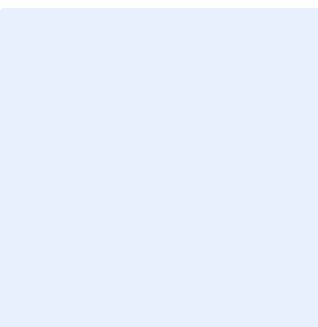


IP5 - NLP Schweizerdeutsch

Fabio Strappazon, Matthias Ernst



Betreuer: Wolfgang Weck, Daniel Kröni
Auftraggeber: Universität Zürich, Institut für Computerlinguistik
Fachhochschule Nordwestschweiz, Studiengang Informatik
Windisch, 07.01.2018

Diese Arbeit ist im Auftrag des Instituts für Computerlinguistik der Universität Zürich entstanden. Das Institut betreibt das Projekt «Citizen Linguistics: Locate that dialect!», in welchem Benutzer spielerisch schweizerdeutsche Texte klassifizieren, transkribieren und übersetzen.

Das Ziel dieser Arbeit ist die Verarbeitung von schweizerdeutschen Transkriptionen, um unterschiedliche Schreibweisen für dieselben Ausdrücke zu finden. Weiter soll es möglich sein, unvollständige Transkriptionen automatisch zu vervollständigen. Im Gebiet des Natural Language Processing ist dies unter dem Begriff der String-, Wort- oder Textalignierung bekannt. Tools und Algorithmen hierfür existieren zwar, müssen aber auf Verwendbarkeit in Bezug auf unsere Datencharakteristiken untersucht werden. Weiter sind Vor- und Nachbearbeitungsschritte (Ausfilterung unbrauchbarer Transkriptionen, qualitative Bewertung von Transkriptionen und Alignierungen, Kombination unabhängiger Alignierungen) nötig.

Wir stellen ein Verfahren vor, das aufgrund mehrerer Transkriptionen in Form eines CSV-Files eine Liste von Wortgruppen generiert. Wörter und Ausdrücke innerhalb derselben Wortgruppe sind dabei unterschiedliche Schreibweisen mit derselben Bedeutung.

Inhaltsverzeichnis

1	Einführung	4
1.1	Zusammenfassung	4
1.2	Projektkontext	4
1.3	Problemstellung	4
1.4	Vorgehen	4
2	Daten	7
2.1	Charakteristiken der Daten	7
3	Alignment von schweizerdeutschen Transkriptionen	9
3.1	Satzbewertung	9
3.2	Ausfilterung	11
3.3	Wortalignment	15
3.4	Filterung des Alignments	19
3.5	Fehlerkorrektur / Interpolation / Verbesserung	27
3.6	Bewertung des Ergebnisses	31
4	Code	35
4.1	Best_word	35
5	How tos	36
5.1	How to load and access Data	36
5.2	How to rate a transcription group	36
5.3	How to directly get the good transcriptions	36
5.4	How to align every sentence to the others	36
5.5	How to align a sentence to the others	36
5.6	How to score an alignment	37
5.7	How to align a list of groups of sentences	37
5.8	How to improve a sentence	37
5.8.1	Normal	38
5.8.2	Mit experimental bad_word_detection	38
5.9	How to print a graph	38
5.10	How to export as a graph	38
5.11	How to export as a list	39
5.12	How to import a graph	39
6	Schlussfolgerung	40
7	Abbildungsverzeichnis	42
8	Literaturverzeichnis	42

1 Einführung

1.1 Zusammenfassung

In dieser Arbeit werden Transkriptionen, die auf der Website www.dindialaekt.ch/ von Benutzern erfasst wurden, mithilfe von NLP-Tools analysiert und verarbeitet.

1.2 Projektkontext

Im Rahmen des vom Schweizerischen Nationalfonds geförderten Agora-Project «Citizen Linguistics: Locate that dialect!» ist die Website www.dindialaekt.ch/ entstanden.

Auf dieser Website können Benutzer unter Anderem folgenden Task erledigen:

- Transkription schweizerdeutschen Audiofiles
- Identifizieren von Dialekten mittels Lokalisierung auf einer Karte
- Übersetzung von schweizerdeutschen zu hochdeutschen Texten

Im Rahmen dieser Arbeit wollen wir aus den so gewonnenen Transkriptionen weitere Informationen gewinnen.

Zu jedem Audiofile gibt es mehrere (etwa 3 bis 7) zusammengehörende Transkriptionen.

Aus den so gewonnenen Transkriptionen sollen nun über Wortalignierung mögliche unterschiedliche Schreibweisen für schweizerdeutsche Ausdrücke gefunden werden.

Die Aufgabe wird dadurch erschwert, dass viele Transkriptionen unvollständig (Mit Platzhaltern in Form von *** oder ???) oder gar komplett falsch sind.

