und Sätze ermöglichen jeweils, mehr Fehler zu machen. Aus diesen Überlegungen ergeben sich die Features in Tabelle 5.

Feature	Datentyp	Beschreibung
Agreement**	boolean	Übereinstimmung von zwei Systemen (exakt identische Zeichenfolge)
ContextAgreement**	boolean	Übereinstimmung von zwei Systemen (Prozentanteil korrekter Kontextwörter +/- 5 Wörter)
StemmedAgreement**	boolean	Übereinstimmung von zwei Systemen (Stemming, dann wie Agreement)
WordSim**	boolean	semantische Übereinstimmung - Word-Embeddings e_1, e_2 $cos\_sim(e_1, e_2) > 0,75$
SentenceLength*	integer	Satzlänge der gesamten Hypothese, in der das Wort vorkommt
WordLength*	integer	Wortlänge
	aus CEASR	
RecordingDevice	enum	Gerät, mit der eine Utterance aufgenommen wurde. (Beispielsweise cell-phone)
Geschlecht	enum	Geschlecht der SprecherIn
Dialect	enum	der gesprochene Dialekt (Beispielsweise en-US)
Accent	enum	Flag, ob die sprechende Person ein Dialekt hat oder nicht
SpeakingRate	float	Sprechgeschwindigkeit der Person
Age	integer	Alter der sprechenden Person
TypeOfSpeech	enum	Art des ASR-Korpus (spontane Sprache, semi-spontane Sprache oder vorgelesene Sprache)

Tabelle 5.: Features aus CEASR und Referenzalignment. Features mit * existieren für jedes System. Features mit ** existieren für jedes paar von System

4.8.3. Extraktion der Features

Für jedes Wort in jeder Utterance im Alignment werden die in Tabelle 5 erwähnten Features extrahiert. Die meisten Features sind selbsterklärend. Es gibt jedoch einige Eigenheiten oder Implementationsdetails.

CEASR Features

Features aus der CEASR-Kategorie werden aus den originalen CEASR-Korpora extrahiert. Die Metadaten in CEASR stammen aus den originalen Korpora. Es sind nicht alle Metadaten in jedem Korpus vorhanden. Ähnlich wäre es allerdings auch in einem produktiven Setup denkbar. Vor der Erstellung eines Transkripts könnte der Benutzer Metadaten angeben, welche jedoch nicht immer bekannt sind. Eine Datenauswertung zeigt, welche Daten in welchen Korpora verfügbar sind.

Korpus	rec. device	gender	dialect	accent	rate	age	ToS
ami_segmented_h	x	x	(x)	x	x	-	x
ami_segmented_mix	x	x	(x)	x	x	-	x
commonvoice	x	(x)	(x)	(x)	x	(x)	x
librispeech_clean	-	x	x	-	x	-	x
librispeech_other	-	x	x	-	x	-	x
st	x	x	x	-	x	-	x
tedlium_segmented	-	x	x	-	x	-	x
voxforge	-	x	(x)	(x)	x	-	x

Tabelle 6.: Vorhandene Metadaten in CEASR für Feature Extraction; x = Vorhanden, (x) = Teilweise vorhanden, - = nicht vorhanden

WordSim**

Das WordSim Feature gibt an, ob es sich bei zwei Wörtern um semantisch identische Wörter handelt. Dazu werden Word-Embeddings e_1, e_2 für die Wörter der zwei verglichenen Systeme gebildet.
WordSim wurde anhand folgender Formel definiert:

$$WordSim = \begin{cases} 1, & \text{wenn } cos\_sim(e1, e2) \geq 0,75 \\ 0, & \text{ansonsten} \end{cases} \tag{4.1}$$

Ähnlich wie bei SemDist, kann die Cosinus-Ähnlichkeit als Ähnlichkeitsmass der zwei Embeddingvektoren angeschaut werden

WordSim-Cuttoff

Der Cutoff von 0,75 in Formel (4.1) wurde durch eine Datenanalyse (Abbildung 18) erhalten. Für eine Liste von ca. 8000 Singular-Plural-Tupeln und den Zeitformen von

ca. 2000 Verben wurde die paarweise Cosinusdistanz statistisch ausgewertet. In einem ersten Schritt wurde das bekannte fastText-Modell [29] verwendet. Die Datenauswertung zeigt jedoch, dass ein Cutoff eher tief bei $Q_1 = 0,47$ angesetzt werden müsste, um eine gute Anzahl Matches zu erhalten. Um dies besser zu beurteilen, wurden in einem nächsten Experiment die Tupel jeweils um einen Eintrag verschoben, sodass Singular / Plural und Zeitform nicht mehr übereinstimmen. Mit diesem «Out-of-Sync» Ansatz sollen Fehlklassifikationen evaluiert werden. Abbildung 18 zeigt, dass für einen Cutoff von $Q_1 = 0,47$ für weniger als 25 % der Daten *WordSim = 1* gelten würde. Da dies relativ hoch ist, wurde als Vergleich *DistilroBERTa*, dasselbe Modell, welches für die SemDist verwendet wurde, hinzugezogen. Es zeigt sich, dass in diesem Fall ein Q_1-Cut-off bei 0,75 % liegen würde. Bei genauerer Betrachtung fällt auf, dass die Ausreisser meistens Vergleiche zwischen dem Infinitiv und irregulären Perfect Continuous Zeitformen sind. So ist zum Beispiel $cos\_sim(bring, brought) = 0,52$. Der «Out-of-Sync» Ansatz ergibt bei *DistilroBERTa* ein ähnliches Bild wie bei fastText. Schlussendlich wurde ein Cut-Off von 0,75 mit *DistilroBERTa* festgelegt. Die Datenpunkte liegen bei *DistilroBERTa* enger beieinander und Ausreisser lassen sich besser erklären.

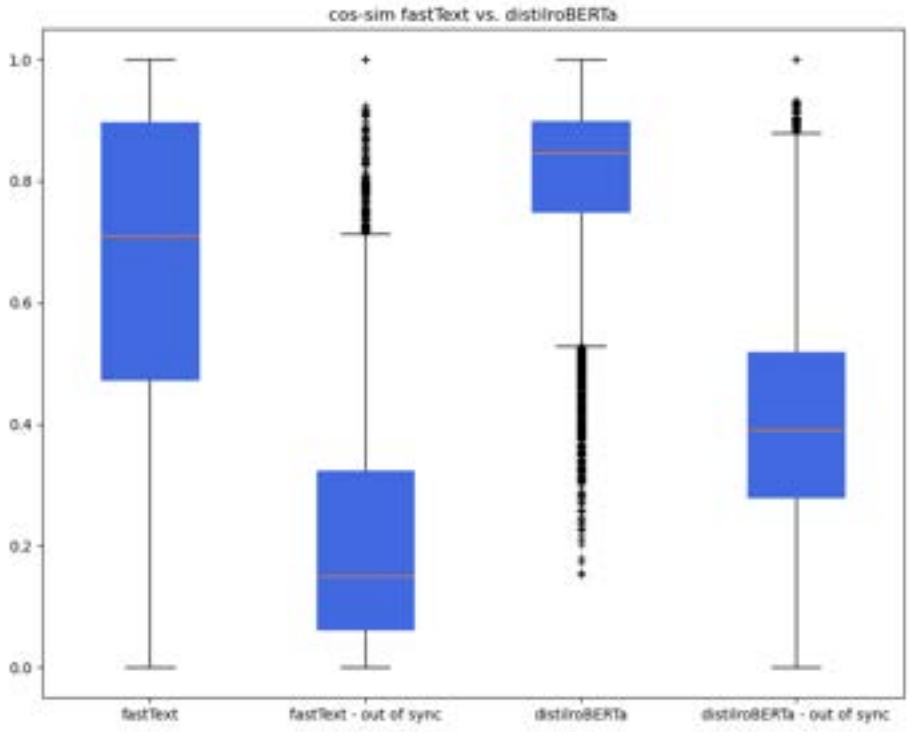


Abbildung 18.: Vergleich FastText und *DistilroBERTa* für WordSim Feature

ContextAgreement**

Beim Context Agreement wird das Umfeld des Wortes in der Utterance betrachtet. Dazu werden für beide Systeme jeweils fünf Worte vor und nach dem Zielwort betrachtet. Die Anzahl Übereinstimmungen an diesen Kontextwörtern werden durch die Anzahl betrachteter Worte geteilt. Falls die Utterance weniger als zehn Worte hat, werden alle Worte verwendet. Falls vor oder nach dem Wort nicht fünf Wörter vorhanden sind, werden auf der anderen Seite mehr Wörter verwendet. So ist immer garantiert, dass zehn Wörter betrachtet werden, sofern die Utterance länger als zehn Worte ist.

4.8.4. Resultat und Fazit

Die implementierte Feature Extraction Pipeline kann schnell und zuverlässig alle Features extrahieren und codieren. Neue Features können einfach definiert und ebenfalls hinzugefügt werden. Die Features können auch aus der Evaluationspipeline extrahiert werden. Für weitere Experimente steht nun ein Datenset mit über 250'000 Datensätzen zur Verfügung. Technisch gesehen wird das Datenset allerdings noch nicht exportiert, sondern nach der Extraktion jeweils sofort im Hauptspeicher verwendet.

4.9. Cross WER Evaluation

Eine wichtige Frage ist es, ob sich die Anwendung eines Ensemble-Algorithmus lohnt. Wie der Best-Possible Ansatz zeigt, sind auf vielen Korpora keine grossen Verbesserungen möglich. In diesen Fällen würde es unnötige Kosten verursachen, drei Systeme zu verwenden, um die WER nur minimal zu verbessern. In diesem Experiment wird untersucht, ob ein Zusammenhang zwischen der WER einzelner Systeme auf die Referenz und der gegenseitigen WER zweier Systeme besteht. Dieser Umstand wird in diesem Dokument als Cross WER bezeichnet. Die Cross WER wurde für alle Kombinationen von Systemen erstellt.

4.9.1. Resultat

Die Tabelle 7 zeigt die Cross WER zwischen den zwei Systemen, B7 und D2, im CEASR Korpus. In der Spalte value findet sich die Cross WER der beiden Systeme. Diese wurde für jeden Korpus in CEASR berechnet. Es kann beobachtet werden, dass bei tiefen WER Werten zwischen Systemhypothese und Referenz auch die Cross WER zwischen den zwei Systemen tief ist. Auf dem AMI Korpus (spontane Sprache) haben beide System auf die Referenz eine hohe WER, die Cross WER der beiden Systeme ist tiefer. Beide Systeme müssen daher an der gleichen Stelle gleich falsch transkribieren. Auf vorgelesener Sprache haben beide Systeme eine tiefe WER auf die Referenz. Die beiden Systeme haben auch eine tiefe Cross WER. Da die Systeme selbst schon eine einstellige WER haben, wird die Verbesserung durch ein Ensemble den Text nicht signifikant verbessern können. Ein Ensemble-Algorithmus könnte sich in diesem Fall nicht lohnen.

corpus	pair	sys1	sys2	value	sys1_ref_wer	sys2_ref_wer
ami_segmented_h.json	B7_D2	B7	D2	0,349507931	0,430510744	0,426202345
ami_segmented_mix.json	B7_D2	B7	D2	0,487123994	0,738798253	0,759956138
commonvoice.json	B7_D2	B7	D2	0,115312349	0,098960993	0,089888743
librispeech_clean.json	B7_D2	B7	D2	0,08540795	0,074920496	0,087840478
librispeech_other.json	B7_D2	B7	D2	0,161570381	0,154866903	0,163919241
st.json	B7_D2	B7	D2	0,038844396	0,032360461	0,036561381
voxforge.json	B7_D2	B7	D2	0,089383528	0,086803072	0,09305573

Tabelle 7.: Resultat Paarweiser Vergleich von System B7 mit D2

Falls zwei Systeme sehr ähnlich sind, wie B3 und B4, die beide vom gleichen Provider stammen. Die Systeme haben zur Referenz eine hohe WER. Die Cross WER dieser zwei Systeme ist aber im einstelligen Prozentbereich. In der Tabelle 8 kann dies aufgezeigt werden. Wobei das System B3 verglichen mit einem anderen schlechteren System, wie D2, keine tiefe Cross WER aufweist.

corpus	pair	sys1	sys2	value	sys1_ref_wer	sys2_ref_wer
ami_segmented_h.json	B3_B4	B3	B4	0,081114522	0,632958301	0,650613469
ami_segmented_h.json	B3_D2	B3	D2	0,595559252	0,632958301	0,426202345
ami_segmented_mix.json	B3_B4	B3	B4	0,092905658	0,826970286	0,834992788
ami_segmented_mix.json	B3_D2	B3	D2	0,886278964	0,826970286	0,759956138

Tabelle 8.: Resultat Paarweiser Vergleich von System B3 mit B4 und D2

Die Tabelle 9 zeigt, dass die Cross WER bei guten Transkripten, wie sie die Systeme auf libirspeech_clean erreichen, sich am schlechteren System orientiert.

pair	value	sys1_ref_wer	sys2_ref_wer
B7_B3	0,117384409	0,074920496	0,14312019
B7_B4	0,117384409	0,074920496	0,14312019
B7_B5	0,009605814	0,074920496	0,073460067
B7_B8	0,210796756	0,074920496	0,233795249
B7_C1	0,152717204	0,074920496	0,160682908
B7_C2	0,152717204	0,074920496	0,160682908
B7_D1	0,08540795	0,074920496	0,087840478
B7_D2	0,08540795	0,074920496	0,087840478
B7_kaldi_aspire	0,2046716	0,074920496	0,213271796
B7_kaldi_librispeech	1,015405017	0,074920496	1,012388501
B7_mozilla_deepspeech___am_generic_mm___lm_generic	0,118640818	0,074920496	0,095822726
B7_sphinx4___am_generic_c___lm_generic	0,257715305	0,074920496	0,264013949
B7_sphinx4___am_generic_ptm___lm_generic_p	0,327363682	0,074920496	0,335495126

Tabelle 9.: Resultat Paarweiser Vergleich von System B7 auf librispeech_clean

4.9.2. Fazit

Ist die Cross WER tief, so handelt es sich entweder um vorgelesene Sprache oder aber die verglichenen Hypothesen sind beide fast identisch. In beiden Fällen ist es zwecklos,

Ensemble-Methoden anzuwenden, da das Transkript entweder schon wenige Fehler aufweist oder aber ein Ensemble nicht divers genug wäre. Ist die Cross WER jedoch hoch, könnten sich Ensemble-Methoden lohnen, um die Vorhersage zu verbessern. In einer produktiven Umgebung könnte etwa eine Minute der Audioaufnahme mit allen drei Systemen transkribiert werden und dann die Cross WER berechnet werden. Nur bei einer hohen Cross WER würde dann mit Ensemble-Methoden fortgefahren werden.

5. Ensemble-Methoden

In diesem Kapitel werden die implementierten Ensemblemethoden erläutert. Es wird auf Implementationsdetails eingegangen und die Verbesserung der WER und SemDist aufgezeigt. Der Input für jede Methode ist wie schon erwähnt immer ein Alignment von den Systemen B7, D2 und C2 ist. Eine Übersicht der erreichten Verbesserungen aller Methoden ist in der Tabelle 13 (WER) bzw. 14 (SemDist) ersichtlich. Zuerst werden in den Kapiteln 5.2 - 5.5 die Methoden aus der originalen Lösung [2] besprochen, wo nötig erweitert sowie deren Daten mit den neuen Metriken erhoben. Anschliessend folgen in den Kapiteln 5.6 - 5.9 neue Ansätze.

5.1. Aufteilung in Training und Test

Um Ansätze, welche mit den Features aus Kapitel 4.8 trainiert wurden, gut messen zu können, werden vor der Feature Extraction 50 % der Utterances aus jedem Sub-Korpus als Testdaten zur Seite gelegt und die verbleibenden 50 % für das Training benutzt. Der Cutoff wurde bei 50 % angesetzt, weil die Lernkurve in Kapitel 5.6.6 nach der Hälfte der Daten keine nennenswerte Verbesserung mehr aufzeigte. Das Berechnen des Best-Possible auf diesen Testdaten zeigt ausserdem, dass die Testdaten das gesamte Datenset gut repräsentieren (Abbildung 19 und 20). Allerdings gibt es Abweichungen im Bereich von 2-4 %, falls man mit dem Best-Possible aus dem gesamten CEASR Korpus (Kapitel 4.5) vergleicht. Damit nun alle Resultate vergleichbar bleiben, wird für alle Methoden in diesem Kapitel das gleiche 50 % - Testdatenset verwendet, falls nichts anderes definiert wurde. Infolgedessen basieren die Resultate also auf der Hälfte von CEASR, was ungefähr 30 Stunden Audio entspricht.

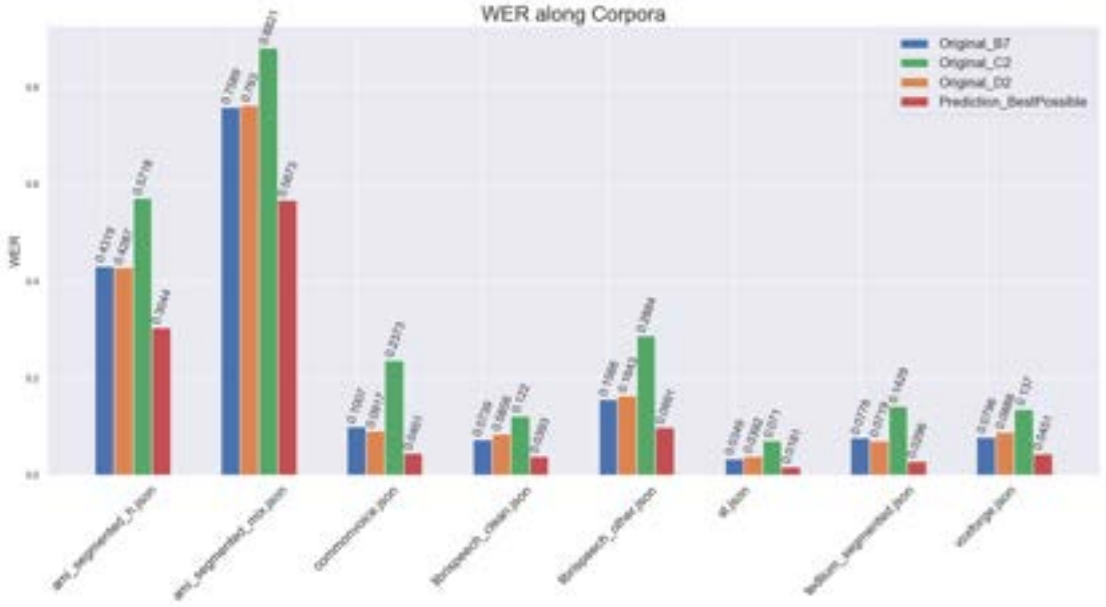


Abbildung 19.: Best-Possible WER (50 % Split)

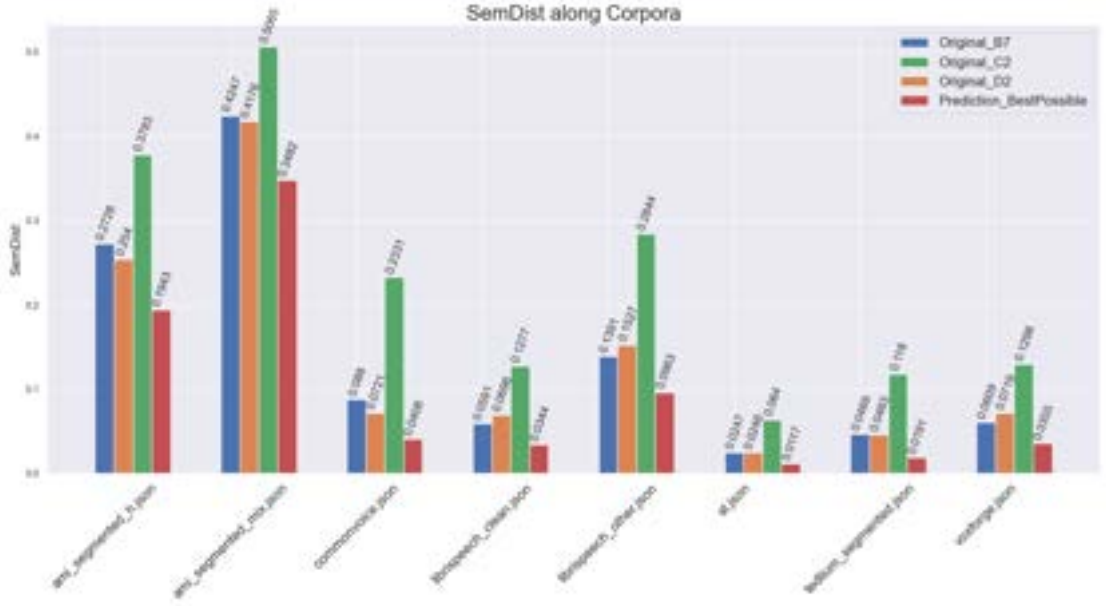


Abbildung 20.: Best-Possible SemDist (50 % Split)

5.2. Voting

Der Voting-Ensemble-Algorithmus entscheidet per Mehrheitsentscheidung, welcher Text aus den Hypothesen als Vorhersage zurückgegeben wird. Dies ist die naheliegendste Art eines Ensembles. Dieser Algorithmus war bereits im erhaltenen initialen Code vorhanden. Es wurde ein Refactoring betrieben und der Algorithmus so angepasst, dass n-Systeme berücksichtigt werden können.
Voting ist in zwei unterschiedlichen Ausprägungen implementiert. In der ersten Ausprägung wird das Alignment der Hypothesen stur sequenziell abgearbeitet. Die zweite Ausprägung aggregiert nacheinander folgende Teile des Alignments, welche nicht in allen Hypothesen übereinstimmen, zu einem zusammengefassten Teilsatz. Der Gedanke hinter dieser Aggregation ist es, lesbarere Resultate zu produzieren. Die Abbildungen 21 und 22 stellen das Alignment vor und nach der Aggregierung gegenüber. «has» und «annotation» wurde von den drei Systemen B7, D2 und C2 übereinstimmend transkribiert. Bei den drei Wörtern dazwischen stimmen nicht alle drei Systeme überein, deshalb werden diese zusammengefügt.