1.3 Problemstellung

- Unbrauchbare Transkriptionen ausfiltern
- Gute Transkriptionen erkennen
- *** Auslassungen interpolieren aus guten Transkriptionen
- Schreibvarianten von Wörtern und Ausdrücken erkennen über String-Alignierung

Daraus ergeben sich uns folgende zwei Fragestellungen:

Welche Tools funktionieren mit unseren Daten?

Natural Language Processing (NLP) ist ein etabliertes Gebiet. Für viele Problemstellungen gibt es Tools und Algorithmen. Viele dieser Tools haben jedoch gewisse Voraussetzungen. Beispielsweise benötigen sie einen Textkorpus einer gewissen Grösse oder sie benötigen Wörterbücher der zu untersuchenden Sprachen. Da es im Schweizerdeutschen keine genormte Rechtschreibung und daher auch keine Wörterbücher gibt, sind uns in der Wahl der Tools einige Grenzen gesetzt. Wir wollen also herausfinden, was für Tools grundsätzlich für unsere Art von Daten verwendbar sind.

Welche und wie viele Daten brauchen wir für gute Resultate?

Die Tools, die wir als «brauchbar» erkennen, werden jedoch auch Grenzen haben. Für die Betreiber der Website ist wichtig zu wissen, dass für eine Audiodatei ein Minimum an Transkriptionen vorhanden sein muss, um sinnvolle Analysen damit anzustellen. Konkret wollen wir also herausfinden, wie viele Transkriptionen zur selben Audiodatei vorhanden sein müssen, damit die von uns angewendeten Techniken funktionieren.

1.4 Vorgehen

Wir teilen den gesamten Prozess in mehrere Transformationen auf, die teilweise voneinander abhängig sind.

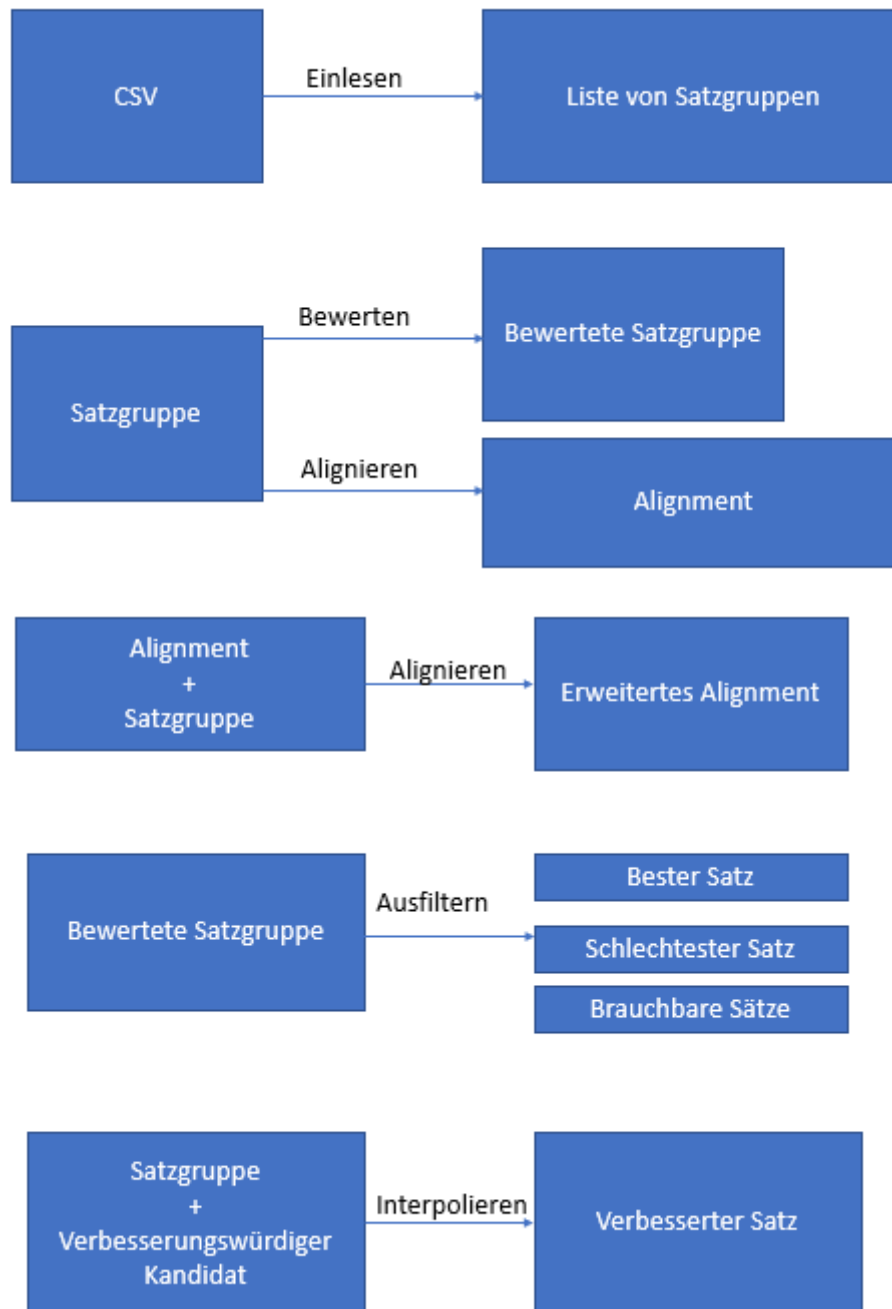


Abbildung 1: Alle Transformationen

Sätze einer Satzgruppe können nach Qualität bewertet werden. Dies macht es möglich, den besten und schlechtesten Satz zu identifizieren. Dieser Vorgang ist nötig, um für das Alignment unbrauchbare Sätze auszufiltern.

Der Alignierungsprozess generiert aus einer Satzgruppe eine Liste von Wort/Ausdrucksgruppen, die bedeutungsgleich sind. Der Alignierungsprozess kann ausserdem eine solche bestehende Liste von Wortgruppen mit den Wörtern einer weiteren Satzgruppe erweitert werden.

Gefilterte Daten sind für den Alignierungsprozess streng genommen nicht zwingend. Da dies jedoch das Ergebnis verbessert, wird es empfohlen. Der Alignierungsprozess kann ausserdem den Satzbewertungsprozess verwenden, um das Ergebnis zu verbessern.

Der Interpolationsprozess findet Auslassungen in Form von *** in einem Satz und ersetzt sie mithilfe der anderen Transkriptionen derselben Satzgruppe. Dieser Prozess basiert auf dem Alignierungsprozess. Auch hier empfiehlt es sich, mit gefilterten Satzgruppen zu agieren.

2 Daten

Tabellarische Daten. Folgende relevante Spalten:

TASK_ID: identifiziert zusammengehörige Transkriptionen

INFO: vom Benutzer erfasste schweizerdeutsch Transkription

Auszug aus CSV:

URL;VALID;TASK_ID;TASK_RUN_ID;USER_ID;INFO;REF

AUDIO;True;1829;24287;2887;Ma het dénn alz zäme glääseni Lüt was me hät chöne zämetriibe allz was Chopf u Loch hégi müessi ez höve héts aube ghéesse; https://dindialaekt.ch/data/transcribe/SDS_CD1_1_11_speaker1_1.mp3

AUDIO;True;1829;30820;3563;ma het denn allz zäme glääsenei lüt was mer het chöne zämetriibe alts was chopf u loch heegi müesi ez höve hets aube gheesse; https://dindialaekt.ch/data/transcribe/SDS_CD1_1_11_speaker1_1.mp3

AUDIO;True;1830;6158;3019;mer s alles us grückt d erste het dri gschnitte es paar schritt gange e zwetei made e dritt, e virtte bi al zäme drin gis sind; https://dindialaekt.ch/data/transcribe/SDS_CD1_1_11_speaker1_2.mp3

AUDIO;True;1830;10807;2218;när isch also alles usgrückt dr erschte het dri gschnitte es paar schritt gange zweite id maade e dritte e vierte bis au zäme drin gsi si; https://dindialaekt.ch/data/transcribe/SDS_CD1_1_11_speaker1_2.mp3

Unser Tool benötigt die Daten als UTF-8 formatiertes CSV mit ';' als Spaltendelimiter.