Abbildung 21.: Alignment vor Aggregierung im Alignment UI

Abbildung 22.: Aggregiertes Alignment im Alignment UI

5.2.1. Aggregated Votes Algorithmus

Im aggregierten Voting wird für jede Hypothese die Summe an Übereinstimmungen gebildet und daraus die beste Hypothese abgeleitet. In dieser Matrix das Beispiel aus der Abbildung 22.

$$\begin{array}{ccc} & its & own \\ a & its & own \\ the & it's & an \end{array}$$

Es wird spaltenweise über die Matrix iteriert. In der ersten Spalte haben alle Systeme einen anderen Wert ausgegeben, daher erhält jedes System einen Vote von $+1$, was zum Zwischenstand $[1, 1, 1]$ führt. In der zweiten Iterationen stimmen die ersten zwei Systeme überein, deshalb erhalten sie je einen Vote von $+2$, während System drei wieder $+1$ erhält. Das Gleiche gilt auch für die letzte Iteration, womit das Voting Resultat dieser Aggregation $[5, 5, 3]$ ist. Über den Index mit der höchsten Anzahl an Stimmen wird nun die Hypothese bestimmt, welche für diesen Abschnitt zurückgegeben wird. In Beispiel aus der Matrix ist das «its own».

5.2.2. Resultate

Wie Abbildungen 24 und 26 zeigen, kann Voting die WER teilweise um 0,5 bis 1 % reduzieren.

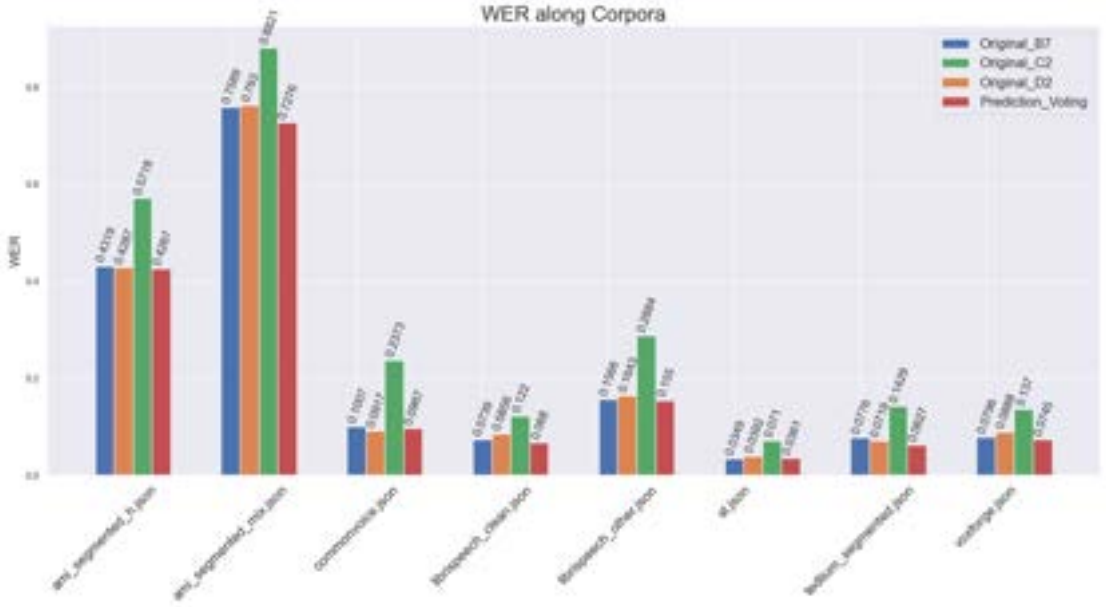


Abbildung 23.: Voting WER

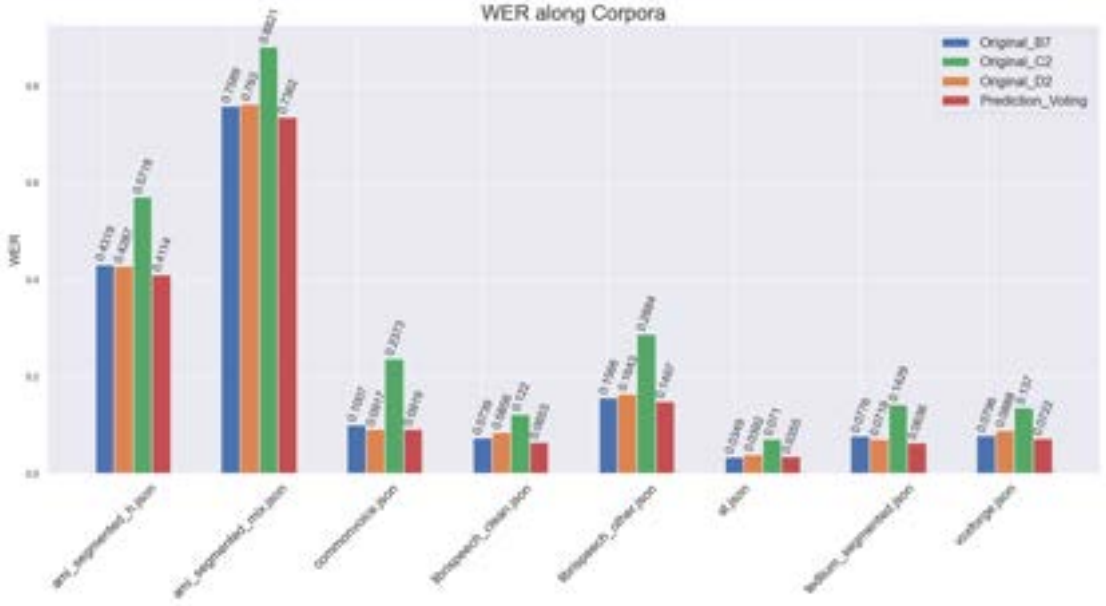


Abbildung 24.: Voting WER aggregiert

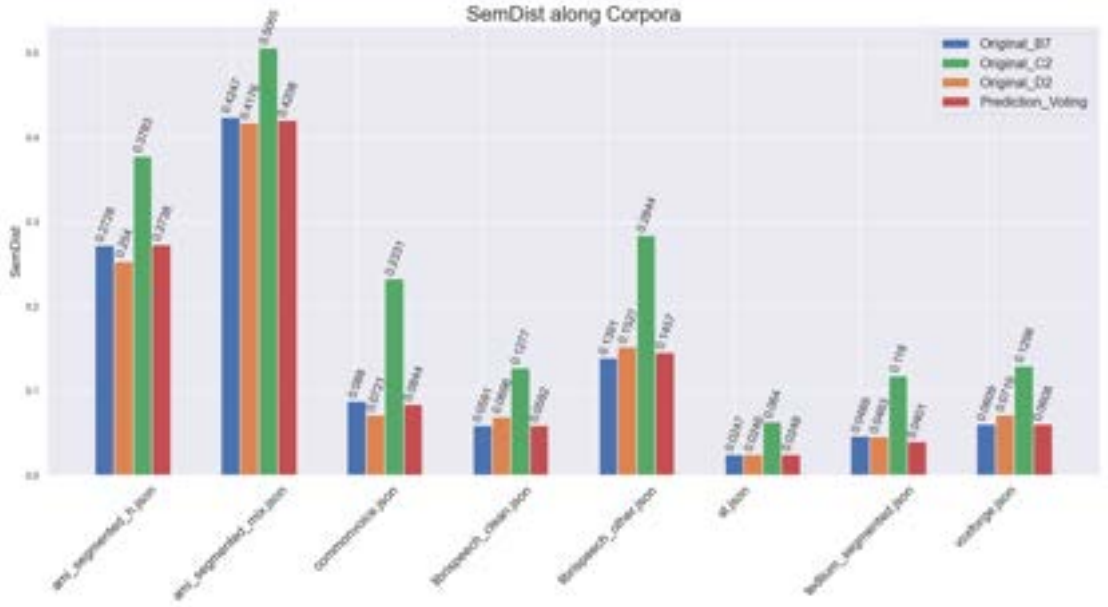


Abbildung 25.: Voting SemDist

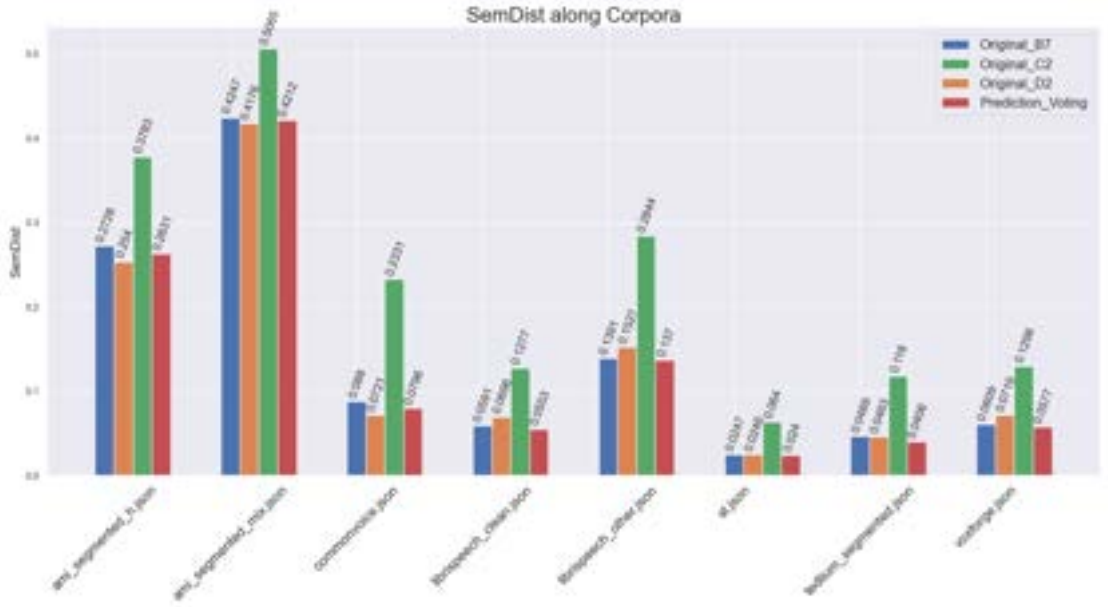


Abbildung 26.: Voting SemDist aggregiert

5.2.3. Fazit

Voting ist ein naheliegender Ansatz, welcher einfach zu implementieren ist. Durch Voting alleine lässt sich die WER oder SemDist nicht drastisch senken. Die aggregierte Version

des Votings liefert sowohl für die WER in Abbildung 24, sowie für SemDist in Abbildung 26 leicht bessere Resultate wie das nicht aggregierte Voting. Dieses wirkt plausibel. Ein System, das an den meisten Orten übereinstimmt, hat die höhere Wahrscheinlichkeit an uneinigen Stellen das richtige Wort transkribiert zu haben.

5.3. Word Similarity Voting

Anders als beim puren Voting Ansatz soll beim Word Similiartiy Voting auch bei nicht exakt gleichen Wörtern eine Aussage getroffen werden, welches System die am wahrscheinlichsten beste Vorhersage liefert. Hierzu werden die Ähnlichkeiten in den Wörtern der Hypothesen betrachtet. Zur Berechnung von der Ähnlichkeit in den aggregierten Hypothesen verwendet dieser Ensemble-Algorithmus pre-trained Word Vectors, welche auf Common Crawl und Wikipedia unter Verwendung von fastText trainiert wurden [47]. Der Algorithmus liefert die Vorhersage des Systems, dass die meisten Stimmen basierend auf der maximalen Ähnlichkeit per Wort erreicht hat. Dieser Algorithmus funktioniert nur auf einem aggregierten Alignment (siehe auch Kapitel 5.2 zur «Aggregation»).
Das folgende Beispiel veranschaulicht die Funktionsweise des Algorithmus:

the	*cap*	*eats*	*at*	*house*
the	*cat*	*eats*	*a*	*mouse*
their	*cats*	*is*	*a*	*mouse*

Für die drei Wörter «the», «the» und «their» werden die Word Vectors berechnet. Für die Word Vectors wird das Skalarprodukt berechnet. Die n-Systeme, die die höchste Cosinus-Ähnlichkeit haben, erhalten eine Stimme. Im Beispiel sind das die zwei gleichen Wörter «the».

0	1	0
1	0	0
0	0	0

In den weiteren Iterationen ergeben sich die folgenden summierten Stimmen:

0	1	0	0	2	0	0	2	0	0	2	0
1	0	$1 \rightarrow 2$	0	$1 \rightarrow 2$	0	$2 \rightarrow 2$	0	3			
0	1	0	0	1	0	0	2	0	0	3	0

Für jede Zeile in der Stimmen Matrix wird die Summe der Stimmen berechnet. Die Hypothese «the cat eats a mouse» hat fünf Stimmen erhalten und wird als Vorhersage zurückgegeben.

5.3.1. Resultate

Die Abbildungen 27 und 28 zeigen die WER und SemDist des Word Similarity Voting. Bis auf den Korpus ami_segmented_mix sind die Resultate im selben Bereich wie beim aggregierten Voting.

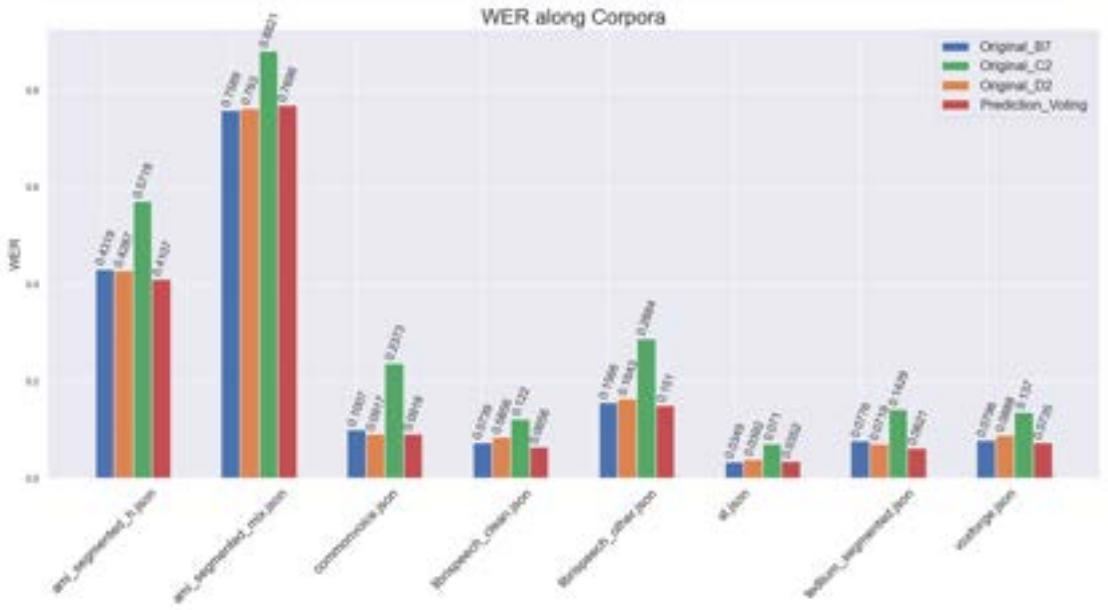


Abbildung 27.: Word Similarity WER

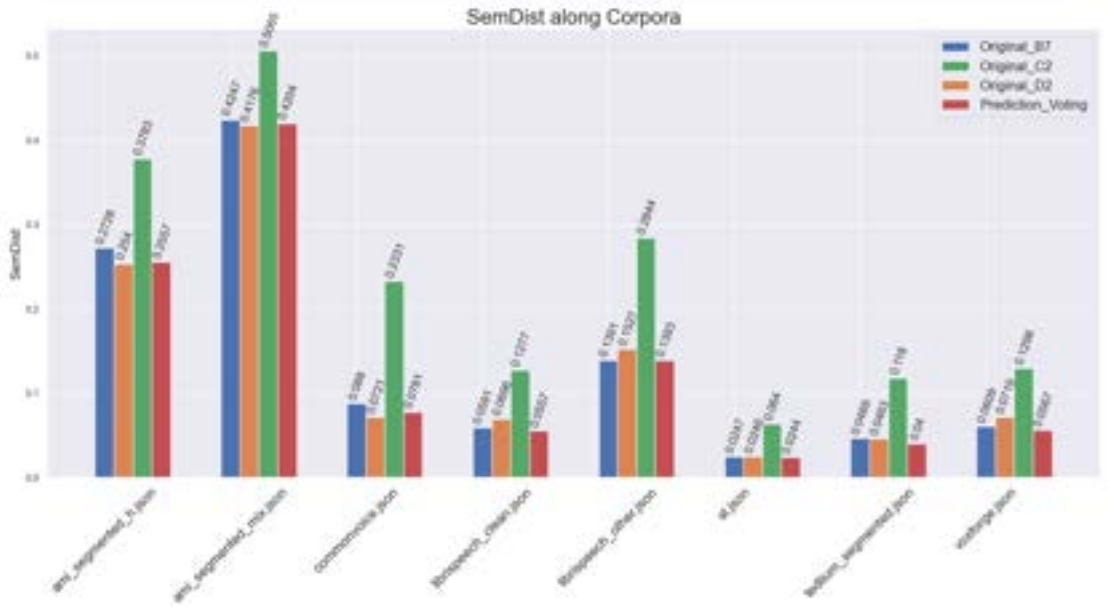


Abbildung 28.: Word Similarity SemDist

5.3.2. Fazit

Das Verwenden von Word-Embeddings für ein ähnlichkeitsbasiertes Voting hat keinen durchgehenden, positiven Einfluss auf die WER im Vergleich zum normalen Voting.

Semantisch ist der ähnlichkeitsbasierte Ansatz hingegen leicht besser (<1 % SemDist) als das normale Voting. Das beobachtete Verhalten ist plausibel. Die WER misst nur exakt korrekte Zeichenfolgen. Durch das Verwenden der Ähnlichkeit werden tendenziell auch unterschiedliche Zeichenfolgen in der Vorhersage begünstigt.

5.4. Perplexities (GPT)

Dieses Kapitel untersucht die Wirksamkeit eines Einsatzes von GPT Modellen in einem Ensemble. GPT-2 generiert in Versuchen synthetische Texte in bis dato unerreichter Qualität. In diesem Kapitel wird untersucht, ob GPT-2 auch in einem ASR Ensemble angewendet werden kann. Eine erste Version des Algorithmus lag zu Beginn der Arbeit bereits vor [2]. In dieser Bachelorarbeit wurde der Algorithmus weiterentwickelt, um mit einer beliebigen Anzahl an Hypothesen umgehen zu können. Des Weiteren wurde mit GPT-2 ein neueres Modell eingesetzt [48]. Die Methode versucht, Wörter anhand der sprachlich höchsten Wahrscheinlichkeit auszuwählen.

5.4.1. Transformer-Architektur und GPT

Ein Transformer ist eine Architektur von neuronalen Netzen. Bis zum Aufkommen von Transformer-basierten Algorithmen sind im Bereich NLP meistens auf einem basierende Algorithmen verwendet worden. Diese Modelle behandeln Wörter in Sätzen als einzelne, lose Wörter. In der Sprache kommt es aber auf den Kontext eines Wortes an. Wenn Wörter in einem Satz die Position tauschen, kann der Satz eine komplett neue Bedeutung erhalten. Transformers, welche von Google im Paper “Attention is All You Need” [49] das erste Mal erwähnt wurden, basieren nicht wie frühere Modelle auf Rekursion, sondern auf einem Aufmerksamkeitsmechanismus. Ein weiterer Vorteil ist die Parallelisierung beim Trainieren der Modelle [49], wodurch Modelle auf riesigen Datenmengen trainiert werden können. GPT sowie auch BERT folgen der Transformer-Architektur.

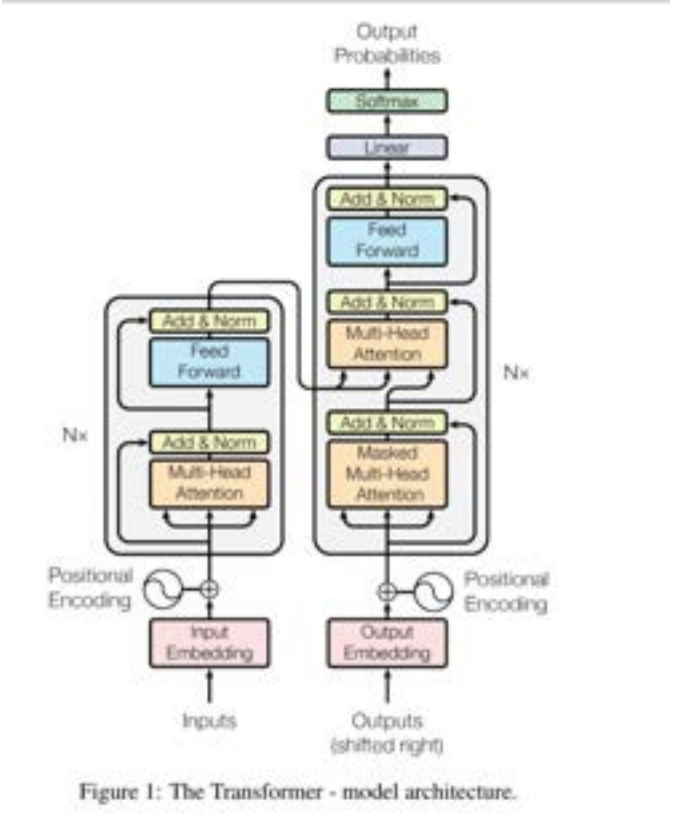


Figure 1: The Transformer - model architecture.

Abbildung 29.: Transformer - Modell Architektur [49]

Das Transformer Modell nutzt eine encode-decoder Struktur mit mehreren Self-Attention und punktweise voll verknüpften Layern, sowohl im Encoder wie auch im Decoder [49]. Vereinfacht ausgedrückt lernt ein Transformer dadurch, durch Positional Encoding und Self-Attention, die Bedeutung der Struktur von Sprache sowie deren Zusammenhänge kennen. Das von Google 2017 vorgestellte Modell erreichte auf zwei WMT Übersetzungstasks einen neuen Stand der Technik [49].

5.4.2. Perplexity

Perplexity (PPL) ist eine gebräuchliche Metrik zur Evaluierung von autoregressiven oder kausalen Sprachmodellen wie GPT eines ist. Perplexität ist definiert als die potenzierte durchschnittliche negative logarithmische Wahrscheinlichkeit einer Sequenz. Für eine tokenisierte Folge von Wörtern ist die Perplexity wie folgt definiert:

$$PPL(x_0, x_1, ..., x_n) = exp\{-\frac{1}{t}\sum_{i}^{n} \log p_0(x_i|x_{<i})\} \tag{5.1}$$

wobei $\log p_0(x_i|x_{<i})$ die logarithmische Wahrscheinlichkeit des i-ten Tokens in Abhängigkeit zu den vorangegangenen Token ist.

5.4.3. Implementierung

Der Algorithmus verwendet das OpenAI GPT-2 Modell, welches auf 40 GB Text trainiert wurde. GPT-2 ist ein Transformer mit 1.5 Milliarden Parameter [48].
Falls alle n-Systeme das gleiche Wort vorhergesagt haben, wird dieses zurückgegeben. Ist das nicht der Fall, wird die Perplexity für jedes der n-Systeme berechnet. Dazu werden die einzelnen Wörter mit dem Kontext erweitert. Es werden bis zu 3 Wörter davor und danach zu einem Satz zusammengebaut. Dieser Satz wird tokenisiert und ins Modell gegeben. Die Forward-Funktion des Modells liefert die Cross Entropie $-\frac{1}{t}\log P(X)$ zurück, wobei $P(X) = \prod_{i=0}^{t} p(x_i|x_{<i})$ das Produkt der Wahrscheinlichkeit jedes Elements ist. Wird die Cross Entropie potenziert, erhält man die Perplexity. Das Wort mit der tiefsten Perplexity wird als wahrscheinlichste Vorhersage zurückgegeben.

5.4.4. Resultate

Die Resultate von einem puren Perplexity Ensemble-Algorithmus sind auf allen Korpora schlechter als von einem puren Majority Voting Ansatz.

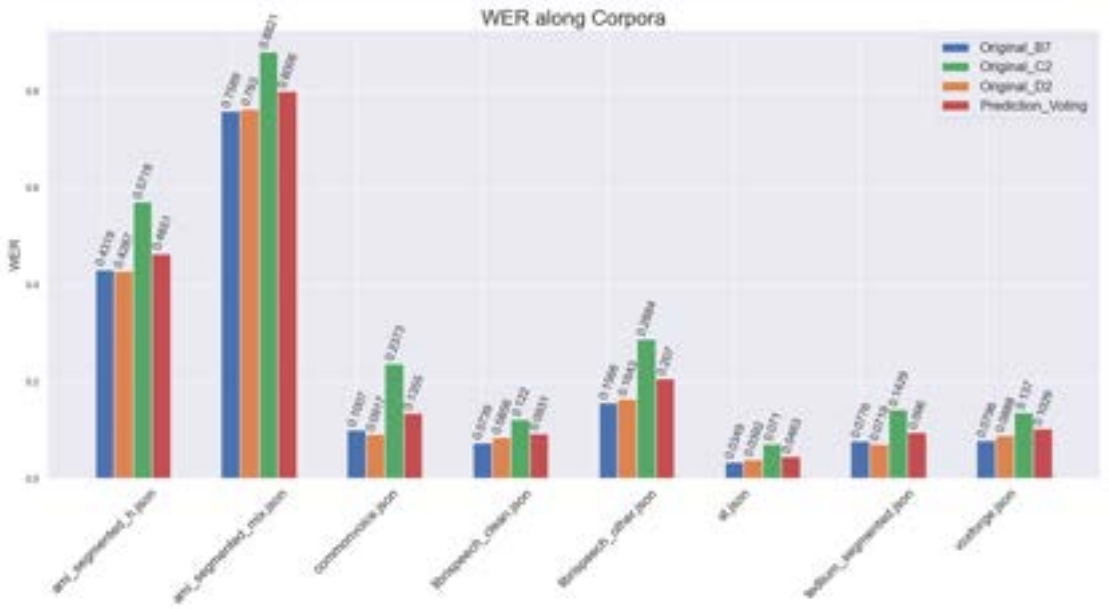


Abbildung 30.: Perplexity WER

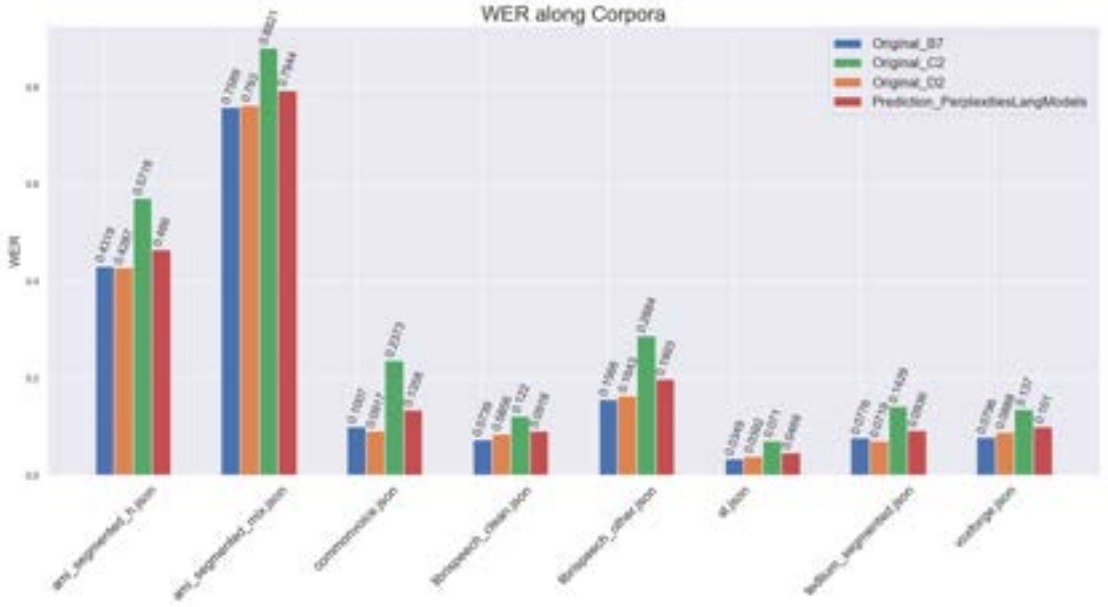


Abbildung 31.: Perplexity WER aggregiert

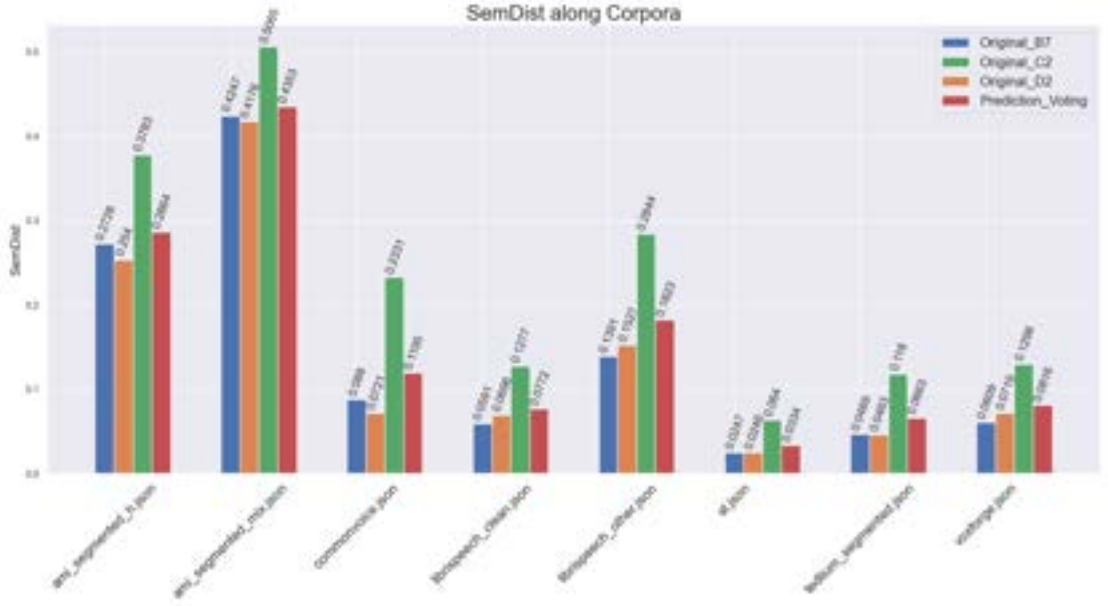


Abbildung 32.: Perplexity SemDist

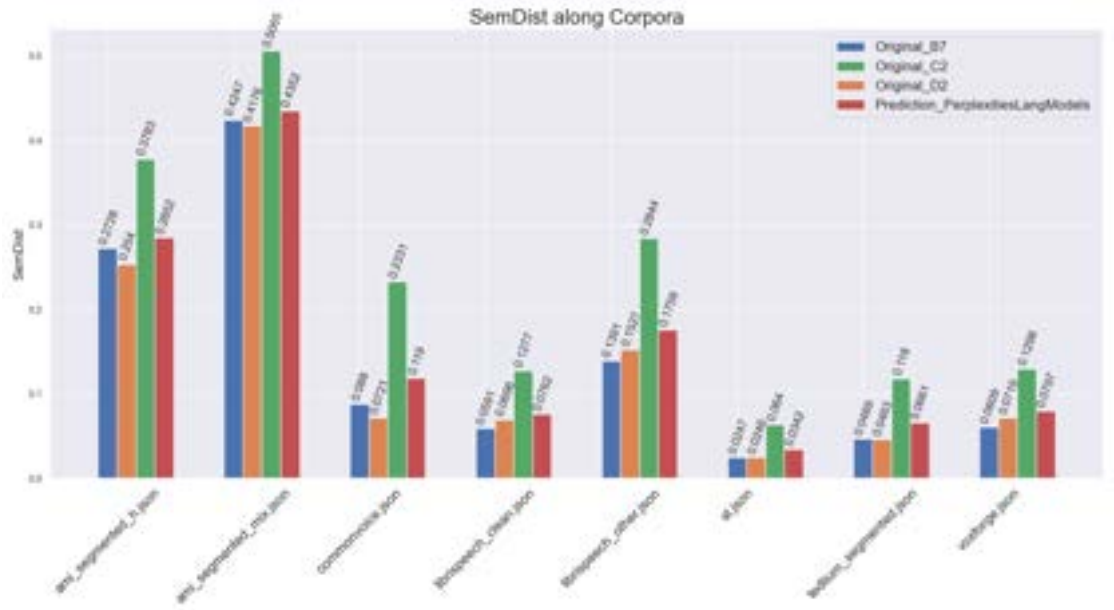


Abbildung 33.: Perplexity SemDist aggregiert

5.4.5. Fazit

Offensichtlich reicht die Wahrscheinlichkeit einer Wortsequenz nicht aus, um auf Wort- oder Teilsatzbasis die passendste Hypothese auszuwählen. Obwohl die WER- und SemDist-Werte mit dieser Methode schlechter werden, kann man nicht sagen, dass die Methode gar nicht funktioniert. Vielmehr beweist diese Methode, dass nicht immer das Wahrscheinlichste gesagt wird. Deshalb würde sich diese Methode eher als Tiebreaker anbieten. Dies wird in Kapitel 5.5 weiter untersucht.

5.5. Hybrid - Perplexities (GPT) & Voting

Dieser Ensemble-Algorithmus ist ein Zusammenschluss von Voting 5.2 und Perplexities 5.4. Sowohl bei der aggregierten Version wie auch in der Word-by-Word basierten Implementierung wird initial der Voting-Algorithmus aufgerufen. Falls beim Voting zwei Wörter oder n-Gramms gleich viele Stimmen bekommen, wird die Perplexities Implementierung aufgerufen und die Vorhersage dadurch bestimmt.

5.5.1. Resultate

Die WER ist sowohl im puren Hybrid Ansatz in Abbildung 34 als auch im aggregierten Ansatz in Abbildung 35 im einstelligen Prozentbereich besser als das beste Originalsystem. Die SemDist in Abbildung 36 (normal) und Abbildung 37 (aggregiert) ist auf den Korpora librispeech, tedlium und voxforge besser als die Originalsysteme.

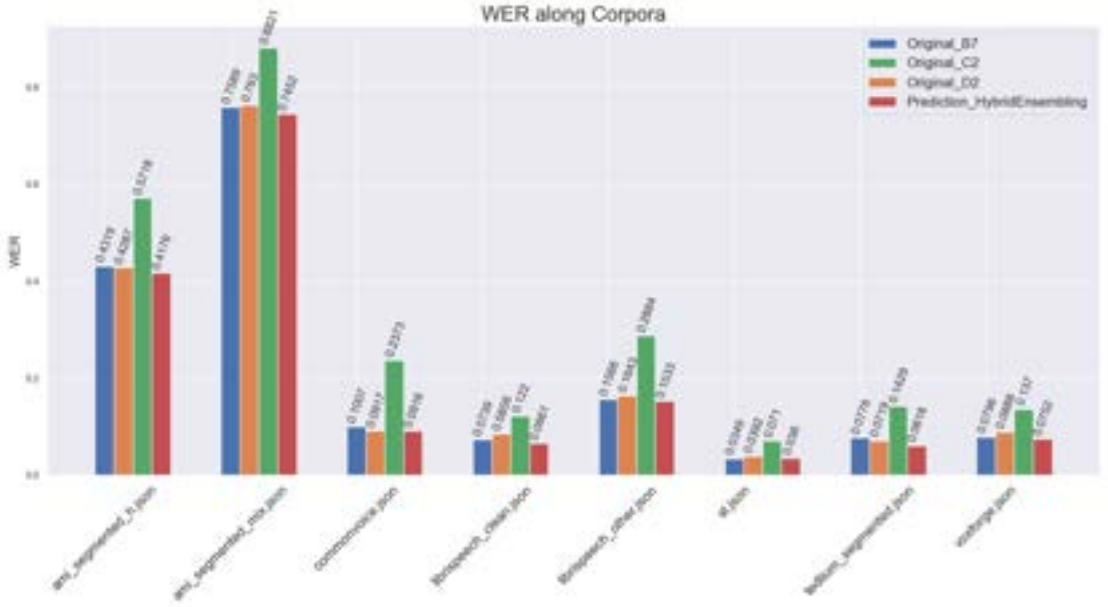


Abbildung 34.: Hybrid WER

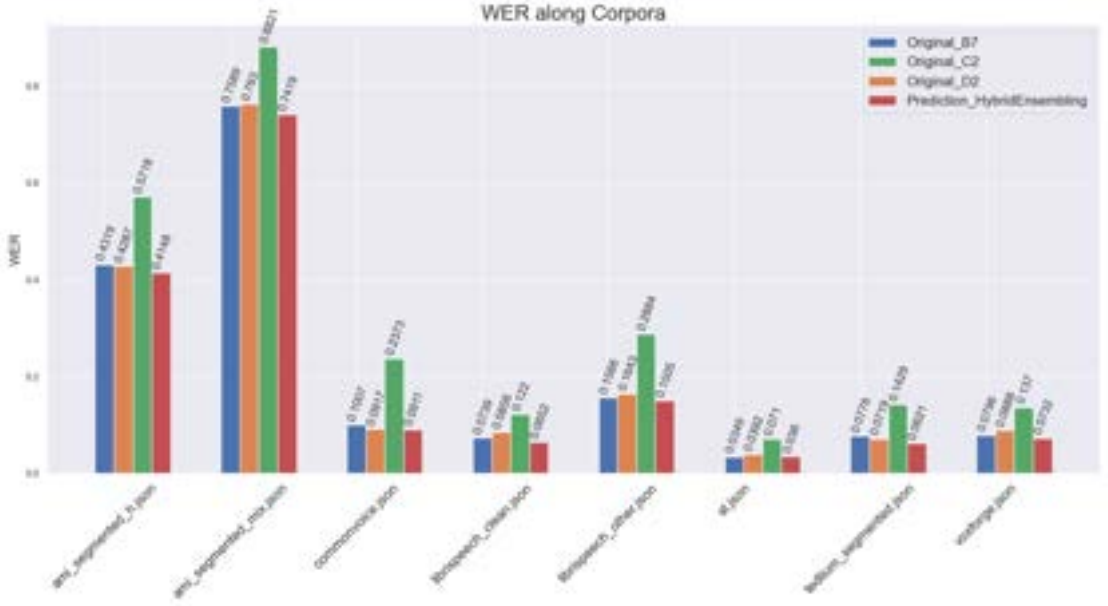


Abbildung 35.: Hybrid WER aggregiert

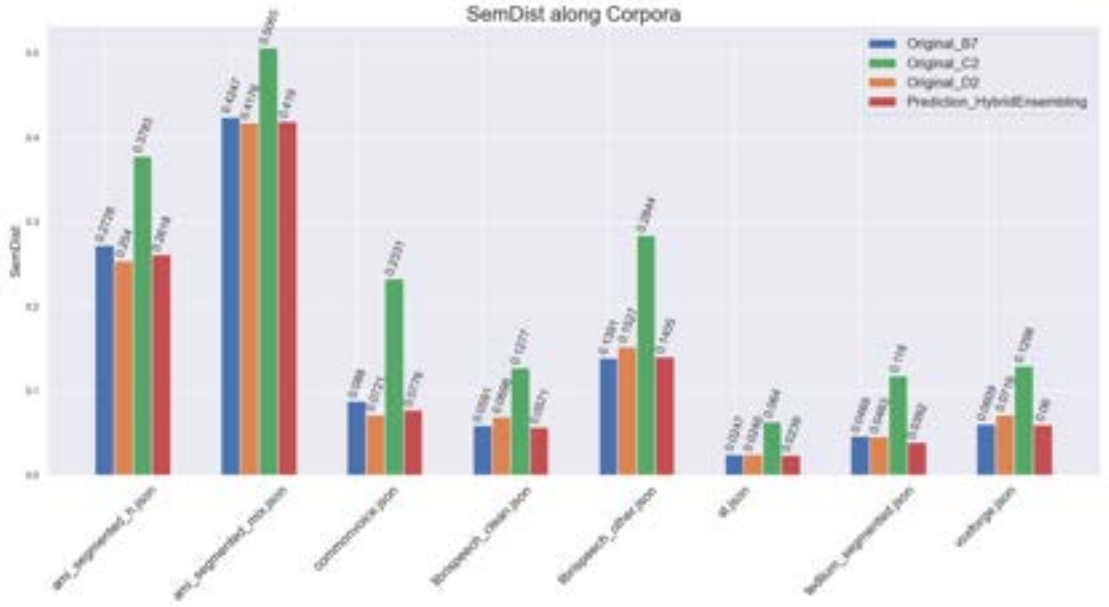


Abbildung 36.: Hybrid SemDist

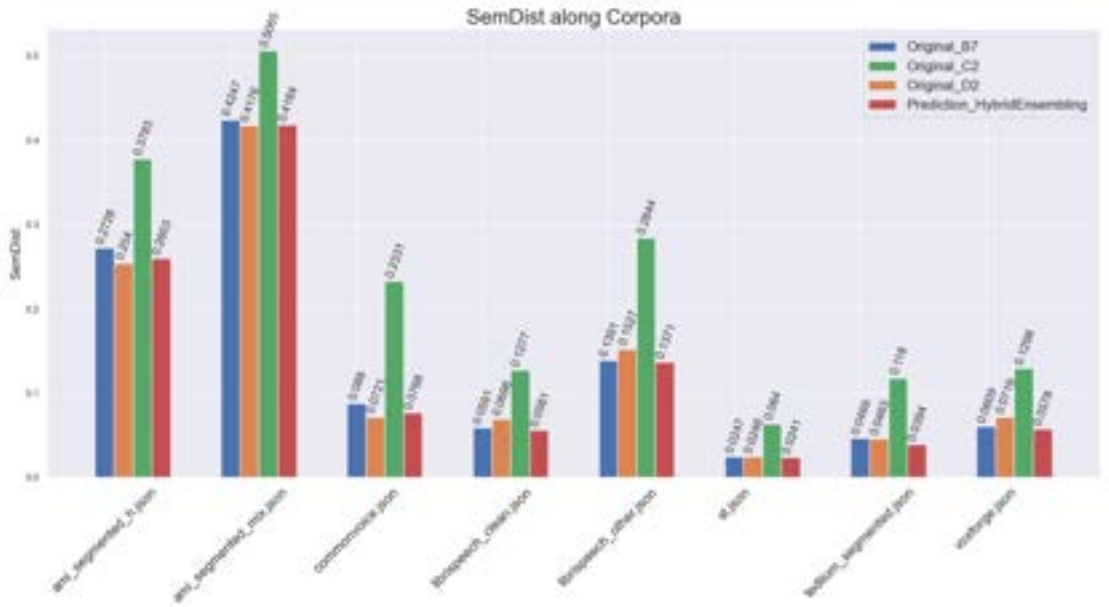


Abbildung 37.: Hybrid SemDist aggregiert

5.5.2. Fazit

Die Kombination von Voting 5.2 und Perplexities 5.4 liefert auf den Testdaten teilweise bessere Resultate als die beiden eigenständigen Implementierungen. Die Verbesserungen

oder Verschlechterungen im Vergleich zum Voting sind aber im einstelligen Prozentbereich. Voting hat in diesem Verfahren ein zu grosses Gewicht. Dadurch dass die Berechnung von der Perplexity oft nicht ausgeführt werden muss, ist der Algorithmus aber schnell in seiner Laufzeit auf den Testdaten.

5.6. Best-Possible Estimator

Mit dem Datenset aus Kapitel 4.8 wurde ein Classifier trainiert, welcher versucht, die richtigen Systeme (das Best-Possible) vorherzusagen. Als erster Schritt wurde ein geeigneter Algorithmus ausgewählt. Anschliessend wurde das Modell einem Finetuning unterzogen und zuletzt in der originalen Pipeline eingebaut, sodass das Modell auch mit WER oder SemDist gemessen werden kann.

5.6.1. Aufbereitung des Datensets und Balancing

Wie in Kapitel 5.1 erwähnt, wurde für das Training das Datenset aufgeteilt in ein 50:50 Train-Test Split. Allerdings ist das Datenset unbalanciert. Die Klasse mit allen Systemen, B7_D2_C2, ist stark überrepräsentiert, was potenziell zu Problemen führen könnte. Glücklicherweise ist dieses Label nicht notwendig. Das Label sagt aus, dass alle Systeme gleich sind. Dies lässt sich aber auch einfach aus dem Alignment ermitteln und muss nicht gelernt werden. Daher können alle Datensätze mit diesem Label ignoriert werden. Durch diese Bereinigung verbleiben noch ca. 70'000 Datensätze für das Training.