2.1 Charakteristiken der Daten

Stand 2017-07-08

2848 Transkriptionen zu 554 verschiedenen Audiodateien.

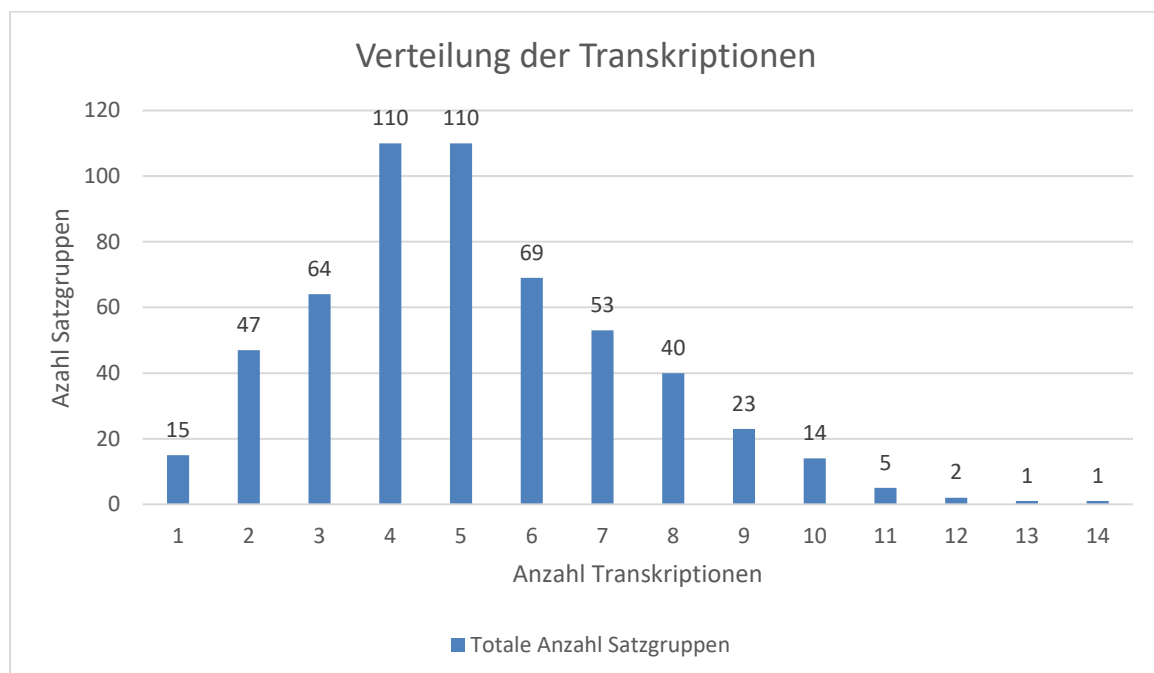


Abbildung 2: Verteilung der Transkriptionen

Pro Audiodatei sind zwischen 1 und 14 Transkriptionen vorhanden. Durchschnitt 5.14, Median 5.

So sehen zwei plausible Transkriptionen aus:

«Äs isch die Zyt wo d'Lüt afänd üsszie für d' Händöpfel understuen fascht alli hän da un dert än Händöpfelblätz. Ds Grosis ämel o.»

«Es isch die Ziit, wo d'Liit afend üsziehn fer d'Händöpfu underztüen Faschd alli hend *** ** en Häröpfublätz, z'Grosis emu o.»

Die Zweite beinhaltet eine Auslassung (**), da der Benutzer den Ausdruck «da un dert» nicht verstanden hat.

So sieht eine unbrauchbare Transkription aus: «dfbdfgh *** ** ** ** **»

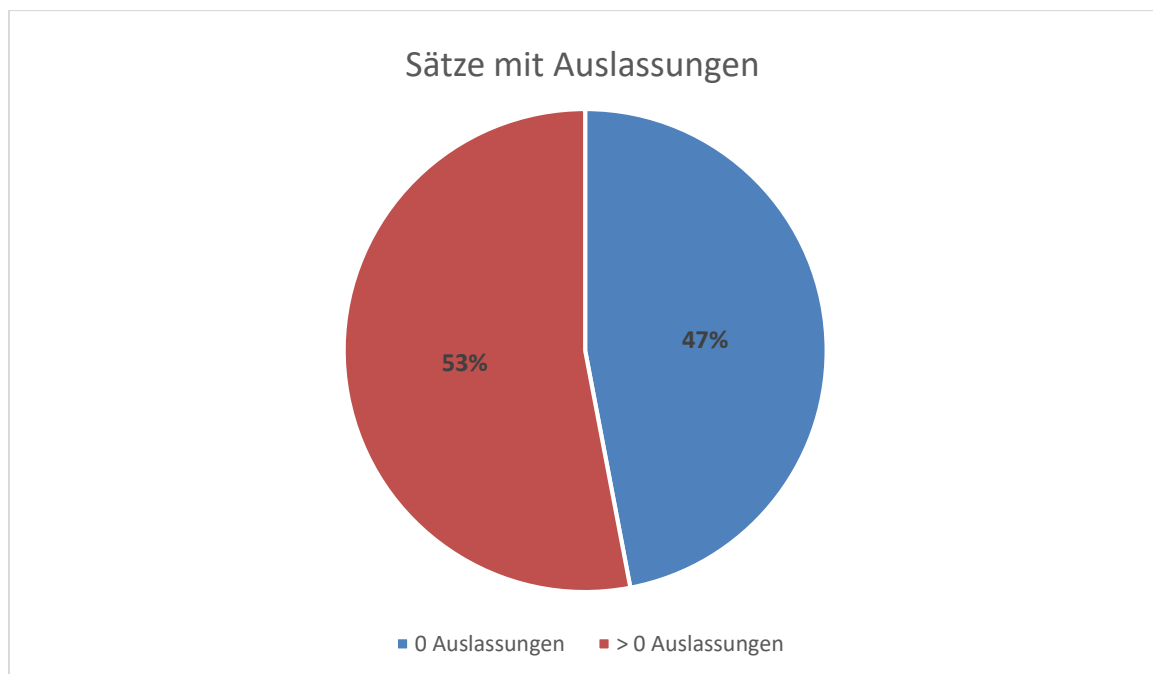


Abbildung 3: Prozentuale Verteilung der Auslassungen

47% unserer Daten weisen keine Auslassung auf. 53% mindestens 1.