5.6.2. Auswahl des Trainingsalgorithmus

Für Klassifikationsprobleme gibt es viele Algorithmen. Das no free lunch theorem besagt, dass es keinen allgemein besten Algorithmus gibt. Um ein Gespür für die Algorithmen zu gewinnen, wurden diverse bekannte Algorithmen evaluiert. Mit jedem Algorithmus wurde ein Modell trainiert. Anschliessend wurde die Accuracy und der F1-Score auf den Testdaten berechnet und eine Confusion-Matrix erstellt. Es wurden folgende Algorithmen betrachtet:

- Logistic Regression
- Decision Tree
- Random Forest (500 Decision Trees)
- Support Vector Machine
- Gradient Boosting
- eXtreme Gradient Boosting
- Bagging mit 10 Decision Trees
- Bagging mit 10 Random Forests (500 Decision Trees)

- Stack mit eXtreme Gradient Boosting, Random Forest & Logistic Regression
- Voting mit eXtreme Gradient Boosting, Random Forest & Logistic Regression

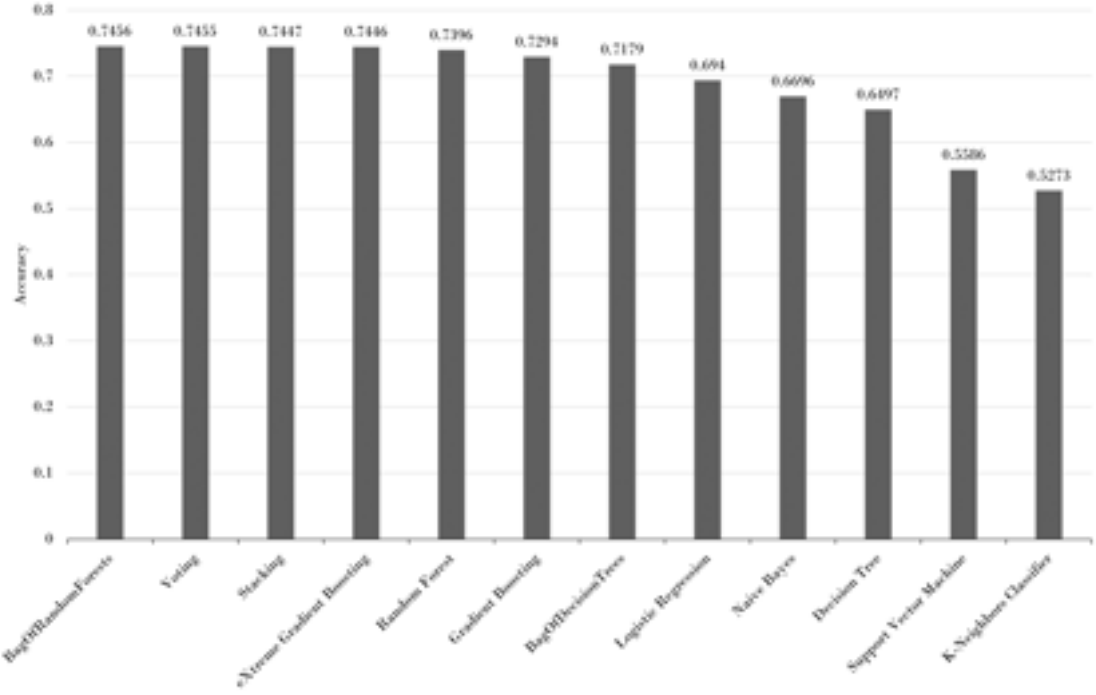


Abbildung 38.: Accuracy verschiedener Modelle für das Datenset aus Kapitel 4.8

Der Vergleich der Scores in Darstellung 38 sowie der F1-Scores in Anhang A.6 zeigt, dass es am oberen Ende kaum Unterschiede zwischen den verwendeten Algorithmen gibt. Weil die Funktionsweise von eXtreme Gradient Boosting bereits bekannt war, wurde mit diesem Algorithmus fortgefahren.

5.6.3. eXtreme Gradient Boosting (XGBoost)

eXtreme Gradient Boosting (XGBoost) ist eine effiziente, leicht modifizierte Implementation von Gradient Boosting. Obwohl Gradient Boosting ein beliebiges Modell als Base Learner zulässt, beschränkt sich XGBoost hier auf Decision Trees. Durch das Ausnutzen der sparsity durch One-Hot Encoding, kann mit XGBoost bis zu zehn mal schneller als mit herkömmlichen Gradient Boosting Machines trainiert werden. Die Machinelearning-Herausforderungen der Plattform Kaggle demonstrieren die Effektivität von XGBoost. Im Jahr 2015 verwendeten 17 der 25 Gewinnerteams XGBoost [50].

5.6.4. Parameter

XGBoost und Gradient Boosting bieten einige Parameter, welche für ein Finetuning verwendet werden können. Die gängigsten Parameter lauten:

- *objective*: Die zu optimierende Zielfunktion
- *n_estimators*: Anzahl Decision Trees im Ensemble (bzw. Anzahl Iterationen)
- *max_depth*: Maximale Tiefe eines einzelnen Decision Trees
- *subsample*: Anteil der Trainingsdaten, welche für eine Iteration verwendet werden
- *colsample_bytree*: Anteil der Features, welche in einer Iteration beachtet werden
- *learning_rate*: Learning-Rate

5.6.5. Parameter Finetuning

Da Multilabel-Klassifikation gemacht wird, ist das Objective *multi:softmax*. Die wahrscheinlichste Klassenzugehörigkeit wird mittels Softmax bestimmt. Um *n_estimators* zu bestimmen, wurde eine Iterationskurve geplottet. Auf dieser war ersichtlich, dass es nach 500 Iterationen keine Verbesserungen mehr gibt. Anschliessend wurde *max_depth* bestimmt, indem der Wert so lange um eins erhöht wurde, bis es keine Verbesserung mehr gab. Als letztes wurden *learning_rate*, *subsample* und *colsample_bytree* angepasst, um Overfittung zu vermeiden. Dazu wurde die initiale Learning-Rate schrittweise von 0,2 bis 0,03 reduziert und gleichzeitig die Anzahl Iterationen erhöht, so dass der Score gleich bleibt. *Subsample* und *colsample_bytree* wurden so lange reduziert, bis sich der Score verschlechterte. Dies war bei $colsample\_bytree = 0,8$ und $subsample = 0,9$ der Fall.

5.6.6. Lernkurve

Um die Qualität des Modells weiter beurteilen zu können, wurde auf einer 80:20 Aufteilung der Daten eine Lernkurve erstellt. Abbildung 39 zeigt eine flache Lernkurve. Die Trainingskurve sinkt monton, was gegen ein Overfitting spricht. Der Abstand der Kurven deutet jedoch auf Underfitting hin. Die Anzahl Trainingsbeispiele haben dabei nur einen geringen Einfluss auf die Accuracy. Daher wurde, wie in Kapitel 5.1 erwähnt wurde, mit einem 50:50 Split fortgefahren, damit die Testdaten für andere Methoden möglichst aussagekräftig zu bleiben.

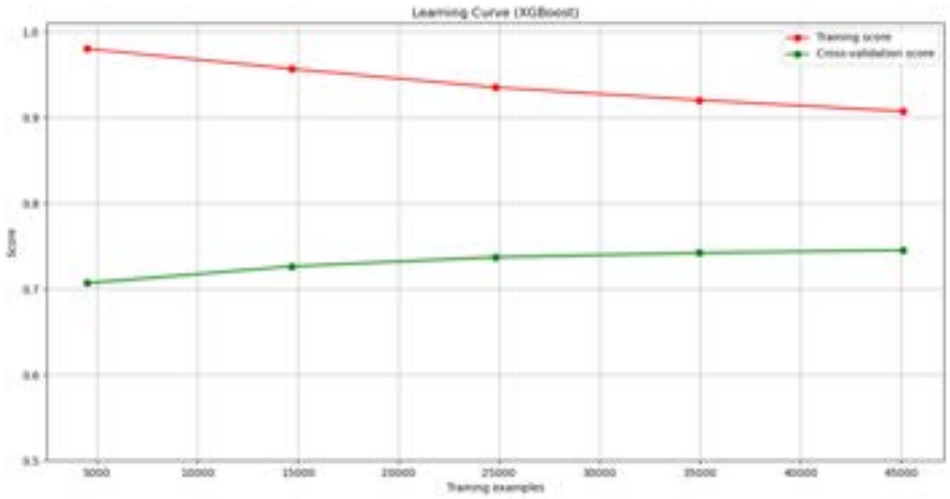


Abbildung 39.: Lernkurve vom XGBoost-Modell (Finetuned)

5.6.7. Finales Modell

Das finale Modell erreicht eine Accuracy von 75,2 % auf den 50 % Testdaten. Mit K-Fold Cross-Validation wurde sichergestellt, dass das Modell stabil ist. Ein Blick auf die Confusion-Matrix in Abbildung 40 zeigt, dass das Modell Zusammenhänge zwischen den Systemen gut gelernt hat. Das Modell ist also in der Lage, aus den Features aus Kapitel 4.8.2 mehr oder weniger gut die Kombinationen von richtigen Systemen vorherzusagen. Allerdings ist die Vorhersage für einzelne Systeme ungenau.

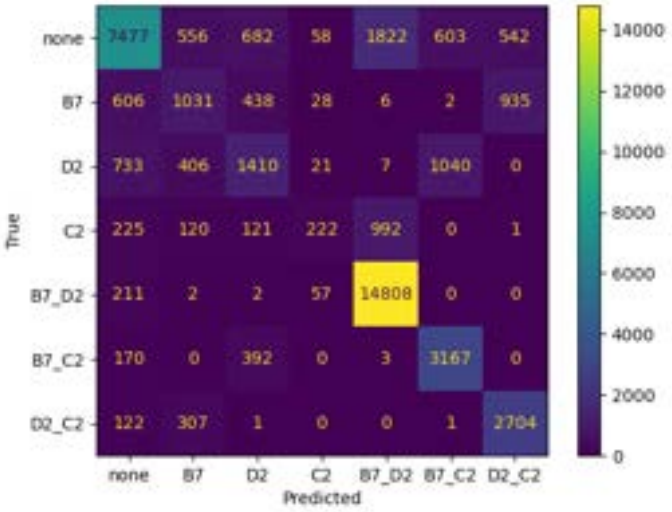


Abbildung 40.: Confusion Matrix XGBoost-Modell

5.6.8. Feature Importance

Ein nützliches Feature von XGBoost ist die Ausgabe der Feature Importance. XGBoost gibt aus, wie stark die einzelnen Features ins Gewicht fallen. Damit kann die Qualität der Features beurteilt werden.

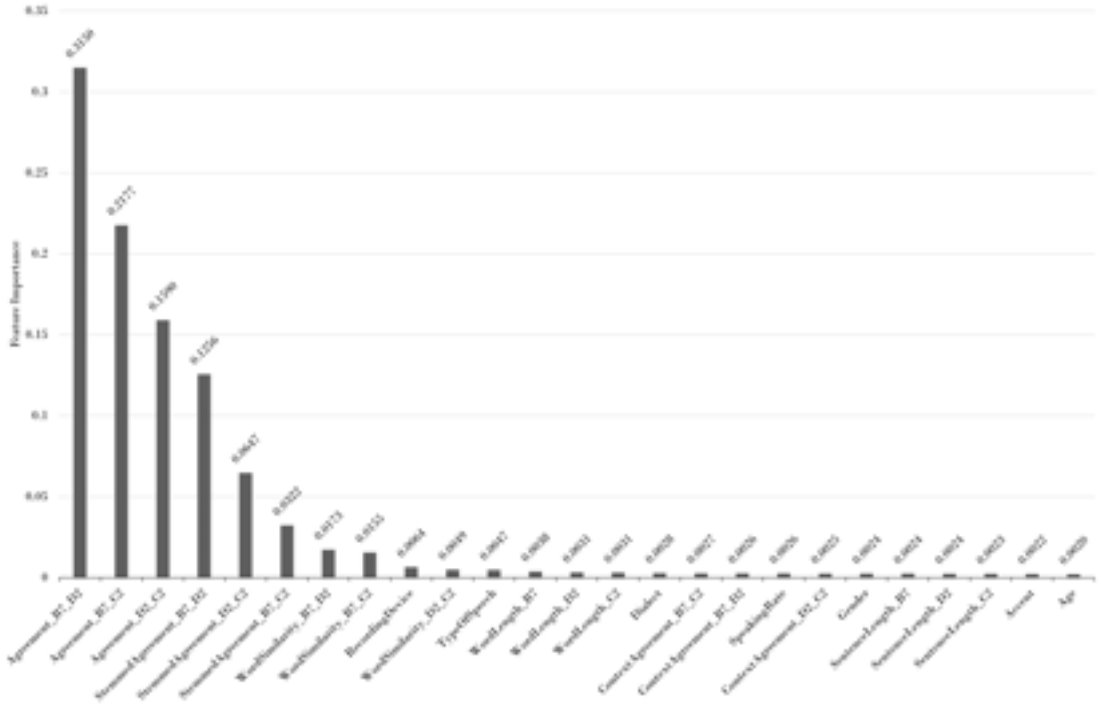


Abbildung 41.: Gelernte Feature Importance vom XGBoost-Model

Abbildung 41 zeigt, dass für das trainierte Modell die Agreement-Features klar bevorzugt werden. Die CEASR-Features fallen kaum ins Gewicht und tragen selten mehr als 1 % zum Resultat bei. Offensichtlich sind die einfachen, auf «string equality» basierten Agreements die besten Features. Das ist plausibel, weil das Label und Best-Possible auch auf «string equality» basieren. Es stellt sich jedoch die Frage, ob der «string equality» Ansatz zu starr ist.

5.6.9. Implementation in der Evaluationspipeline

Bis jetzt wurde das Modell anhand von F1-Score oder Accuracy gemessen. Eine Bewertung der WER oder SemDist wäre gewünscht, um andere Ensemble-Methoden zu vergleichen. Dazu wurde das trainierte Modell in die Evaluationspipeline integriert. Die Funktionsweise dieser Implementation gestaltet sich dabei sehr einfach. Für jeden Eintrag im Alignment werden die Features aus Kapitel 4.8.2 gebildet. Danach wird mit dem trainierten Modell vorausgesagt, welche Systeme korrekt liegen. Das Wort des ersten dieser Systeme wird als Resultat in die neue Hypothese übernommen.

5.6.10. Resultate

Mit dem Best-Possible Estimator kann die durchschnittliche WER auf spontaner Sprache moderat verbessert werden. Abbildung 42 zeigt eine Verbesserung der WER von fast 6 % auf den AMI-Korpora. Auf normaler Sprache liegt die Verbesserung im Rahmen von 1-2 %. Wie Abbildung 43 zeigt, gibt es semantisch in vielen Fällen keine oder nur eine geringe Verbesserung. Bemerkenswert ist jedoch, dass fast überall die SemDist des besten Systems (D2) erreicht wird, obwohl B7 die Primärhypothese ist. So ist die SemDist zumindest im Vergleich zu B7 auch moderat gesunken.

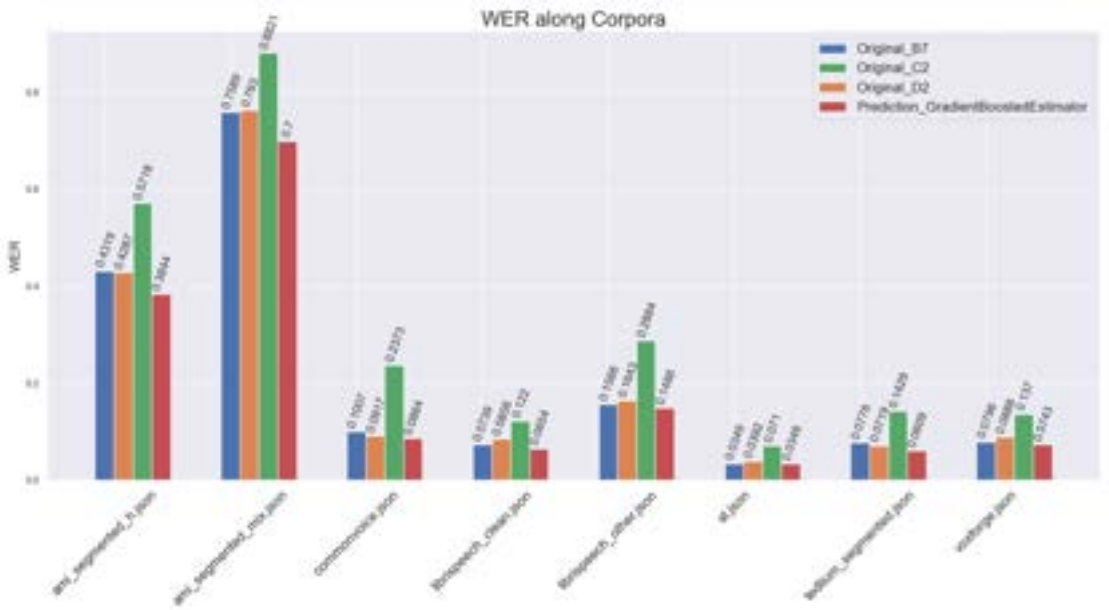


Abbildung 42.: GradientBoostedEstimator WER

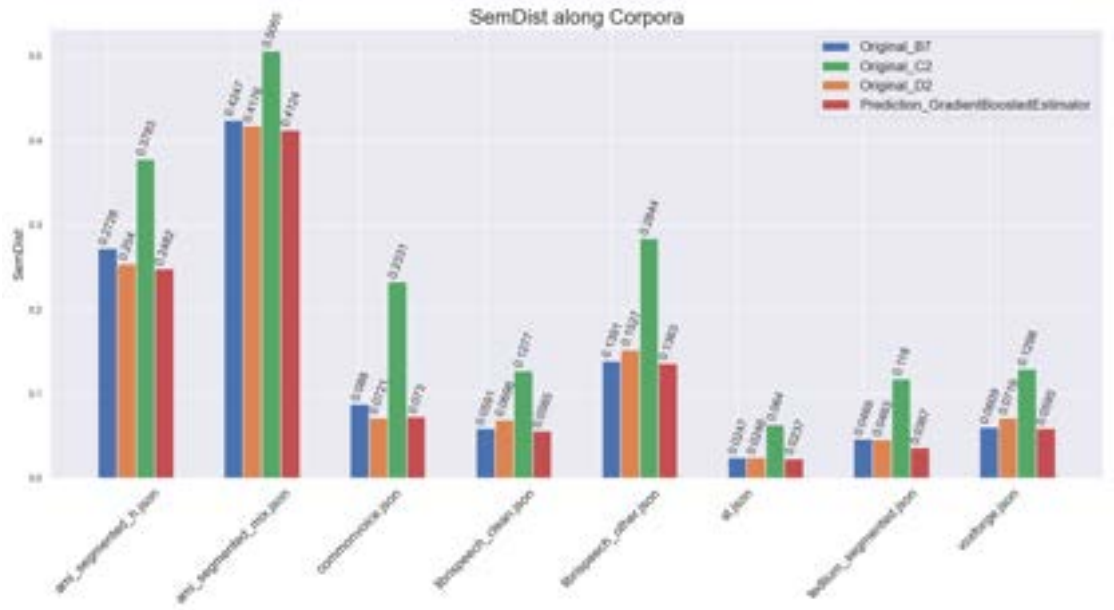


Abbildung 43.: GradientBoostedEstimator SemDist

5.6.11. Fazit

Es zeigt sich, dass gelernt werden kann, in welchen Situationen welches System ausgewählt werden soll. Verschiedene Indikatoren wie Confusion Matrix, F1-Score oder Feature Importance zeigen, dass es noch viel Potenzial in einem solchen Ansatz gibt. Der Klassifikator bringt allerdings auch noch andere Möglichkeiten mit sich. Die Confusion-Matrix zeigt, dass übereinstimmende Systeme gut klassifiziert werden, einzelne Systeme aber nicht. Ausserdem wird 'none' auch gut klassifiziert. Dies könnte auf verschiedene Weisen ausgenutzt werden. Zum einen könnte man den Ansatz mit anderen Ensemble-Methoden kombinieren. Falls das Modell zwei Systeme als richtig klassifiziert, wird dieser Aussage vertraut. Anderenfalls wird eine andere Methode (Voting, Perplexity, BERT etc.) verwendet. Zum anderen bringt die 'none' Prediction die Chance mit, falsche Stellen im Text zu markieren. So könnte zum Beispiel in Interscriber markiert werden, falls ein Wort falsch sein könnte. Alternativ könnten die falschen Wörter auch einfach entfernt werden. Damit das möglich wäre, müsste festgestellt werden, ob ein Text lesbarer ist, wenn falsche Worte oder gar keine Worte da stehen. Dieser Ansatz wurde allerdings nicht mehr weiter verfolgt.

5.7. Best-Possible Estimator - Semantisches Label

Im Kapitel über die Feature Importance (5.6.8) stellte sich die Frage, ob der «string equality» Ansatz des Labels zu starr für eine Verbesserung der SemDist ist. Grundsätzlich kann man sagen, dass mit «string equality» die WER optimiert wird. Mit semantischen Ansätzen wird tendenziell eher die SemDist besser, auch wenn die WER möglicherweise

gleich bleibt. Daher wurde untersucht, ob sich die SemDist verbessert und wie sich die WER verhält, falls das Label durch ein semantisches Matching gebildet wird. Es wird gleich wie in Kapitel 5.6 vorgegangen. Der einzige Unterschied ist, dass das Label im Datenset anhand der WordSim-Formel (Formel 4.1) gebildet wurde.

5.7.1. Training

Beim Training mit diesem Label fällt auf, dass die Feature Importance in Abbildung 44 nun deutlich umverteilt wurden. Es überrascht nicht, dass WordSim nun viel wichtiger ist. Allerdings hat das normale Agreement nach wie vor einen eher grösseren Einfluss. Umgekehrt war beim «string equality» Ansatz WordSim nahezu irrelevant.