3 Alignment von schweizerdeutschen Transkriptionen

3.1 Satzbewertung

Für jeden Satz aus einer Satzgruppe wollen wir wissen, wie gut er ist. «Gut» in diesem Zusammenhang bedeutet, wie akkurat die Transkription ist. Eine Bewertung pro Satz hilft uns in folgenden Szenarien:

- Ausfiltern von unbrauchbaren Sätzen:
Unbrauchbare Sätze verschlechtern das Alignmentsergebnis. Deshalb wollen wir sie so früh wie möglich erkennen und entfernen. Dieses Verfahren ist im Kapitel Ausfilterung beschrieben.
- Satzpaarbildung bei der Wortalignierung:
Die Information, welcher Satz potentiell der beste ist, hilft im eigentlichen Prozess der Wortalignierung. Dies ist im Kapitel Wortalignierung beschrieben.

Für die Task-ID 1851 (Audio: https://dindialaekt.ch/data/transcribe/SDS_CD1_1_3_speaker1_4.mp3) gibt es beispielsweise folgende vier Transkriptionen:

1. élk ék ékl ékl asda
2. Nacher heisis uusenageretschöplet ond *** zämebonge weder us weder of d'Reite uecheto od'Frocht heisi am Bode usegrächet ond nachere na Huufe geschtoosed.
3. nächäär heisis userenangere gschüttlet und Burdine zämme bunge wider us wider ufd Reiti ufetoo ud Frucht heisi am Bode usegrächet und nächäär ane Huufe gschtoosse
4. Nachhäär héysis usenangèrègsgschüttlet und Puurdine zämepunge wider us wider uft Reyti uechetoo u Pfrucht heysi am Bode usegrächet unt nachhäär ane Huuffe gschtoosse

Sofort ersichtlich ist, dass Transkription 1 nicht gut und in diesem Fall sogar völlig unbrauchbar ist.

Die andern drei Transkriptionen sind zwar brauchbar, weisen jedoch dennoch qualitative Unterschiede auf:

Transkription 2 ist unvollständig: *** ersetzt hier «Burdine» oder «Puurdine».

Transkription 4 verwendet die Buchstaben è und é, was für schweizerdeutsch eher ungewöhnlich ist.

Die automatische Bewertung der Sätze beruht auf der Prämisse, dass «gute» Transkriptionen sich ähneln und «schlechte» Transkriptionen sich von den anderen stark unterscheiden. In unserem Beispiel ist Transkription 1 zu allen andern unähnlich. Transkription 2 ist zu Transkription 3 und 4 etwa gleich ähnlich, tendenziell etwas ähnlicher zu Transkription 3, da diese keine è und é verwendet. Transkription 3 und 4 sind sich sehr ähnlich.

Wenn wir in der Lage sind, die Ähnlichkeit zwischen den Sätzen zu berechnen, können wir also relativ gut entscheiden, wie «gut» eine Transkription ist.

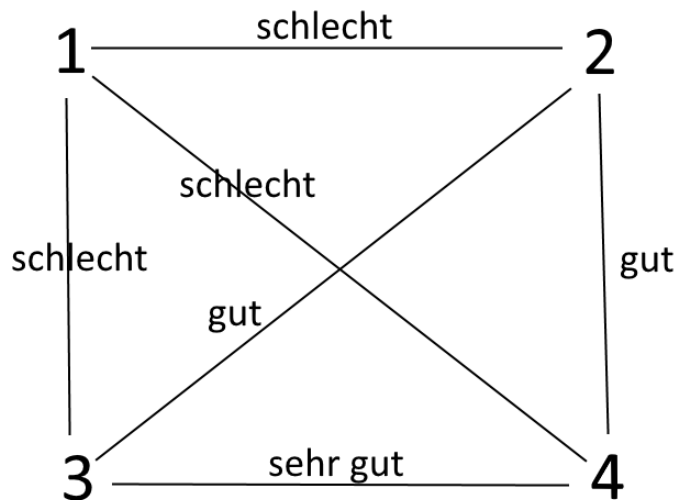


Abbildung 4: Relationen der Sätze bezüglich Ähnlichkeit

Pro Satz also:

Satz 1: schlecht, schlecht, schlecht.

Satz 2: schlecht, gut, gut.