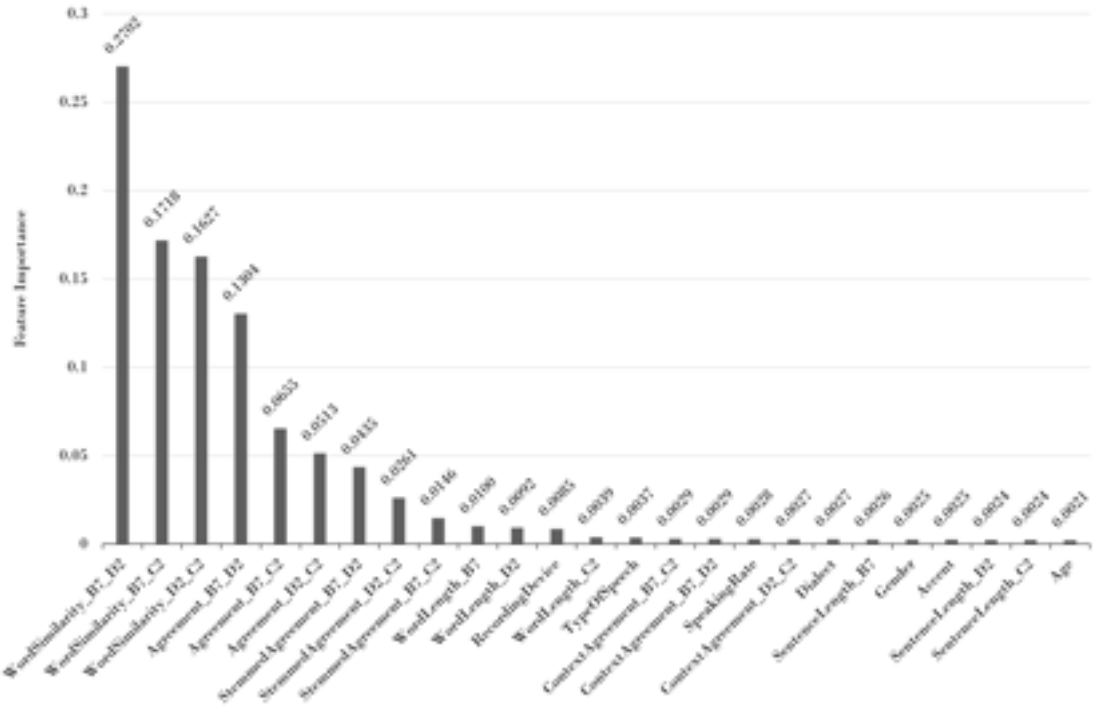


Abbildung 44.: Word Sim Feature Importance

5.7.2. Resultate

Auf den spontanen Korpora werden mit dem semantischen Ansatz leicht bessere Resultate erzielt als mit dem «string equality» Ansatz in Kapitel 5.6. Die Tabelle 10 zeigt, dass die WER auf spontaner Sprache mit semantischem Label im Vergleich zum «string equality» Ansatz leicht sinkt. Auf den anderen Korpora gibt es sowohl Verbesserungen als auch Verschlechterungen, jedoch nur im ‰- Bereich. Bei der SemDist ist die Lösung mit semantischem Label allerdings in den meisten Fällen besser geworden.

	WER		SemDist	
corpus	**string equality**	**wordsim**	**string equality**	**wordsim**
ami_segmented_h	0,3844	**0,3839**	0,2482	**0,2468**
ami_segmented_mix	0,7000	**0,6945**	0,4124	**0,4066**
commonvoice	0,0864	**0,0862**	0,073	**0,727**
librispeech_clean	**0,0654**	0,0701	**0,0565**	0,057
librispeech_other	**0,1486**	0,1508	**0,1363**	0,1366
st	0,0346	**0,0318**	0,0237	**0,0216**
tedlium_segmented	**0,0609**	0,0639	0,0367	**0,0358**
voxforge	**0,0743**	0,0776	0,0595	**0,0593**

Tabelle 10.: Vergleich WER und SemDist für string equality und semantisches Label. Markierte Werte sind bessere Werte

5.7.3. Fazit

Das Experiment zeigt, dass es kaum einen Einfluss auf die Performance hat, ob das Label mit «string equality» oder mit Word Similarity bestimmt wird. Die Unterschiede sind sehr gering. Dennoch erscheint ein semantisches Label sinnvoll, um Sonderfälle, wie Vertauschungen von Einzahl und Mehrzahl, richtig zu klassifizieren.

5.8. BERT Ensemble-Algorithmus

Dieser Ensemble-Algorithmus untersucht die Wirksamkeit der Anwendung eines Masked language modeling auf die Ausgabe der ASR Systeme. BERT hat bei mehreren NLP Tasks den Stand der Technik auf ein neues Level gehoben, daher die Frage, ob man diese Fähigkeit in einem Ensemble-Algorithmus ausnutzen kann, um das richtige Wort auszuwählen. Im Abschnitt 5.8.1 wird BERT eingeführt.

5.8.1. Bidirectional Encoder Representations from Transformers

Bidirectional Encoder Representations from Transformers, oder kurz BERT, ist ein Sprachdarstellungsmodell, das von Google 2019 veröffentlicht wurde [51]. Aus Anwendersicht wird BERT als vortrainiertes Modell bereitgestellt, das auf den spezifischen Anwendungsfall abgestimmt werden kann. Das vortrainierte Modell wurde auf Wikipedia [52] und dem Bookcorpus [53] trainiert, beide Datensätze beinhalten mehrere Gigabyte nicht annotierte Daten. BERT wurde mit zwei Aufgaben vortrainiert. Bei Masked language modeling wurden teile eines Satzes maskiert und das Modell musste vorhersagen, was im maskierten Wort steht. Bei der zweiten Aufgabe geht es um Next sentence prediction, bei dieser Aufgabe werden zwei maskierte Sätze konkateniert und das Modell muss entscheiden, ob diese zwei Sätze nacheinander vorgekommen sind. Durch diese zwei Aufgaben lernt BERT die innere Darstellung der englischen Sprache. BERT hat den Stand der Technik bei elf NLP Aufgaben auf ein neues Level gehoben [51].

5.8.2. Algorithmus Aufbau

Dieser Ensemble-Algorithmus untersucht die Wirksamkeit der Anwendung von MLM auf die Ausgaben der ASR Systeme. Wie im Abschnitt 5.8.1 erwähnt ist, wird BERT als vortrainiertes Modell bereitgestellt. Da das CEASR Korpus kein eindeutiges Thema enthält, ist auf ein fine-tuning von BERT verzichtet worden.
Der Algorithmus folgt dem folgenden Ablauf:

1. Vorschläge berechnen, was im maskierten Wort sein könnte: «I have <mask> a new car»
2. Entfernen von Satzzeichen in Vorschlägen
3. Schnittmenge aus Vorschlägen und Hypothesen berechnen
4. Falls eine Übereinstimmung vorhanden ist, wird diese zurückgeben
5. Falls nicht, wird das Wort mit minimaler Levenshtein-Distanz zwischen den Vorschlägen gewählt

5.8.3. Resultate

In der ersten Version wird der obige Algorithmus 5.8.2 auf jedes Wort im Alignment angewendet. Das macht die Bearbeitung des ganzen CEASR Korpus sehr langsam. Die Resultate für die WER und SemDist sind in der Bildern 45 und 46 ersichtlich. Der Algorithmus erreicht in dieser Form keine akzeptablen Resultate.

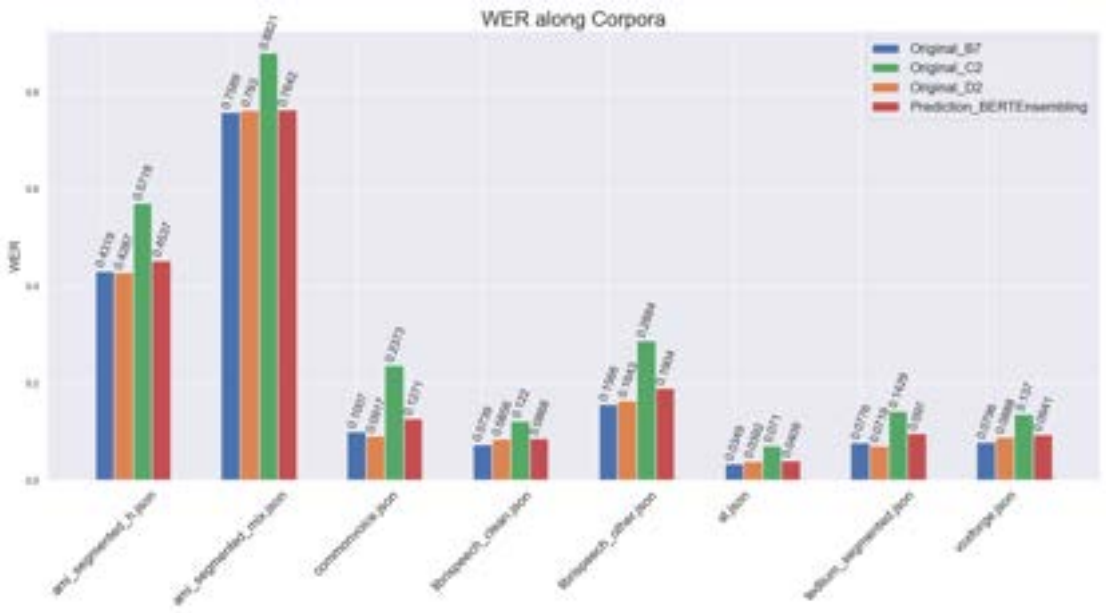


Abbildung 45.: WER BERT Ensemble-Algorithmus

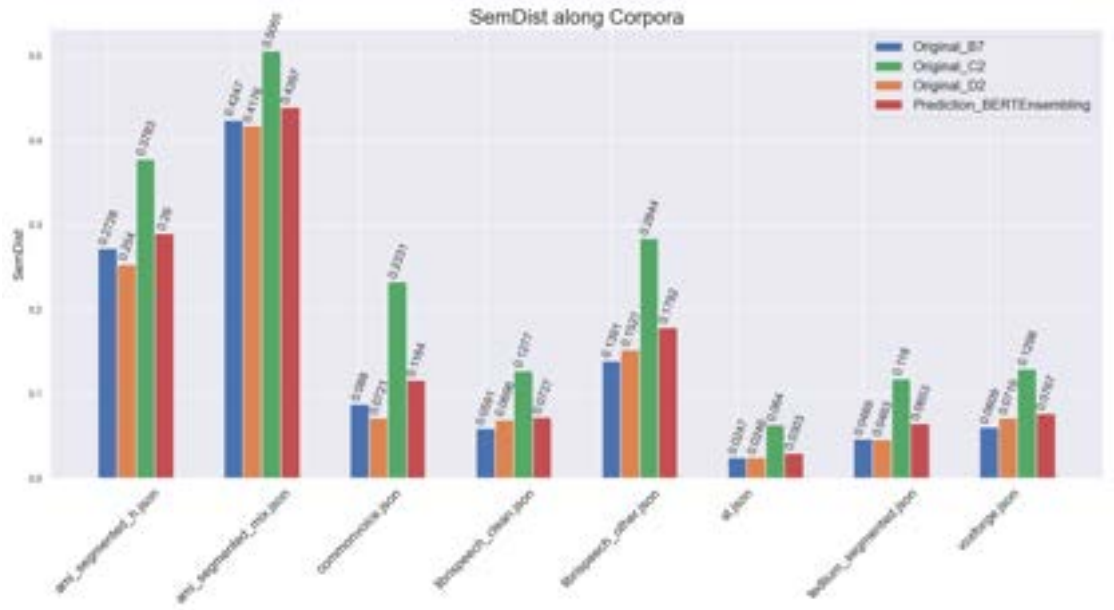


Abbildung 46.: SemDist BERT Ensemble-Algorithmus

Um die Laufzeit sowie die Genauigkeit des Algorithmus zu verbessern, wird dieser mit Voting aus dem Kapitel 5.2 verbunden, d.h. nur wenn die drei Systeme in ihren Hypothesen keine Übereinstimmung finden, wird BERT angewendet. Durch diese Erweiterung kann die Verarbeitungszeit um 90 % reduziert werden. Die berechnete WER auf den Systemen B7, D2 und C2 ist in der Abbildung 47 ersichtlich. Im Vergleich zum puren Voting 5.2 werden keine hohen Verbesserungen erreicht. BERT kommt in diesem Ensemble-Algorithmus über alle Korpora hinweg bei 6 % der Fälle zum Einsatz, wobei dieser Schnitt stark durch den Ausreisser Commonvoice erhöht wird. Auf diesem kleinen Korpus wird BERT für 26 % der Wörter hinzugezogen. Das Sprachmodell von BERT findet im Durchschnitt bei 15 % der Wörter einen Treffer und kann das gefundene Wort zurückgeben.

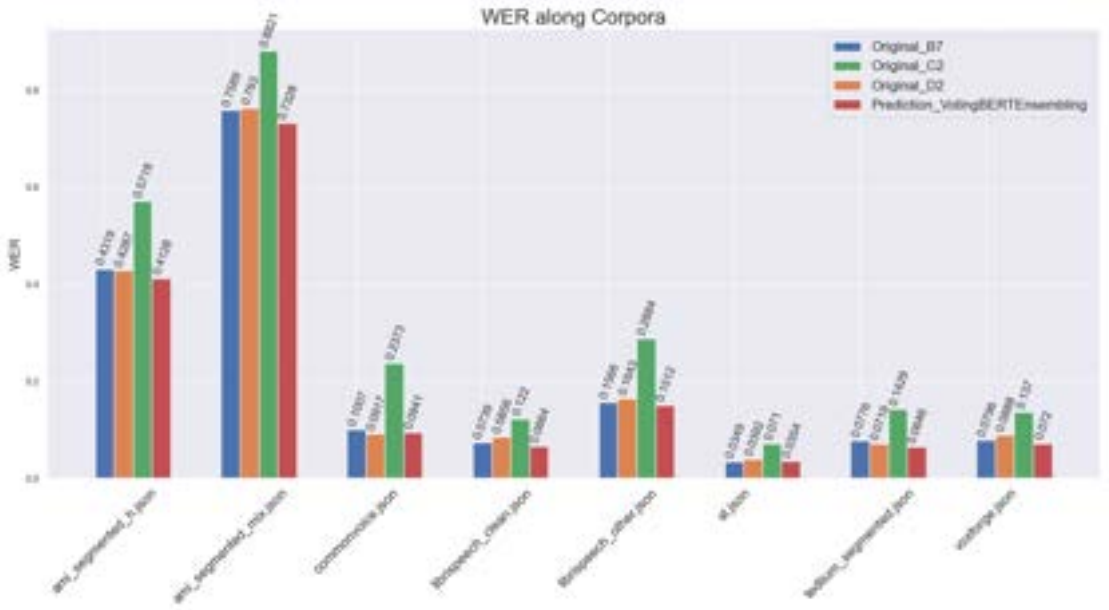


Abbildung 47.: WER Voting BERT Ensembling

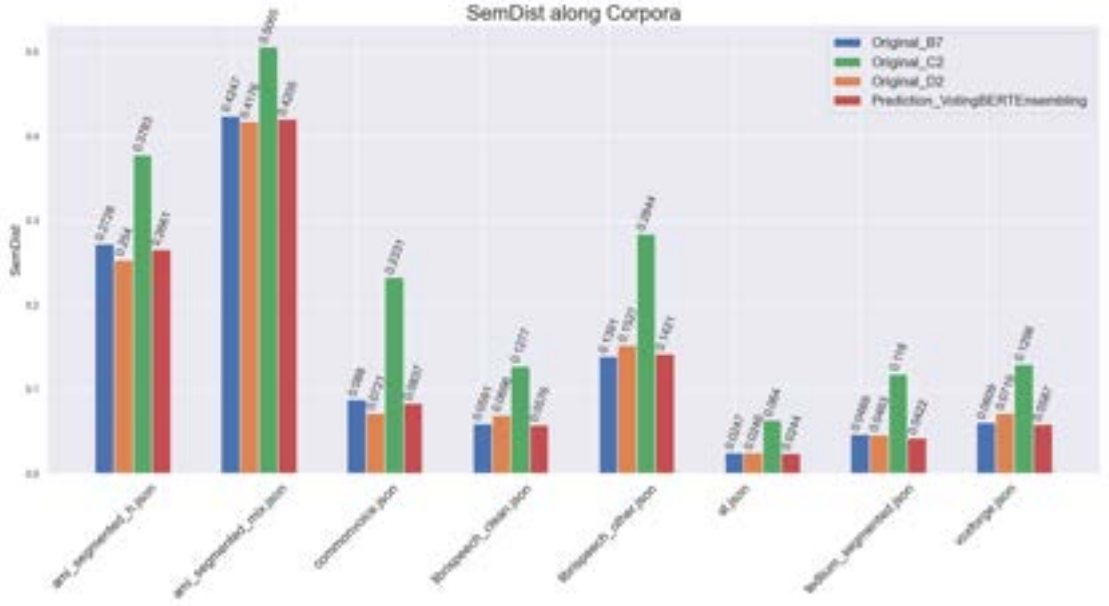


Abbildung 48.: SemDist Voting BERT Ensembling

Für Fälle, in welchen keine Schnittmenge zwischen dem BERT Output und den Hypothesen von den Systemen vorhanden sind, wurden mehrere Optionen ausprobiert. Die Beste, die gefunden wurde, ist die Selektion des Wortes über die minimale Levenshtein-

Distanz zwischen dem BERT Output und System-Hypothesen. Diese Implementierung wurde für die Resultate in den Abbildungen 47 und 48 angewendet.
Eine weitere Option, welche keine besseren Resultate lieferte, ist die Verwendung von phonetischer Codierung mit dem American Soundex Algorithmus. Die Resultate dieser Implementierung sind auf allen Korpora etwa zwei Prozent schlechter als die Levenshtein-Distanz Implementierung in Abbildung 47.

5.8.4. Analyse von fehlerhaften Klassifizierungen

Auf dem gesamten CEASR Korpus wird für 8253 Wörter durch BERT keine Übereinstimmung in den Ausgaben der Systeme gefunden. Ein Vergleich mit der Referenz zeigt, dass bei etwas mehr wie 25 % eine der drei Hypothesen das richtige Wort enthält. In 56 % der Fälle ist das Wort im System B7 vorhanden. Das erklärt, warum die Ensemble-Methode ähnlich gute Werte erreicht, wenn bei keiner Übereinstimmung als Vorhersage das Wort von dem System B7 zurückgegeben wird.

5.8.5. Fazit

Eine Ensemble-Implementierung nur auf Basis von MLM bringt wie gezeigt keine tiefen WER Resultate. Durch die Kombination mit Voting konnten im Vergleich mittelmässige Resultate erzielt werden. Jedoch wird in der kombinierten Implementierung die Anwendung von MLM auch stark minimiert, wodurch MLM keine grosse Gewichtung auf das Resultat mehr hat.

5.9. Boosted BERT und Boosted Perplexities

In der Confusion Matrix (Abbildung 40) vom Best-Possible Estimator in Kapitel 5.6 ist ersichtlich, dass die Klassifikation relativ gut ist, falls zwei Systeme als richtig klassifiziert wurden. Falls allerdings nur ein System klassifiziert wird, ist so ist die Vorhersage eher ungenau. Dieses Kapitel untersucht, ob die WER und SemDist verbessert wird, wenn bei der Klassifikation von nur einem System der BERT- oder Perplexity-Ensemble-Algorithmus angewendet wird. Diese Ansätze werden Boosted BERT und Boosted Perplexities genannt.

5.9.1. Resultate

In Tabelle 11 sind die erreichten WER dargestellt. Die WER von Boosted BERT und Boosted Perplexities werden absolut mit der WER vom Best-Possible Estimator mit semantischem Label (Kapitel 5.7) verglichen. Tabelle 12 zeigt denselben Vergleich für die SemDist.

	Original WER			Predicted WER		
Corpus Name	B7	D2	C2	BPE - WordSim (5.7)	Boosted BERT	Boosted Perplexity
ami_segmented_h	0,4319	0,4287	0,5718	0,3839	(-2,93 %) 0,4132	(-3,59 %) 0,4198
ami_segmented_mix	0,7589	0,763	0,8821	0,6945	(-3,76 %) 0,7321	(-4,84 %) 0,7429
commonvoice	0,1007	0,0917	0,2373	0,0862	(-0,75 %) 0,0937	(-0,38 %) 0,09
librispeech_clean	0,0739	0,0856	0,122	0,0701	(0,42 %) 0,0659	(0,07 %) 0,0694
librispeech_other	0,1556	0,1643	0,2884	0,1508	(0,17 %) 0,1491	(-0,49 %) 0,1557
st	0,0349	0,0392	0,071	0,0318	(-0,32 %) 0,035	(-0,22 %) 0,034
tedlium_segmented	0,0776	0,0719	0,1429	0,0639	(0,03 %) 0,0636	(-0,11 %) 0,065
voxforge	0,0796	0,0888	0,137	0,0776	(0,53 %) 0,0723	(-0,23 %) 0,0799

Tabelle 11.: Vergleich der SemDist für Best-Possible Estimator, Boosted BERT und Boosted Perplexity. In Klammern ist jeweils der Vergleich mit dem Best-Possible Estimator zu sehen

	Original SemDist			Predicted SemDist		
Corpus Name	B7	D2	C2	BPE - WordSim (5.7)	Boosted BERT	Boosted Perplexity
ami_segmented_h	0,2728	0,254	0,3783	0,2468	(-1,77 %) 0,2645	(-1,05 %) 0,2573
ami_segmented_mix	0,4247	0,4176	0,5065	0,4066	(-1,36 %) 0,4202	(-0,95 %) 0,4161
commonvoice	0,088	0,0721	0,2331	0,0727	(-1,05 %) 0,0832	(-0,26 %) 0,0753
librispeech_clean	0,0591	0,0696	0,1277	0,057	(0,05 %) 0,0565	(0 %) 0,057
librispeech_other	0,1391	0,1527	0,2844	0,1366	(-0,21 %) 0,1387	(-0,22 %) 0,1388
st	0,0247	0,0246	0,064	0,0216	(-0,23 %) 0,0239	(-0,09 %) 0,0225
tedlium_segmented	0,0469	0,0463	0,118	0,0358	(-0,48 %) 0,0406	(-0,24 %) 0,0382
voxforge	0,0609	0,0609	0,1298	0,0593	(0,09 %) 0,0584	(-0,12 %) 0,0605

Tabelle 12.: Vergleich der SemDist für Best-Possible Estimator, Boosted BERT und Boosted Perplexity. In Klammern ist jeweils der Vergleich mit dem Best-Possible Estimator zu sehen

5.9.2. Fazit

Offensichtlich eignen sich BERT und Perplexities nicht für Fälle, wo das Best-Possible Estimator Modell nur ein System vorschlägt. Zum einen zeigen die Resultate, dass diese Ansätze vor allem auf spontaner Sprache schlechter als der reine Best-Possible Estimator funktionieren. BERT funktioniert dabei ein wenig besser als Perplexities. Zum anderen scheint es in der Praxis viele Fälle zu geben, in welchen auch ein einzelnes System richtig klassifiziert wurde. Folglich müsste der Best-Possible Estimator weiter verbessert werden, um die Vorhersagen genauer und somit diesen Ansatz tauglicher zu machen.