Satz 3: schlecht, gut, sehr gut.

Satz 4: schlecht, gut, sehr gut.

Folglich brauchen wir ein Verfahren, die Ähnlichkeit zwischen Texten zu bestimmen:

So könnte beispielsweise die Levenshtein-Distanz (Levenshtein, 1966) verwendet werden:

Es wird die Levenshtein-Distanz zwischen jedem Satzpaar berechnet und dabei pro Satz die Distanz zu jedem anderen Satz aufsummiert.

Im Bereich der automatischen Bewertung von Maschinenübersetzungen wird eine ähnliche Idee genutzt: «the closer a machine translation is to a professional human translation, the better it is» (Papineni, et al., 2002), je näher eine Maschinenübersetzung an einer professionellen menschlichen Übersetzung ist, desto besser ist sie.

Papineni et al. stellen den Algorithmus «BLEU» vor, der eine Maschinenübersetzung mit einem Set von Referenzübersetzungen vergleicht. Je ähnlicher die Maschinenübersetzung zu den Referenzübersetzungen ist, desto besser wird er bewertet, wobei 0 die schlechteste und 1 die beste Bewertung ist. BLEU hat eine hohe Korrelation zu menschlichen Bewertungen und ist eine beliebte und einfach zu berechnende Metrik.

Wir können dieses Prinzip auf unser Problem anwenden, indem wir jeweils einen der Sätze als zu bewertende Maschinenübersetzung ansehen, und die jeweils anderen Sätze als die Referenzübersetzungen.

Mit diesem Verfahren erhalten wir für unsere vier Sätze folgende Bewertungen, wobei 1 sehr gut und 0 sehr schlecht ist:

Satz 1: 0.00033940394936699897

Satz 2: 0.5970591840896966

Satz 3: 0.8215011858242206

Satz 4: 0.7990890169242502

Satz 3 schneidet am besten ab, dicht gefolgt von Satz 4. Zu Satz 2 besteht ein gewisser Abstand und Satz 1 hat die mit Abstand am schlechteste Bewertung. Dies korreliert schon ziemlich gut mit unserer intuitiven Bewertung.

Für unsere Zwecke eignet sich BLEU also sehr gut. Es gibt einige weitere Metriken, die wir nicht in Betracht gezogen haben:

Word Error Rate, eine Metrik, die auf dem gleichen Prinzip wie die Levenshtein-Distanz basiert, jedoch auf Wortebene agiert, anstatt auf Phonem-Ebene.

METEOR (Banerjee, et al., 2005) und NIST (Doddington, 2002) bauen auf BLEU auf und versprechen einige Kleinigkeiten zu verbessern.

LEPOR (Han, et al., 2012) und die verbesserten Iterationen hLEPOR und nLEPOR, die versprechen einige der Probleme mit BLEU und den darauf aufbauenden Metriken zu lösen.

Sowie eine Vielzahl weiterer Metriken und Verfahren. Da BLEU der de facto Standard ist und wir mit BLEU ausreichend gute Ergebnisse erhalten und nicht erwarten, mit einem anderen Verfahren fundamental bessere Ergebnisse zu erhalten, werden wir diese in dieser Arbeit ignorieren.

All diese Metriken wurden für die Dokumentenebene konzipiert. Auf der Satzebene, wie wir sie hier verwenden, liefern sie etwas schlechtere Ergebnisse. Für BLEU wurden deshalb einige sogenannte Smoothing Functions entwickelt. Smoothing Function 7 liefert die besten Ergebnisse auf der Satzebene (Chen, et al., 2014). wir werden diese deshalb hier verwenden.

Satz	Default BLEU	BLEU mit Smoothing Function 7
1	0.00033940394936699897	0.0002645904060323999
2	0.5970591840896966	0.6781557399224509
3	0.8215011858242206	0.9185006575847926
4	0.7990890169242502	0.8947321304759088

Mit der Smoothing Function ist also der Unterschied zwischen guten und schlechten Sätzen noch etwas grösser geworden. Erwähnenswert ist, dass mit Smoothing Function die Bewertung eines Satzes besser als 1 sein kann.

3.2 Ausfilterung

Als Bereinigungsprozess wollen wir alle Sätze ausfiltern, die für uns keinen sinnvollen Informationsgehalt haben und potentiell das Ergebnis verschlechtern. Da wir bereits ein Verfahren zur qualitativen Satzbewertung haben, können wir diese Ausfilterung auf den Satzbewertungen basieren.

Im obigen Beispiel zur Task-ID 1851 ist Satz 1 mit der Bewertung von 0.0003 der einzige, den wir ausfiltern wollen. Satz 2 mit einer Bewertung von 0.678 ist für uns wertvoll und soll nicht ausgefiltert werden.

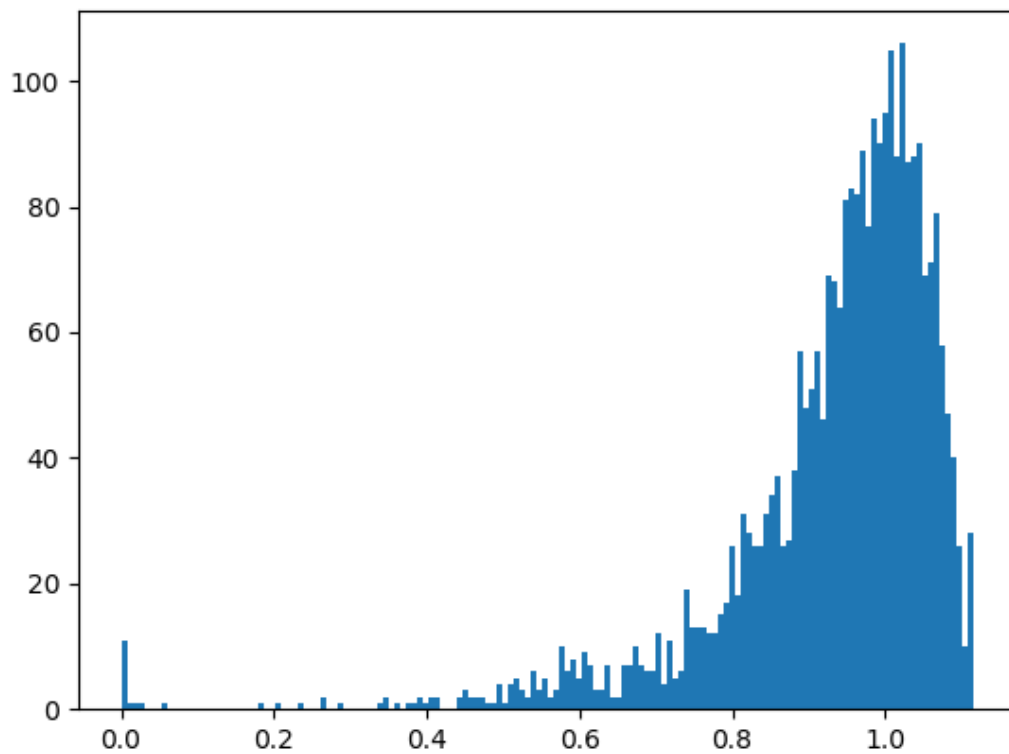


Abbildung 5: Histogramm der Bewertungen aller Sätze

Aus obenstehender Abbildung wird die Verteilung der Bewertungen ersichtlich. Die meisten Bewertungen liegen zwischen 0.8 und 1.1. Der Durchschnitt liegt bei 0.9332, der Median bei 0.9698. Nur 51 Sätze haben eine Bewertung von unter 0.5 und 15 davon liegen sogar unter 0.1.

Bewertung	Satz	Referenzsatz (mit besserer Bewertung)
0.00026	élkékéklékl asda	nächäär heisis userenangere gschüttlet und Burdine zämme bunge wider us wider ufd Reiti ufetoo ud Frucht heisi am Bode usegrächet und nächäär ane Huufe gschoosse
0.00896	thurgauer thurgauer thurgau	dem isch scho so, das mis s Thema sehr interessiert und au imer interessiert hett. aso, ich machs nüd wegem Dschob noch i wäss no gär nid won ig will here nochane. äm- also mi interessierets wüekli und es isch kuul.
0.0551	Altwiibervolk *** *** Das fuärät gad achli *** ***	Au z Wybervolch cho do nid e so lang schloofe die müend de des ds Belätebrösöm zum z Früstück rüschte und das fueret denn e chli und de mögeds de die eiwer Lüüt ender öppe bis zum Zmit-tag verliid
0.2042	Kleider hät er eifsch nebed s'bett gleit *** *** ***	Klaider hät er eifach näb z Bett gworffe so müad ischer gsii und sis Bett isch es bizli ghurz und sini Füass luegend une use
0.3914	nacher wo ich bueb gsi bin sägese gmäht worde *** so het mer e sbögli dra gmacht, an d sägese und hets schön aaböglet *** ***	daher won ich Bueb gsi bi isch so mit de Sägesse gmeiht worde, neh d Fruchte no schön gschtange isch so hett mer s Beegli dra gmacht, a d Sägesse und hett schön abeeglet und sind d'Schnitter no cho und hei das eweg gno und scheen of d Zettle gleit
0.5744	Last tragen Jeder trägt eine Last Jeder trägt einen Teil Wäre es etwas zum Verkaufen, so würde es jeder anbieten	Büürdeli träge ei jede trät es Bürdeli ei jede trät e Teil wärs eppis zum Verchräämere, es häts e jede feil.
0.6588	Ein jeder wollte mehr Land besitzen, .. und sie wurden sich nicht enig. Zuletzt machten sie ab, sie wollten an einem bestimmten Tag am frühen Morgen... so wie der Hahn gekräht habe, von jedem Hauptort den Landläufer (Landboten) aussenden.. Und dort, wo die beiden aufeinander treffen, .. sollte für alle Zeit die Grenze sein.	e jede Teil het welle mi Welt ha und si sind eifach nid z'einte worde do heit's jetzt letscht abgmached si welled amene bschtimmte Tag am morged früe so wi dr Guli kräht heig vo jedem Hauptort der Landläufer usschicke und wo dä beed zämechömet söll für all ebig Ziite d'Gränze si
0.7991	de Bäärgamme mües dä alläg im Herbst emene *** übergä. dä miends die Gäischtleche Bihäärdig und de Tokter is Aesse iilade und dem Eerebrediger de Chääs psoorge	de Bäärgamme müesti allig im Herbst emene Wiil d'Chilbi vergä, de miends di Geischtleche, di *** und dr Dokter is Ässe iilade und dem Ehreprediger dr Chääs bsorge