6. Resultate

Diese Kapitel liefert dem Leser einen Überblick über alle erreichten Resultate und Ergebnisse aus den vorherigen Kapiteln.

6.1. n-Align Algorithmus

Der vorhandene Triple-Alignment-Algorithmus wurde erweitert, sodass eine beliebige Anzahl von Hypothesen an eine Basishypothese angeglichen werden kann. Das Resultat wird in ein JSON-Format geschrieben, welches vom Alignment UI verarbeitet werden kann.

6.2. Alignment UI

Das Alignment UI stellt den Output des n-Align Algorithmus in einer visuell übersichtlichen Form dar. In der farbcodierten Übersicht fallen Unregelmässigkeiten sofort auf und können in einer Detailansicht analysiert werden.
Das Alignment UI kann verwendet werden, um Alignments in ihrer Qualität zu bewerten und Fehler im Alignment zu diagnostizieren.

Abbildung 49.: Alignment UI Implementierung

6.3. Alignment Qualität

Mit der Auswertung mit dem Alignment UI und einer einfachen Heuristik kann davon ausgegangen werden, dass das Alignment im Grossen und Ganzen gut funktioniert. Es können in unter 1 % der Alignments Fehler auftreten, welche die Ensemble-Methoden direkt beeinflussen könnten.

6.4. Best-Possible

Der Best-Possible Ansatz im Kapitel 4.5 zeigt, dass die WER und die SemDist durch das Anwenden von Ensemble-Algorithmen deutlich verbessert werden könnten. In der ersten Spalte der Tabelle 13 (WER) bzw. Tabelle 14 (SemDist) sind die theoretisch bestmöglichen Werte auf jedem Korpus ersichtlich. Für spontane Sprache (AMI) liegt die theoretisch grösste, absolute Verbesserung der WER bei 12–20 %. Die restlichen WER Verbesserungen liegen im Bereich von 3–6 %. Die absoluten SemDist-Verbesserungen liegen allesamt im Bereich von 1–8 %

6.5. Feature Extraction Pipeline

Es wurde eine Feature-Extraction Pipeline implementiert, um ein Best-Possible Datenset zu erstellen. Das Datenset ist für die Klassifikation des richtigen Systems auf Wortbasis konzipiert. Es wurde aus einem Alignment zwischen der Referenz aus CEASR und allen Hypothesen aus B7,D2,C2 sowie Metadaten aus CEASR gebildet. Das resultierende Datenset umfasst in einer unbalancierten Version fast 250'000 und in einer balancierten Version fast 100'000 Datensätze.

6.6. Ensemble-Methoden

Es wurden total neun Ensemble-Methoden implementiert und ausgewertet. Tabelle 13 zeigt die WER auf allen Korpora von allen implementierten Methoden. Tabelle 14 zeigt die SemDist auf allen Korpora von allen implementierten Methoden. Das beste Modell, der Best-Possible Estimator, erreicht auf den AMI-Korpora eine WER von 38,39 % bzw. 69,45 %, was einer Verbesserung von 5 %-6 % gegenüber B7 bzw. D2 entspricht. Durchschnittlich kann dieses Modell die WER gegenüber B7 bzw. D2 um 2 % auf 19,43 % senken. Um das Best-Possible zu erreichen, wäre eine Senkung um weitere 5 % nötig.

6.7. Weitere Resultate und Erkenntnisse

Welches der drei Systeme als Primärhypothese verwendet wird, verändert das Alignment stark. Dies hat einen Einfluss im Rahmen von 0,5 % auf die resultierende WER bzw. SemDist.
Der Vergleich in Kapitel 4.7 zeigt, dass fünf statt drei Systeme die theoretische Verbesserung leicht erhöhen. Auf dem ami-mix Korpus könnten die zwei zusätzlichen Systeme 4 % Verbesserung gegenüber drei Systemen aufweisen. Auf allen anderen Korpora liegen die Verbesserungen im Bereich von 0,1-0.6%. Auf die praktische Anwendung mit Voting ist der Einfluss der zwei Systeme negativ. Gegenüber drei Systemen ist ein einfacher Voting-Algorithmus mit 5 Systemen zwischen 2 und 5 % WER schlechter.
Die Cross WER kann verwendet werden, um den Nutzen von Ensemble-Methoden a priori zu beurteilen. Eine tiefe Cross WER deutet auf eine niedrige Diversität des Ensembles und somit auf einen geringen Nutzen hin. Eine hohe Cross WER kann als Indikator dienen, dass sich Ensemble-Methoden lohnen würden, um die Vorhersagen zu verbessern.

		Kein Ensemble			Ensemble-Methoden								
Corpus Name	**BestPossible**	**B7**	**D2**	**C2**	**Voting**	**Perplexities**	**Hybrid**	**WordEmbeddings**	**Best-Possible Estimator**	**BPE - WordSim**	**BERT**	**BoostedBert**	**BoostedPerplexity**
ami_segmented_h	0,3044	0,4319	0,4287	0,5718	0,4114	0,466	0,4148	0,4107	0,3844	**0,3839**	0,4128	0,4132	0,4198
ami_segmented_mix	0,5673	0,7589	0,763	0,8821	0,7362	0,7944	0,7419	0,7696	0,7	**0,6945**	0,7328	0,7321	0,7429
commonvoice	0,0461	0,1007	0,0917	0,2373	0,0919	0,1358	0,0911	0,0918	0,0864	**0,0862**	0,0941	0,0937	0,09
librispeech_clean	0,0393	0,0739	0,0856	0,122	0,0653	0,0918	**0,0652**	0,0656	0,0654	0,0701	0,0664	0,0659	0,0694
librispeech_other	0,0991	0,1556	0,1643	0,2884	0,1497	0,1993	0,1505	0,151	**0,1486**	0,1508	0,1512	0,1491	0,1557
st	0,0181	0,0349	0,0392	0,071	0,0355	0,0469	0,036	0,0352	0,0346	**0,0318**	0,0354	0,035	0,034
tedlium_segmented	0,0296	0,0776	0,0719	0,1429	0,0636	0,0936	0,0621	0,0621	**0,0609**	0,0639	0,0646	0,0636	0,065
voxforge	0,0451	0,0796	0,0888	0,137	0,0722	0,101	0,0732	0,0735	0,0743	0,0776	**0,072**	0,0723	0,0799
average	0,1436	0,2141	0,2167	0,30656	0,2032	0,2411	0,2044	0,20744	**0,1943**	0,1949	0,20366	0,2031	0,207

Tabelle 13.: WER Vergleich aller implementierten Ensemble-Methoden. Die besten Resultate sind markiert.

		Kein Ensemble			Ensemble-Methoden								
Corpus Name	**BestPossible**	**B7**	**D2**	**C2**	**Voting**	**Perplexities**	**Hybrid**	**WordEmbeddings**	**Best-Possible Estimator**	**BPE - WordSim**	**BERT**	**BoostedBert**	**BoostedPerplexity**
ami_segmented_h	0,1943	0,2728	0,254	0,3783	0,2631	0,2852	0,2603	0,2557	0,2482	**0,2468**	0,2661	0,2645	0,2573
ami_segmented_mix	0,3482	0,4247	0,4176	0,5065	0,4212	0,4352	0,4184	0,4204	0,4124	**0,4066**	0,4205	0,4202	0,4161
commonvoice	0,0408	0,088	**0,0721**	0,2331	0,0796	0,119	0,0768	0,0781	0,073	0,0727	0,0837	0,0832	0,0753
librispeech_clean	0,0344	0,0591	0,0696	0,1277	**0,0553**	0,0762	0,0561	0,0557	0,0565	0,057	0,0576	0,0565	0,057
librispeech_other	0,0963	0,1391	0,1527	0,2844	0,137	0,1759	0,1371	0,1393	**0,1363**	0,1366	0,1421	0,1387	0,1388
st	0,0117	0,0247	0,0246	0,064	0,024	0,0342	0,0241	0,0244	0,0237	**0,0216**	0,0244	0,0239	0,0225
tedlium_segmented	0,0191	0,0469	0,0463	0,118	0,0406	0,0661	0,0394	0,04	0,0367	**0,0358**	0,0422	0,0406	0,0382
voxforge	0,0355	0,0609	0,0609	0,1298	0,0577	0,0797	0,0578	**0,0567**	0,0595	0,0593	0,0587	0,0584	0,0605
average	0,0975	0,1395	0,1372	0,2302	0,1348	0,1589	0,1338	0,1338	0,1308	**0,1296**	0,1369	0,1358	0,1332

Tabelle 14.: SemDist Vergleich aller implementierten Ensemble-Methoden. Die besten Resultate sind markiert.

7. Diskussion und Ausblick

Diese Arbeit zeigt auf, wie mit Alignment und Ensemble-Methoden zum einen die Transkripte von ASR-Systemen verbessert und zum anderen wertvolle Auswertungen gemacht werden können.

7.1. Alignment

Das Alignment ist die absolute Grundlage für die implementierten Ensemble-Methoden. Mit dem n-Align Algorithmus wurde eine Möglichkeit geschaffen, beliebig viele Hypothesen anzugleichen. Neben dem Implementieren von Ensemble-Methoden kann dies auch für Datenanalysen verwendet werden. Der Best-Possible Ansatz und das extrahierte Datenset demonstrieren weitere Möglichkeiten des Alignments.

Ein Alignment ist eine grosse, unübersichtliche Datenmenge. Es stellt sich heraus, dass es schwer ist, die Qualität eines Alignments zu beurteilen. Heuristisch lassen sich potenzielle Fehler im Alignment finden. Diese potenziellen Fehler müssen allerdings händisch bewertet werden. Um diese Bewertung zu beschleunigen und das Alignment besser zu verstehen, kann das implementierte Alignment UI verwendet werden.

Die vorhandene Implementation des Alignments ist von der Alignment-Implementation in ROVER [35] inspiriert. Der Algorithmus wurde jedoch an einigen Stellen angepasst und weicht von der Theorie der WTN ab. Es ist unklar, ob von einer Implementation nach ROVER profitiert werden könnte. Insbesondere werden in ROVER Deletions als problematisch betrachtet. Ein solches Verhalten ist während dieser Arbeit nicht beobachtet worden und müsste angeschaut werden.

Schlussendlich kann man sagen, dass das Alignment schon fast produktiv eingesetzt werden könnte. Der Algorithmus könnte noch ein wenig in seiner Laufzeit optimiert werden. So dauert das Alignment pro System für Hypothesen mit 2000 Wörtern etwa 3 Minuten. Das Kombinieren der Alignments mit n-Align ist allerdings schnell und bedarf keiner Verbesserung.

7.2. Ensemble-Methoden

Ensemble-Learning ist ein naheliegendes Verfahren, um die Vorhersagekraft zu verbessern. Mit dieser Arbeit konnte gezeigt werden, dass Ensemble-Methoden für ASR durchaus Potenzial haben können. Mit dem Best-Possible Ansatz wurde ein Weg gefunden, ein

theoretisch bestmögliches Ensemble zu definieren. Es wurden diverse Ensemble-Ansätze vorgestellt, implementiert und untersucht. Dabei wurden moderate Verbesserungen der WER von 6 % auf spontaner Sprache und von bis zu 2 % auf vorgelesener Sprache erreicht. Diese Verbesserungen sind dabei nicht bahnbrechend. Spontane Sprache bleibt weiterhin mit durchschnittlichen Fehlerraten von über 30 % schwer leserlich. Die Systeme sind auf vorgelesener Sprache auch ohne die 2 % Verbesserung schon so gut, dass sich die Verwendung von drei Systemen wirtschaftlich und auch technisch schwer begründen lassen würde.

Ein weiteres Phänomen zeigt sich im Vergleich mit den originalen Systemen. Die Resultate zeigen, dass die meisten Ensemble-Methoden nie schlechter als das beste, originale System (B7 oder D2) sind. Weil nicht auf jedem Korpus dasselbe originale System am besten ist, könnte das Ensemble also auch verwendet werden, um zumindest sicherzustellen, dass immer die beste Performance der Systeme erreicht wird.

Um mit Ensembles eine noch bessere Lösung zu finden, wäre der nächste Schritt eine detaillierte Analyse der aktuell besten Methode und des Best-Possible mit dem Fokus auf spontaner Sprache. Das Ziel wäre es, für jeden Eintrag im Best-Possible sagen zu können, wieso genau ein System richtig liegt. Ferner wurde in dieser Arbeit festgestellt, dass viele Systeme alternative Hypothesen ausgeben könnten. Diese könnten ebenfalls in eine weitere Methode einfliessen.

Die Outputs von CEASR aus 2019 können den Stand der Technik aus 2022 auch mit Ensemble-Learning nicht erreichen. Es wäre aus diesem Grund interessant, eine neue Version von CEASR mit aktuellen Stand der Technik Systemen zu erstellen und die Experimente dieser Arbeit damit zu wiederholen. Es spricht nach aktuellem Kenntnisstand nichts dagegen, dass auch mit modernsten Systemen die Transkripte durch das Anwenden von Ensemble-Methoden verbessert werden könnten.

Letztlich muss man sich allerdings auch fragen, wann und in welchem Umfang sich das Anwenden von Ensemble-Methoden überhaupt lohnt. Jedes verwendete System verursacht in einem produktiven Umfeld Kosten. Obwohl die Performance mit jedem System potenziell steigt, können nicht unlimitiert viele Systeme kombiniert werden. Die Verbesserungen von Ensemble-Learning sind zudem auch nicht immer nötig. Die 2% Verbesserung auf vorgelesener Sprache werden das Transkript wahrscheinlich in den seltensten Fällen wirklich lesbarer machen. Es bräuchte also Methoden, um spontane Sprache zu erkennen, bevor Ensemble-Methoden angewendet werden. Mit der Cross WER wurde dies kurz erkundet und festgestellt, dass die Cross WER möglicherweise für so eine Erkennung geeignet wäre. Es sind aber weitere Nachforschungen in diesem Bereich nötig.

Die erreichten Lösungen sind für einen produktiven Einsatz in Interscriber noch nicht ausgereift genug. Als Erstes ist die Auswirkung auf den Stand der Technik noch unklar. Als Zweites sind die Verbesserungen eher moderat. Es ist unklar, ob sich die Verbesserungen angesichts der erhöhten Komplexität und Kosten lohnen würden. Als Drittes

müsste die Laufzeit der Methoden verbessert werden. Vor allem das iterative Aufbauen von Word-Embeddings hat einen grossen Overhead und verlangsamt den Prozess. Als Viertes ist unklar, ob die produzierten Hypothesen auch wirklich lesbarer sind. Hierzu wäre eine weitergehende Analyse nötig. Als Letztes fehlen Methoden, um zu erkennen, ob sich das Anwenden von Ensemble-Methoden in einer Situation lohnen würde.

7.2.1. Best-Possible Estimator

Die Ensemble-Methode 'Best-Possible Estimator' stellte sich als sehr interessant heraus. Es konnte gezeigt werden, dass sich aus den Daten und Zusammenhängen der Systeme durchaus lernen lässt, welches System die richtige Wahl in einer gegebenen Situation ist. Dabei ist das wichtigste Feature die Übereinstimmung zwischen verschiedenen Systemen

Es fällt in einigen Metriken auf, dass das Modell relativ gut vorhersagen kann, dass kein System richtig liegt. So liegt die Genauigkeit für 'none' bei 78 % und der F1-Score bei 0.7. Diesem Umstand könnte weiter nachgegangen werden. Sollte sich herausstellen, dass sich so tatsächlich Fehler zuverlässig erkennen lassen würden, wären Integrationen in Interscriber denkbar. Zum Beispiel könnten mögliche Fehler markiert werden und phonetisch ähnliche Wörter vorgeschlagen werden.

Die Auswertungen zeigen, dass die vorgeschlagene Lösung mit xgboost noch viel Verbesserungspotenzial hat. Zum einen verwendet die Lösung die Systeme direkt als Features. Für eine Integration in Interscriber müssten in einem nächsten Schritt also genau die Systeme aus Interscriber anstelle der Systeme aus CEASR evaluiert und verwendet werden. Zum anderen ist klar, dass die gewählten Features noch keine optimale Klassifikation zulassen. Das extrahierte Datenset ist als Proof of Concept anzusehen. Es verbergen sich sehr wahrscheinlich noch Fehler, wie zum Beispiel Datensätze mit identischen Features, aber unterschiedlichen Labels in den Daten. In einem weiteren Schritt müssten also bessere Features gesucht werden, welche diese Probleme lösen. Ferner könnte auch versucht werden, die Loss-Funktion im Training direkt an die WER oder SemDist zu koppeln.

8. Verzeichnisse

Literatur

[1] S. AG. "Interscriber Homepage". [Stand: 3.05.2022]. (2022), Adresse: `https://interscriber.com/`.

[2] myln(at)zhaw.ch. "STT Alignment Repository". [Stand: 18.05.2022]. (2020), Adresse: `https://github.zhaw.ch/mlyn/STT_Alignment`.

[3] S. AG. "SpinningBytes AG Homepage". [Stand: 3.05.2022]. (2022), Adresse: `https://spinningbytes.com/`.

[4] B. Juang und L. Rabiner, "Automatic Speech Recognition - A Brief History of the Technology Development", Jan. 2005.

[5] K. H. Davis, R. Biddulph und S. Balashek, "Automatic recognition of spoken digits", *The Journal of the Acoustical Society of America*, Jg. 24, Nr. 6, S. 637–642, 1952.

[6] S. E. Levinson, L. R. Rabiner und M. M. Sondhi, "An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition", *The Bell System Technical Journal*, Jg. 62, Nr. 4, S. 1035–1074, 1983. DOI: `doi:10.1002/j.1538-7305.1983.tb03114.x`.

[7] D. Wang, X. Wang und S. Lv, "An Overview of End-to-End Automatic Speech Recognition", *Symmetry*, Jg. 11, Nr. 8, 2019, ISSN: 2073-8994. Adresse: `https://www.mdpi.com/2073-8994/11/8/1018`.

[8] A. Baevski, Y. Zhou, A. Mohamed und M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations", *Advances in Neural Information Processing Systems*, Jg. 33, S. 12 449–12 460, 2020.

[9] Y.-A. Chung, Y. Zhang, W. Han et al., "w2v-BERT: Combining Contrastive Learning and Masked Language Modeling for Self-Supervised Speech Pre-Training", *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, S. 244–250, 2021.

[10] M. Riviere, J. Copet und G. Synnaeve, "ASR4REAL: An extended benchmark for speech models", *arXiv preprint arXiv:2110.08583*, 2021.

[11] N. Kanda, G. Ye, Y. Wu et al., "Large-Scale Pre-Training of End-to-End Multi-Talker ASR for Meeting Transcription with Single Distant Microphone", in *Interspeech*, 2021.

[12] T. G. Dietterich, "Ensemble Methods in Machine Learning", in *Multiple Classifier Systems*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, S. 1–15, ISBN: 978-3-540-45014-6.

[13] R. Maclin und D. Opitz, "An Empirical Evaluation of Bagging and Boosting", *Proceedings of the National Conference on Artificial Intelligence*, Feb. 1998.

[14] L. Kuncheva und C. Whitaker, "Measures of Diversity in Classifier Ensembles and Their Relationship with the Ensemble Accuracy", *Machine Learning*, Jg. 51, S. 181–207, Mai 2003. DOI: doi:10.1023/A:1022859003006.

[15] O. Sagi und L. Rokach, "Ensemble learning: A survey", *WIREs Data Mining and Knowledge Discovery*, Jg. 8, Nr. 4, e1249, 2018. DOI: https://doi.org/10.1002/widm.1249. eprint: https://wires.onlinelibrary.wiley.com/doi/pdf/10.1002/widm.1249. Adresse: https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1249.

[16] N. Littlestone und M. K. Warmuth, "The weighted majority algorithm", *Information and computation*, Jg. 108, Nr. 2, S. 212–261, 1994.

[17] L. Breiman, "Bagging predictors", *Machine Learning*, Jg. 24, S. 123–140, 2004.

[18] E. Bauer und R. Kohavi, "An Empirical Comparison of Voting Classification Algorithms : Bagging, Boosting, and Variants", *Machine Learning*, Jg. 36, S. 1–38, Jan. 1996.

[19] Bhanwar Saini, *Ensemble Techniques— Bagging (Bootstrap aggregating)*, https://medium.datadriveninvestor.com/ensemble-techniques-bagging-bootstrap-aggregating-c7a7e26bdc13, Accessed: 2022-04-05, 2021.

[20] L. Breiman, "Random forests", *Machine learning*, Jg. 45, Nr. 1, S. 5–32, 2001.

[21] D. H. Wolpert, "Stacked generalization", *Neural Networks*, Jg. 5, Nr. 2, S. 241–259, 1992, ISSN: 0893-6080. DOI: https://doi.org/10.1016/S0893-6080(05)80023-1. Adresse: https://www.sciencedirect.com/science/article/pii/S0893608005800231.

[22] A. Chatzimparmpas, R. M. Martins, K. Kucher und A. Kerren, "StackGenVis: Alignment of Data, Algorithms, and Models for Stacking Ensemble Learning Using Performance Metrics", *IEEE Transactions on Visualization and Computer Graphics*, Jg. 27, Nr. 2, S. 1547–1557, Feb. 2021, ISSN: 2160-9306. DOI: doi:10.1109/tvcg.2020.3030352. Adresse: http://dx.doi.org/10.1109/TVCG.2020.3030352.

[23] Amol Mavuduru, *A Practical Guide to Stacking Using Scikit-Learn*, https://towardsdatascience.com/a-practical-guide-to-stacking-using-scikit-learn-91e8d021863d, Accessed: 2022-04-05, 2020.

[24] R. E. Schapire, "The boosting approach to machine learning: An overview", *Nonlinear estimation and classification*, S. 149–171, 2003.

[25] Y. Freund und R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting", *Journal of Computer and System Sciences*, Jg. 55, Nr. 1, S. 119–139, 1997, ISSN: 0022-0000. DOI: https://doi.org/10.1006/jcss.1997.1504. Adresse: https://www.sciencedirect.com/science/article/pii/S002200009791504X.

[26] J. H. Friedman, "Greedy function approximation: a gradient boosting machine", *Annals of statistics*, S. 1189–1232, 1999.

[27] A. Natekin und A. Knoll, "Gradient boosting machines, a tutorial", *Frontiers in Neurorobotics*, Jg. 7, 2013, ISSN: 1662-5218. DOI: doi:10.3389/fnbot.2013.00021. Adresse: https://www.frontiersin.org/article/10.3389/fnbot.2013.00021.

[28] J. Pennington, R. Socher und C. D. Manning, "Glove: Global vectors for word representation", in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, S. 1532–1543.

[29] P. Bojanowski, E. Grave, A. Joulin und T. Mikolov, "Enriching word vectors with subword information", *Transactions of the association for computational linguistics*, Jg. 5, S. 135–146, 2017.

[30] Y. Liu, M. Ott, N. Goyal et al., "Roberta: A robustly optimized bert pretraining approach", *arXiv preprint arXiv:1907.11692*, 2019.

[31] J. Woodard und J. Nelson, "An information theoretic measure of speech recognition performance", in *Workshop on standardisation for speech I/O technology, Naval Air Development Center, Warminster, PA*, 1982.

[32] S. Kim, A. Arora, D. Le et al., "Semantic Distance: A New Metric for ASR Performance Analysis Towards Spoken Language Understanding", *arXiv preprint arXiv:2104.02138*, 2021.

[33] A. C. Morris, V. Maier und P. Green, "From WER and RIL to MER and WIL: improved evaluation measures for connected speech recognition", in *Eighth International Conference on Spoken Language Processing*, 2004.

[34] M. A. Ulasik, M. Hürlimann, F. Germann, E. Gedik, F. Benites und M. Cieliebak, "CEASR: a corpus for evaluating automatic speech recognition", in *Proceedings of The 12th Language Resources and Evaluation Conference*, 2020, S. 6477–6485.

[35] J. Fiscus, "A Post-Processing System To Yield Reduced Word Error Rates: Recognizer Output Voting Error Reduction (ROVER)", *IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings*, Aug. 2000.

[36] N. I. of Standards und Technology. "NIST Scoring Toolkit - DP Alignment". [Stand: 17.05.2022]. (2015), Adresse: https://github.com/chinshr/sctk/blob/master/src/sclite/net_dp.c.

[37] P. S. Foundation. "Difflib". [Stand: 18.05.2022]. (), Adresse: https://docs.python.org/3/library/difflib.html.

[38] J. W. Ratcliff und D. E. Metzener, "Pattern-matching-the gestalt approach", *Dr Dobbs Journal*, Jg. 13, Nr. 7, S. 46, 1988.

[39] C. Dimitrakakis und S. Bengio, "Phoneme and sentence-level ensembles for speech recognition", *EURASIP Journal on Audio, Speech, and Music Processing*, Jg. 2011, S. 1–17, 2011.

[40] O. Siohan, B. Ramabhadran und B. Kingsbury, "Constructing ensembles of ASR systems using randomized decision trees", *Proceedings. (ICASSP '05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, Jg. 1, I/197–I/200 Vol. 1, 2005.

[41] L. Deng und J. C. Platt, "Ensemble deep learning for speech recognition", in *INTERSPEECH*, 2014.

[42] N. Vaessen. "JIWER Repository". [Stand: 4.06.2022]. (2021), Adresse: `https://github.com/jitsi/jiwer`.

[43] *Angular*, `https://angular.io`, Accessed: 2022-06-06.

[44] *NgRx*, `https://ngrx.io`, Accessed: 2022-06-06.

[45] A. Ahamad, A. Anand und P. Bhargava, "AccentDB: A Database of Non-Native English Accents to Assist Neural Speech Recognition", in *Proceedings of the 12th Language Resources and Evaluation Conference*, Marseille, France: European Language Resources Association, Mai 2020, S. 5351–5358, ISBN: 979-10-95546-34-4. Adresse: `https://aclanthology.org/2020.lrec-1.659`.

[46] M. G. Elfeky, P. Moreno und V. Soto, "Multi-Dialectical Languages Effect on Speech Recognition: Too Much Choice Can Hurt", *Procedia Computer Science*, Jg. 128, S. 1–8, 2018, 1st International Conference on Natural Language and Speech Processing, ISSN: 1877-0509. DOI: `https://doi.org/10.1016/j.procs.2018.03.001`. Adresse: `https://www.sciencedirect.com/science/article/pii/S1877050918302102`.

[47] E. Grave, P. Bojanowski, P. Gupta, A. Joulin und T. Mikolov, "Learning Word Vectors for 157 Languages", in *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[48] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei und I. Sutskever, "Language Models are Unsupervised Multitask Learners", 2019.

[49] A. Vaswani, N. Shazeer, N. Parmar et al., "Attention is All You Need", 2017. Adresse: `https://arxiv.org/pdf/1706.03762.pdf`.

[50] T. Chen und C. Guestrin, "Xgboost: A scalable tree boosting system", in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, S. 785–794.

[51] J. Devlin, M. Chang, K. Lee und K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", *CoRR*, Jg. abs/1810.04805, 2018. arXiv: `1810.04805`. Adresse: `http://arxiv.org/abs/1810.04805`.

[52] W. Foundation. "Wikimedia Downloads". (), Adresse: `https://dumps.wikimedia.org`.

[53] Y. Zhu, R. Kiros, R. Zemel et al., "Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books", in *The IEEE International Conference on Computer Vision (ICCV)*, Dez. 2015.

Glossar

Hypothese Mit Hypothese ist im ASR Umfeld das Transkript eines Systems gemeint. Das System gibt eine Hypothese über den Inhalt einer Audio-Aufnahme aus. 16, 18–20, 22, 23, 25, 26, 31, 32, 34, 38, 44, 52, 58, 75–77, 91

Referenz Eine Referenz bezeichnet den wahren Inhalt einer Audio-Aufnahme. Die Referenz wird durch Menschen annotiert und wird für Vergleiche mit Hypothesen hinzugezogen. 17

STT-Ensemble-Learning - Analyse Eine Voranalyse der ZHAW zum Thema Ensemble-Learning und Alignment von ASR. Siehe auch [2]. 7, 8, 25, 26, 28

Utterance Eine Utterance (deutsch: Äusserung) bezeichnet eine Audio-Aufnahme eines oder mehreren Sprechern. Meistens werden einzelne Audio-Aufnahmen in einem ASR-Korpus Utterance gennant und eindeutig gekennzeichnet. 8, 17, 23, 25, 29, 30, 35, 38, 40, 41, 44, 45, 47, 50

Weak Learner Ein Weak Learner ist ein Modell, welches nur marginal genauer als ein Zufallsmodell ist.. 13

Akronyme

ASR Automatic Speech Recognition. 16, 18, 23, 58, 73, 74

BERT Bidirectional Encoder Representations from Transformers. 58, 73–75, 77

DP Dynamic Programming. 19

FSM Finite State Machine. 19

GPT Generative Pre-trained Transformer. 58

MER Match Error Rate. 17, 28

MLM Masked language modeling. 73, 74, 77

NLP Natural language processing. 58, 73

NSP Next sentence prediction. 73

ROVER Recogniser output voting error reduction. 23

WER Word Error Rate. 2, 3, 16, 17, 23–30, 38–42, 47–50, 53, 55, 56, 62, 65, 69–72, 74, 75, 77, 80, 81, 84, 85

WIL Word Information Lost. 17, 28

WIP Word Information Preserved. 17, 28

WTN Word Transition Network. 18, 19, 32, 83

Abbildungsverzeichnis

Tabellenverzeichnis

A. Anhang

A.1. Aufgabenstellung

School of Engineering

[Online Administration Praktische Arbeiten]

[Dept. T Admin Tools]

zurück Logout

Bachelorarbeit 2022 - FS: BA22_ciel_03

Allgemeines:

Titel:	Ensemble Methods for Speech Recognition
Anzahl Studierende:	2
Durchführung in Englisch möglich:	Ja, die Arbeit kann vollständig in Englisch durchgeführt werden und ist auch für Incomings geeignet.

Betreuer:

HauptbetreuerIn: Mark Cieliebak, ciel

Zugeteilte Studenten:

Diese Arbeit ist vereinbart mit:
- Lars Mosimann, mosimlar (IT)
- Ralph Scheu, scheural (IT)

Fachgebiet:

AI	Artificial Intelligence
DA	Datenanalyse
ML	Machine Learning
SOW	Software

Studiengänge:

IT	Informatik
WI	Wirtschaftsingenieurwesen

Zuordnung der Arbeit :

CAI	Centre for Artificial Intelligence

Infrastruktur:

benötigt keinen zugeteilten Arbeitsplatz an der ZHAW

Interne Partner :

Es wurde kein interner Partner definiert!

Industriepartner:

Es wurden keine Industriepartner definiert!

Beschreibung:

Project Description:

Automatic Speech Recognition (ASR) has many applications, for instance transcribing interviews and meetings, chatbots, automatic answering for phonecalls etc. The speech-to-text quality has significantly increased in the last years and so has the demand for ASR-based solutions. Tech companies developing ASR systems, such as IBM and Google, are in constant pursuit of reaching the best possible transcription quality, and in some experiments even reach human performance.

However, in many cases the output of a particular ASR system contains large amounts of errors. One potential solution would be to go beyond one ASR system and use/combine the output of several systems to produce the transcript. In other contexts (e.g. sentiment analysis), it has been shown that such "ensemble methods" outperform each single participating system. Is the same possible for ASR?

Project Goals:

- Define a method for comparing ASR system transcriptions on a word level and retrieving the most accurate ones.
- Investigate discrepancies between system transcriptions on a word level.
- Develop a software tool for generating an optimized transcription using the outputs from multiple ASR systems.
- Generate optimized transcriptions for a provided set of utterances and calculate the quality of the meta-system per utterance.

Available Data:

- More than 60hours of manual transcriptions with metadata coming from English and German corpora
- Machine transcriptions produced by state-of-the-art ASR systems
- All data is provided in a unified format as a JSON file.

If you are interested in this challenging topic, please get in touch. We will then meet and discuss the details. Email: ciel@zhaw.ch, Phone: 058 934 72 39.

Informations-Link:

Unter folgendem Link finden sie weitere Informationen zum Thema:
http://www.lrec-conf.org/proceedings/lrec2020/pdf/2020.lrec-1.798.pdf

zurück Logout

A.2. Beispiel n-Alignment

Das unten stehende Alignment ist das fertige Beispiel aus Kapitel 4.3 Hinweis: Das Alignment folgt der Struktur, welche in Anhang A.5 beschrieben ist.


```
{
            "words": [
                {
                    "items": [
                        {
                            "text": "i",
                            "type": "Correct"
                        },
                        {
                            "text": "",
                            "type": "Deletion"
                        }
                    ],
                    "text": "i"
                },
                {
                    "items": [
                        {
                            "text": "saw",
                            "type": "Correct"
                        },
                        {
                            "text": "saw",
                            "type": "Correct"
                        }
                    ],
                    "text": "saw"
                },
                {
                    "items": [
                        {
                            "text": "them",
                            "type": "Correct"
                        },
                        {
                            "text": "them",
                            "type": "Correct"
                        }
                    ],
```

```
                    "text": "them"
                },
                {
                    "items": [
                        {
                            "text": "and",
                            "type": "Insertion"
                        },
                        {
                            "text": "",
                            "type": "Skipped"
                        }
                    ],
                    "text": ""
                },
                {
                    "items": [
                        {
                            "text": "before",
                            "type": "Substitution"
                        },
                        {
                            "text": "be for",
                            "type": "MergedAlignment"
                        }
                    ],
                    "text": "be for"
                }
            ]
        }
```


A.3. Code

Der Code dieser Bachelorarbeit kann in den folgenden Git Repositories gefunden werden.

- Alignment UI: `https://github.zhaw.ch/scheural/ba_alignment_ui`
- Pipelines: `https://github.zhaw.ch/mosimlar/ba_evaluation_pipeline`

A.4. Dokumentation Alignment UI

README.md

Alignment UI

Tool to display alignment of n system. The tool takes a JSON input. For the format read the format section. The project is Single Page Application written with Angular, NgRx and some Angular Material components.

Setup

1. Install latest node version
2. Run `npm install`
3. Start developing

Development server

Run `ng serve` for a dev server. Navigate to `http://localhost:4200/`. The app will automatically reload if you change any of the source files.

JSON Format

The `WER` on utterance level is an optional attribute. If not provied the WER wil be calculated on parsing the uploaded json file.

```
{
    "systems": [
        {
            "name": "D1",
            "alignment_scores": []
        },
        {
            "name": "C1",
            "alignment_scores": [
                {
                    "name": "WER",
                    "score": "0.998"
                }
            ]
        },
        {
```

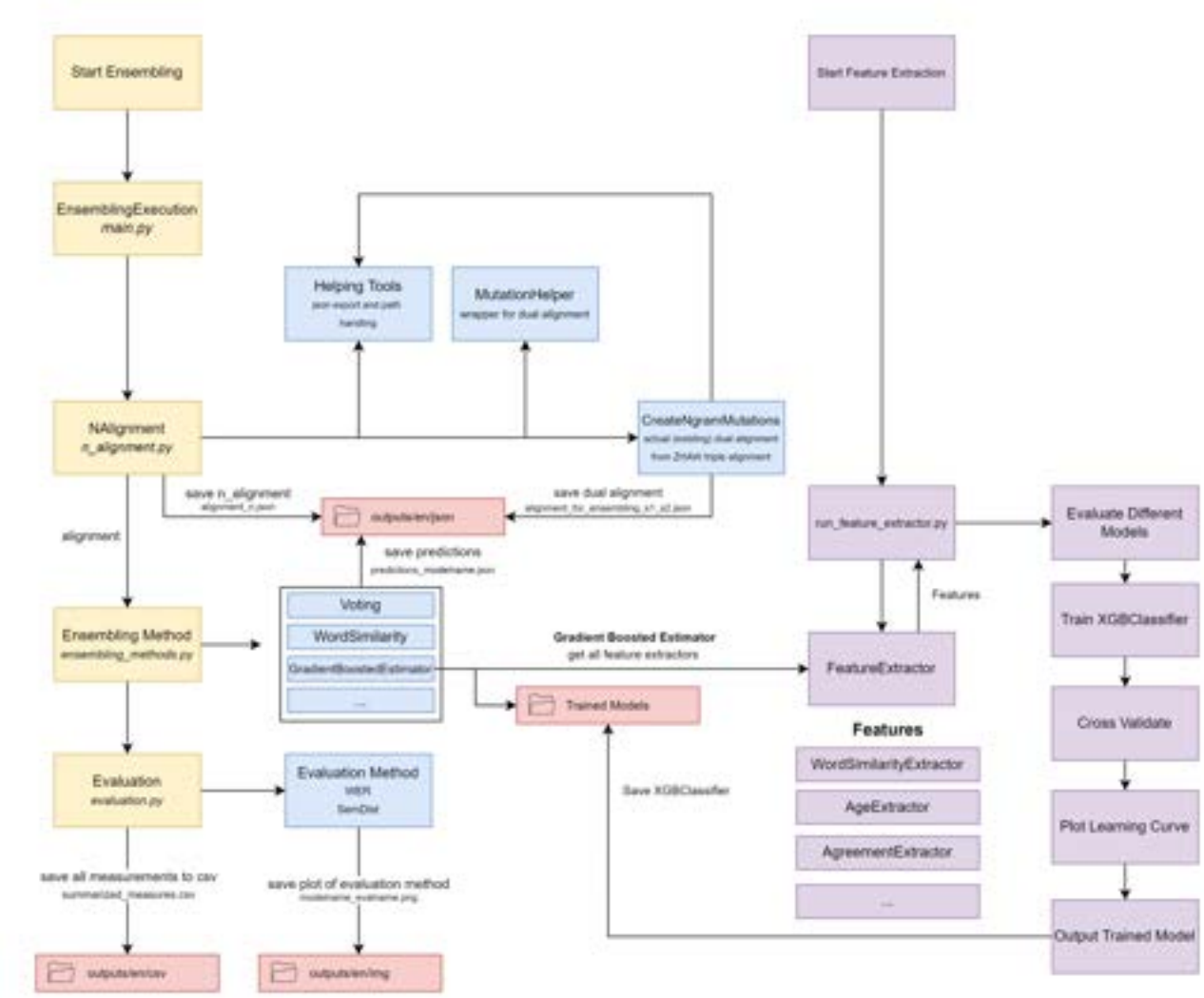
```
            "name": "B1",
            "alignment_scores": [
                {
                    "name": "WER",
                    "score": "0.998"
                }
            ]
        }
    ],
    "alignment": {
        "commonvoice.json": {
            "sample-000335": {
                "reference": "in spite of this i still believed that there wer
                "wer": 0.912,
                "utterance_id": "sample-000335",
                "words": [
                    {
                        "items": [
                            {
                                "text": "in",
                                "type": "Correct"
                            },
                            {
                                "text": "in",
                                "type": "Correct"
                            }
                        ],
                        "text": "in"
                    },
                    {
                        "items": [
                            {
                                "text": "spite",
                                "type": "Correct"
                            },
                            {
                                "text": "spite",
                                "type": "Correct"
                            }
                        ],
                        "text": "spite"
                    }
                    ...
                ]
            }
        }
    }
}
```

A.5. Dokumentation Pipelines

README.md

Ensembling for Speech Recognition

Overview



Getting Started

0. Install conda (miniconda or anaconda). anaconda is recommended as it was used during development.
1. Clone this repository
2. Setup conda environment by running `conda env create -f environment.yml` in a conda-enabled shell
3. From now on, use the conda environment `ensembling_3.7`
4. run the `init_nltk.py` file or execute the following code in the environment:

```
import nltk
nltk.download('averaged_perceptron_tagger')
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('stopwords')
```

5. (*optional*) For CUDA-Support (SemDist), run `conda install pytorch torchvision torchaudio cudatoolkit=11.3 -c pytorch`

6. Download CEASR from this git repo and extract it to a sibling folder to this repository called `data/en`. See the following folder structure for reference:

```
ba_evaluation_pipeline
│   README.md
│   main.py
│   ...
data
   └──en
       └──CEASR_en
       │   B3__ami_segmented_h.json
       │   ...
```

7. The pipelines should now be usable.

Quickstart

To get mostly the same results as in the paper, copy the contents of the archive `important_outputs/quickstart` into the main working dir. Quickstart contains the following files (Most are copies from the `important_outputs/alignments/B7_D2_C2` folder):

- `alignment_n_B7_D2_C2.json` - The full **normal** alignment of B7,D2,C2
- `alignment_n_reference_B7_D2_C2.json` - the full **reference** alignment of B7,D2,C2
- `fe_alignment_reference_train_B7_D2_C2.json` - the 50% split **reference** alignment that can be used for training
- `fe_alignment_reference_test_B7_D2_C2.json` - the 50% split **reference** alignment that can be used for testing (e.g. Best Possible for the 50% split)
  - `alignment_n_reference.json` is a copy of `fe_alignment_reference_test_B7_D2_C2.json` since the pipeline demands that filename for the reference alignment
- `fe_alignment_n_test_B7_D2_C2.json` - the 50% split **normal** alignment that can be used for testing Ensembling Methods
  - `alignment_n.json` is a copy of `fe_alignment_n_test_B7_D2_C2.json` since the pipeline demands that filename for the normal alignment
- `intermediate_hypotheses_D2_REFERENCE.json`, `intermediate_hypotheses_C2_REFERENCE.json`, `intermediate_hypotheses_B7_REFERENCE.json`, `intermediate_hypotheses_B7_D2.json`, `intermediate_hypotheses_B7_C2.json` are the cached intermediate files for all systems / references
- `alignment_for_ensembling_reference_D2.json`, `alignment_for_ensembling_reference_C2.json`, `alignment_for_ensembling_reference_B7.json`, `alignment_for_ensembling_B7_D2.json`,

`alignment_for_ensembling_B7_C2.json` are the cached alignment pairs for all systems / references

Evaluation Pipeline (Alignment, Ensembling and Evaluation)

Basic Usage

1. configure pipeline in `main.py` (the instance of `EnsemblingExecution` at the bottom)
    - configure `lang` (currently only `en` is supported)
    - set `data_directory` to the CEASR-Directory that contains all the json files
    - Add all Systems that should be evaluated into the `configurations` tuple. **NOTE** The first System will be the primary hypothesis.
    - Add all Models that should be evaluated into the `models`-Array. Available options are
        - `BestPossible`
            - **Note**: Since best possible needs a `reference alignment`, it cannot be mixed with other models. Make sure that `BestPossible` is the only entry in the `models` array when evaluating best possible. Using Best Possible will force the alignment to be a reference alignment, rendering the other methods useless and vice-versa.
        - `Voting`
        - `Perplexities`
        - `Hybrid`
        - `WordEmbbedings`
        - `GradientBoostedEstimator`
        - `BERT`
        - `VotingBERT`
        - `BoostedBert`
        - `BoostedHybrid`
        - `BoostedPerplexity`
    - Modify the `get_evals` function to include or exclude different ASR Metrics. Possible Values are:
        - WER: WEREvaluation
        - MER: MerEvaluation
        - WIL: WILEvaluation
        - WIP: WipEvaluation

        - SemDist(Bulk): SemDistEvaluationBulk
        - SemDist(Single): SemDistEvaluation

2. run main.py
    - First launch may take several minutes as multiple models need to be downloaded
3. In a first step, the alignments are created.
    - generating all data from scratch might take several hours for 3 Systems. Alignments for the Systems B7,C2,D2 are provided in the `important_outputs/alignments` folder, for use with Alignment Caching.
4. In the next step, the new prediction of a model is created and evaluated. If the model is complex (like BERT, BestPossible Estimator, Perplexities, ...), this might take a while. Also, if SemDist is selected in `get_evals`, using a GPU is recommended for faster SemDist calculations.
5. Outputs will be written into different folders
    - Alignments and Predictions can be found in `outputs/en/json`
    - Resulting Plots can be found in `outputs/en/img`
    - Plot data can be found in `outputs/en/csv`

Alignment Files

To create the alignment of n systems, mutliple files are created.

- for the alignment between two systems there are two files.
    - `intermediate_hypotheses_sys1_sys2.json` contains the CEASR Hypotheses in an easier to read format.
    - **NOTE**: This file is also used in the evaluation to load the predictions for the original system value. Therefore, all files of this type are needed at all times. (Relevant if using Alignment Caching)
    - **NOTE**: For Reference Alignment sys1 is always `REFERENCE`
    - `alignment_for_ensembling_sys1_sys2` contains the actual alignment from the two systems. NOTE: For Reference Alignment sys1 is always `REFERENCE`
- the merged alignment of the n-systems will be stored in the `alignment_n.json` file. NOTE: Reference Alignments will be stored in `alignment_n_reference.json`

Alignment Data Structure

The following shows the data structure of the final `alignment_n.json` file. The items in the `words` array can be interpreted the following way:

- `text` is the text of the primary hypothesis -> Sys1
- `items` contains the texts and the mutation type of the auxilary hypotheses. The items are ordered the same way as the `configurations` array at the end of the file.

The `alignment_n-reference.json` file is identical except that the `text` of the `word`-object is the reference and there is an entry more inside `items`. It is because of this that the `alignment_n-reference.json` File needs to be interpreted differently than the normal alignment file.


```
{
    "alignment": {
        "corpus1": {...},
        "corpus2": {...},
        "corpus3": {
            "utterance_id": {
               "reference": "okay yeah that should be for",
               "utterance_id": "EN2001a_B_1886_525",
               "words": [
                   {
                       "items": [
                           {
                               "text": "ok",
                               "type": "Substitution"
                           },
                           {
                               "text": "okay",
                               "type": "Correct"
                           }
                       ],
                       "text": "okay"
                   },
                   {
                       "items": [
                           {
                               "text": "yeah",
                               "type": "Insertion"
                           },
                           {
                               "text": "",
                               "type": "Skipped"
                           }
                       ],
                       "text": ""
                   },
                   {
                       "items": [
                           {
                               "text": "that",
                               "type": "Correct"
                           },
```

```
                            {
                                "text": "the",
                                "type": "Substitution"
                            }
                        ],
                        "text": "that"
                    },
                    {
                        "items": [
                            {
                                "text": "should",
                                "type": "Correct"
                            },
                            {
                                "text": "trip",
                                "type": "Substitution"
                            }
                        ],
                        "text": "should"
                    },
                    {
                        "items": [
                            {
                                "text": "be for",
                                "type": "MergedAlignment"
                            },
                            {
                                "text": "before",
                                "type": "Substitution"
                            }
                        ],
                        "text": "be four"
                    }
                ]
            },
        },
        ...
    },
    "configurations": [
        "Sys1",
        "Sys2",
        "Sys3"
    ],
    "language": "en"
}
```


Alignment Caching

Since generating alignments for the whole CEASR-Corpus can take a long time, the Pipeline uses caching extensively by default. When calling `n_alignment`, caching is controlled by the `use_cache` argument. If using cache, the pipeline checks for every file described in the section Alignment Files and uses the files content instead of recreating the file. If e.g only 1 of 2 alignment pairs exist, one will be loaded from cache (the file `alignment_for_ensembling_sys1_sys2`) and the other will be recreated before combining them.Furthermore, if a `alignment_n.json` (`alignment_n_reference.json` for reference alignment) already exists, this file will be loaded as the resulting alignment, skipping the whole alignment process altogether. This will happen regardless if the systems of the alignment are different as configured.

therefore, when using cache, always make sure that the alignments contain the same systems as configured in `main.py`

Ensemble Methods

The Pipeline supports the following Ensemble Methods

- `Voting`
- `Perplexities`
- `Hybrid`
- `WordEmbbedings`
- `GradientBoostedEstimator`
- `BERT`
- `VotingBERT`
- `BoostedBert`
- `BoostedHybrid`
- `BoostedPerplexity`

Most Ensemble Methods are self-explanatory and work-as-is. However, some methods can be configured.

Aggregated subphrases

`Voting`, `Perplexities`, `Hybrid` and `WordEmbbedings` all support the aggregation of subphrases to (potentially) improve readability of the outputs. This feature can be turned on by `specifying aggregate_disagreeing_parts = True` on model instantiation.

Perplexity

the amount of context-words to consider can be configured by setting `self.n` in the constructor of the `PerplexitiesLangModels` class.

Gradient Boosted Estimator

This ensemble methods loads a `XGBClassifier` from a specified file path. The model to load can be configured in `self.model`. Please note that at the current time, the features that the classifier needs are not loaded automatically. The Ensemble Method just loads the same features as the Feature Extraction Pipeline, as they both use the same function to extract the features (`FeatureExtractor.get_extractors`).

When training a new model with the feature extraction pipeline, the final model needs to be loaded in `self.model` as described above.

Implementing a new Method

A new method can be implemented by inheriting from the `Ensembling` class. A minimal implementation needs to implement a constructor and the function `get_word_from_ensembling`. Example:

```
class MyEnsemblingMethod(Ensembling):
    """Always returns primary hypothesis"""

    def __init__(self, aligned_hypotheses, working_directory, model_name, aggregate_d
        super().__init__(aligned_hypotheses, working_directory, "MyCoolModel", aggreg
        print("Initializing MyCoolModel..")

    def get_word_from_ensembling(self, word, all_words, current_index, utt_name, corp
        return word["text"]
```

Tips and Tricks

Update Conda Environment

This command updates the conda environment to a file. Helpful when troubleshooting dependencies. c `conda env update --name myenv --file local.yml --prune`

Use Cached Predictions

To use the predictions of a json file, simply use the following code in `main.py` for setting the predictions. This can be helpful to repeat the scoring of a method without regenerating the predictions.

```
print("Loading prediction from cache...")
path_to_intermediate = self.tools.get_path_to_intermediate()
```

```
pred_file_name = "FILENAME.json"
path_to_pred = os.path.join(path_to_intermediate, pred_file_name)

with open(path_to_pred, "rt", encoding="utf-8") as inp:
    predictions = json.load(inp)
print("Loaded predictions from cache.")
```

Feature Extraction Tool

The feature excation pipeline can be used to train a "Best Possible Estimator". The File `run_feature_extractor.py` demonstrates a possible usage of the Feature Extraction Tool.

The Feature Extraction Tool `feature_extraction/FeatureExtractor.py` needs a path to a reference alignment, the ceasr data path, the normal alignment path, the current working dir and the names of the systems. Calling `extract` will extract all registered features and return them in a `pandas` data-frame for further use.

Use `similarity_labels` to toggle wordsim / string equality approaches of the label.

Feature Extraction Workflow

1. all the input files are split into a train and a test dataset.
2. for every word in the reference aligned dataset all features and the label are extracted and added to the result
   - Some Features (like `WordSimilarity`) are extracted in bulk due to performance reasons
   - These Features are only prepared for extraction in this step
3. A post-processing engine extracts all Bulk Features based on the prepared features and overwrites the prepared data with the real data in the result.

Features

In this Pipeline, there is a class for every feature. The features can be found in the folder `FeatureExtractors`. As described in the paper, there are multiple different types of features.

- Simple Features from CEASR (e.g. Age)
- Features per System (e.g. Word Length)
- Features per System-Pair (e.g. Word Similarity)

Define a new Feature

To define a new feature

1. Implement the feature in a new class that inherits from `Extractor`
    - At minimum, implement the constructor and the `extract` function.
    - Implement the `encode` function if the feature needs to be encoded differently than ordinal. For no encoding just implement the function with `pass`
    - Make sure to give the feature a unique name in the constructor call
2. Register the feature in the `get_extractors` - Method.
3. Thats it, the feature should now get extracted upon calling `extract` on an instance of FeatureExtractor()

Bulk Features

If the feature is a bulk feature, use `extract` to prepare the data for a bulk extraction. Keep in mind to provide a non-bulk implementation in case the feature is extracted from the ensembling pipeline(the pipeline only calls the `extract` function). This can be achieved with the instance-variable `self.from_ensembling`.

Then, implement `extract_bulk`, where the whole data set can be accessed. Use all the data you need to create a new feature for every data-row. For a fully fledged implementation example, please refer to `FeatureExtractors/WordSimilarityExtractor.py`

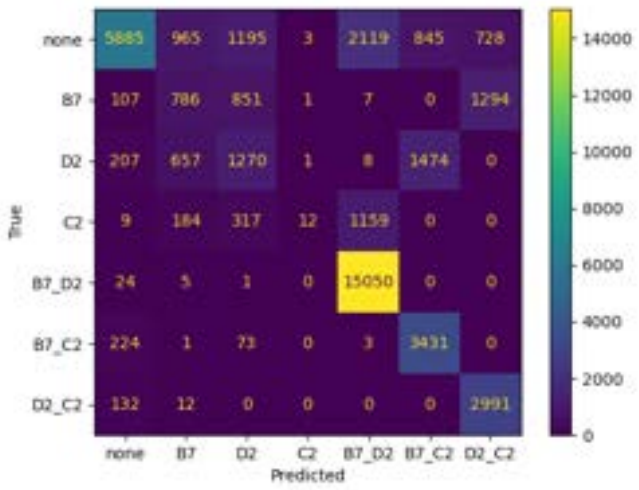
A.6. Modellvergleich: F-Score & Confusion Matrix

A.6.1. Logistic Regression

F1 Scores

	precision	recall	f1-score	support
none	0,89	0,50	0,64	11740
B7	0,30	0,26	0,28	3046
D2	0,34	0,35	0,35	3617
C2	0,71	0,01	0,01	1681
B7_D2	0,82	1,00	0,90	15080
B7_C2	0,60	0,92	0,72	3732
D2_C2	0,60	0,95	0,73	3135

Confusion Matrix

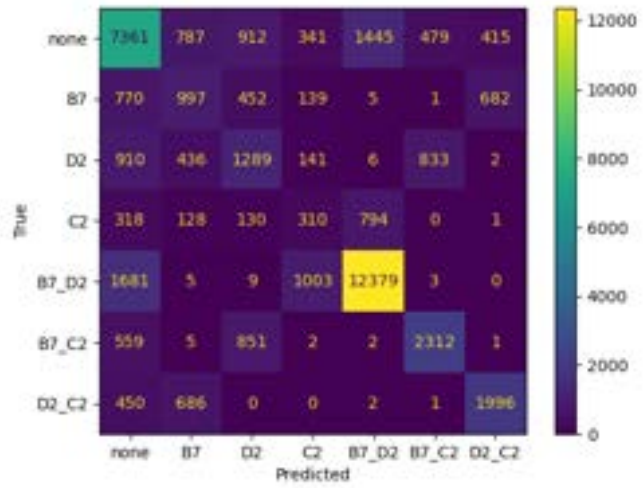


A.6.2. Decision Tree

F1 Scores

	precision	recall	f1-score	support
none	0,61	0,63	0,62	11740
B7	0,33	0,33	0,33	3046
D2	0,35	0,36	0,36	3617
C2	0,16	0,18	0,17	1681
B7_D2	0,85	0,82	0,83	15080
B7_C2	0,64	0,62	0,63	3732
D2_C2	0,64	0,64	0,64	3135

Confusion Matrix

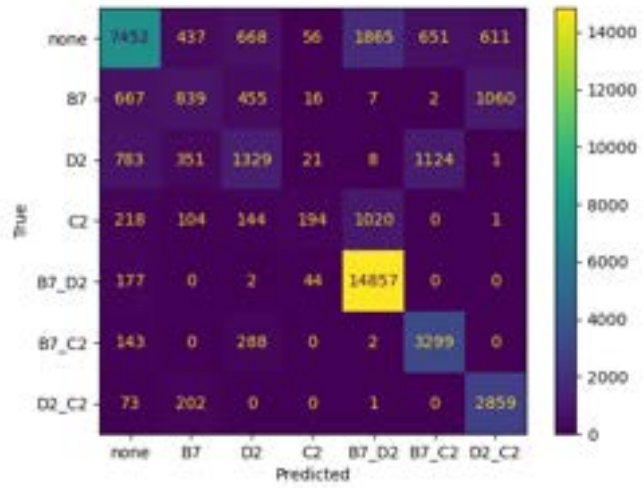


A.6.3. Random Forest

F1 Scores

	precision	recall	f1-score	support
none	0,78	0,63	0,70	11740
B7	0,43	0,28	0,34	3046
D2	0,46	0,37	0,41	3617
C2	0,59	0,12	0,19	1681
B7_D2	0,84	0,99	0,90	15080
B7_C2	0,65	0,88	0,75	3732
D2_C2	0,63	0,91	0,75	3135

Confusion Matrix

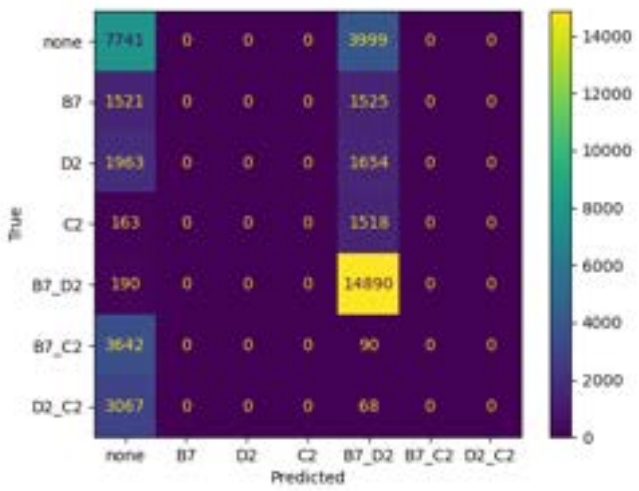


A.6.4. Support Vector Machine

F1 Scores

	precision	recall	f1-score	support
none	0,42	0,66	0,52	11740
B7	0,00	0,00	0,00	3046
D2	0,00	0,00	0,00	3617
C2	0,00	0,00	0,00	1681
B7_D2	0,63	0,99	0,77	15080
B7_C2	0,00	0,00	0,00	3732
D2_C2	0,00	0,00	0,00	3135

Confusion Matrix

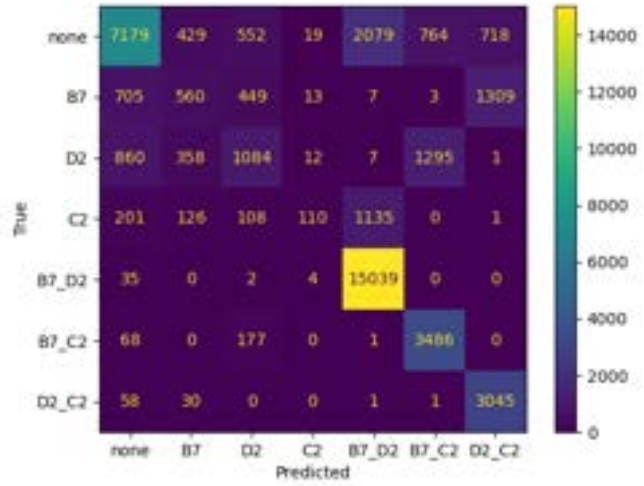


A.6.5. Gradient Boosting

F1 Scores

	precision	recall	f1-score	support
none	0,79	0,61	0,69	11740
B7	0,37	0,18	0,25	3046
D2	0,46	0,30	0,36	3617
C2	0,70	0,07	0,12	1681
B7_D2	0,82	1,00	0,90	15080
B7_C2	0,63	0,93	0,75	3732
D2_C2	0,60	0,97	0,74	3135

Confusion Matrix

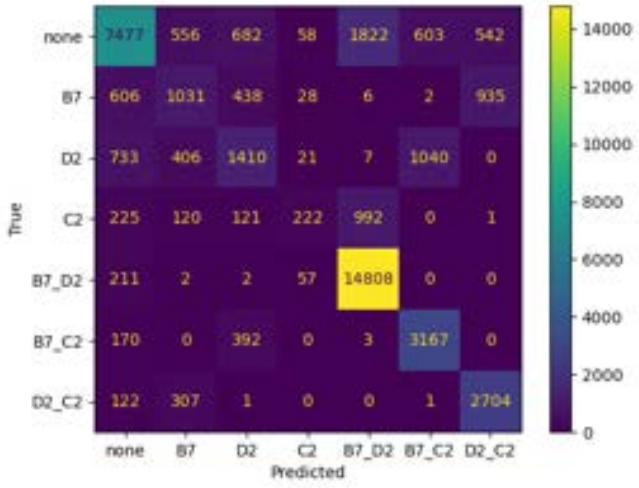


A.6.6. eXtreme Gradient Boosting

F1 Scores

	precision	recall	f1-score	support
none	0,78	0,64	0,70	11740
B7	0,43	0,34	0,38	3046
D2	0,46	0,39	0,42	3617
C2	0,58	0,13	0,21	1681
B7_D2	0,84	0,98	0,91	15080
B7_C2	0,66	0,85	0,74	3732
D2_C2	0,65	0,86	0,74	3135

Confusion Matrix

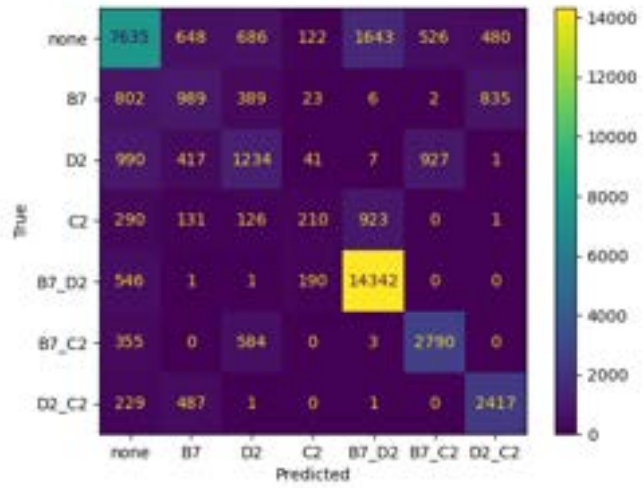


A.6.7. Bagging mit 10 Decision Trees

F1 Scores

	precision	recall	f1-score	support
none	0,70	0,65	0,68	11740
B7	0,37	0,32	0,35	3046
D2	0,41	0,34	0,37	3617
C2	0,36	0,12	0,19	1681
B7_D2	0,85	0,95	0,90	15080
B7_C2	0,66	0,75	0,70	3732
D2_C2	0,65	0,77	0,70	3135

Confusion Matrix

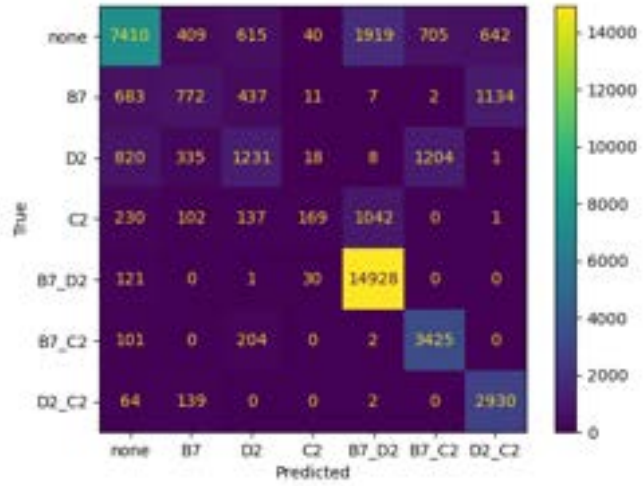


A.6.8. Bagging mit 10 Random Forests

F1 Scores

	precision	recall	f1-score	support
none	0.79	0.63	0.70	11740
B7	0.44	0.25	0.32	3046
D2	0.47	0.34	0.39	3617
C2	0.63	0.10	0.17	1681
B7_D2	0.83	0.99	0.91	15080
B7_C2	0.64	0.92	0.76	3732
D2_C2	0.62	0.93	0.75	3135

Confusion Matrix

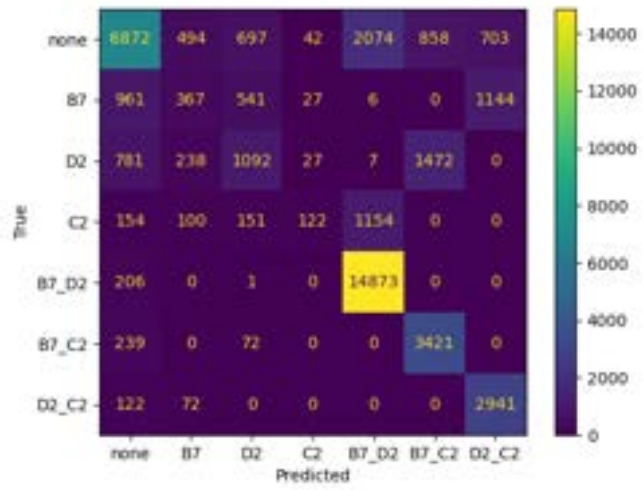


A.6.9. Stack mit eXtreme Gradient Boosting, Random Forest & Logistic Regression

F1 Scores

	precision	recall	f1-score	support
none	0.74	0.59	0.65	11740
B7	0.29	0.12	0.17	3046
D2	0.43	0.30	0.35	3617
C2	0.56	0.07	0.13	1681
B7_D2	0.82	0.99	0.90	15080
B7_C2	0.59	0.92	0.72	3732
D2_C2	0.61	0.94	0.74	3135

Confusion Matrix



A.6.10. Voting mit eXtreme Gradient Boosting, Random Forest & Logistic Regression

F1 Scores

	precision	recall	f1-score	support
none	0.80	0.63	0.70	11740
B7	0.42	0.27	0.33	3046
D2	0.48	0.36	0.41	3617
C2	0.72	0.10	0.17	1681
B7_D2	0.83	0.99	0.91	15080
B7_C2	0.64	0.91	0.75	3732
D2_C2	0.63	0.93	0.75	3135

Confusion Matrix