Aus dieser Tabelle ist ersichtlich, wie schlechte Sätze bewertet werden. Zwei der hier präsentierte Sätze wurden dabei fälschlicherweise auf hochdeutsch transkribiert, was für uns nicht wünschenswert ist. Es sollte klar werden, dass es keine klare Grenze zwischen brauchbaren und unbrauchbaren Sätzen gibt.

Der simpelste Weg, Sätze auszufiltern, ist einen Minimalwert einzuführen. Je tiefer dieser ist, desto mehr unbrauchbare Sätze werden nicht ausgefiltert. Je höher der Minimalwert, desto mehr brauchbare Sätze gehen verloren. Unter 0.3 sind kaum noch brauchbare Sätze zu finden, über 0.7 sind praktisch alle Sätze brauchbar. Als Default schlagen wir einen Wert von 0.5 vor.

Etwas schwieriger wird es in Satzgruppen, in denen es nur 2 Transkriptionen gibt.

Bewertung	Satz A	Satz B
0.35	Also, Sie fragen wann ich geboren bin ? Ich bin am 26. Januar 1912 geboren.	jaa *****wän das ich geboore bi? eem ...ich bin am sächsezwänzigschte Jäner nünzehundertzwölf geboore
0.53	aber es Bögli für e Chäs üse z'näh, do het Hans Fähri gseit, das sig z'churzes do manglet mer denn o no z'gseh für nes anders - jo das wei mer denn luege, ja	oderes Bögli für è Chääs üüseznäh und det hät hans fääre gseit das sig z churzes damol luegtemer de scho z gsee fürnes anders
0.65	jetze chumensch güetlech witer und lüegen über de abrosch uf ds Dach u gseh - da isch ds zennusch hüt üsgspreiteti gsi	Jetzt bechums güet för hütte, und lüegend über de äberarsch ufs Dach und gseh, da isch s'Tenusch *** üsbreiteti gsi
0.80	Z Hasli hett der Rott-Bueb gläbt Sy Übernae isch offebar derthär cho wiu är als Bueb, wie's früecher Bruuch gsi isch lang e Rock agha het Als Schwinger isch är gförderet worde Und är heig Ross-lise vo blosser Hand abenand grisse	zHasli hät de Rockbueb gläbt si Uebername isch offebar deet häärchoo wüu är as Bueb, wis früner Bruuch gsi isch lang ne Rock aagha héd Us Schwinger ischèr gförchtet worde und er häig Rossiise vo bloosser hand abenang-grisse

Satzgruppen mit nur 2 Transkriptionen scheinen tendenziell schlechter bewertet zu werden. Diese Sätze haben eine durchschnittliche Bewertung von lediglich 0.676.

Auch hier gibt es weitere Ansätze, Outlier auszufiltern, die wir nicht weiterverfolgen konnten:

In der Statistik werden Outlier mittels Durchschnitt und Standardabweichung erkannt. Dieser Ansatz ist für unsere Daten jedoch weniger geeignet, da statistische Methoden oft von normalverteilten Daten und grösseren Datenmengen ausgehen.

Pro Gruppe könnten die Sätze in unterschiedliche Qualitätskategorien klassifiziert werden. Danach könnten die schlechtesten Kategorien verworfen werden. Dies liesse sich als eine Art eindimensionaler Clustering Algorithmus implementieren (Jenks, 1967).

Man könnte versuchen, die durchschnittliche Bewertung innerhalb einer Satzgruppe zu verbessern, indem iterativ der Satz mit der schlechtesten Bewertung ausgeschlossen und der Durchschnitt neu berechnet wird. Sobald sich der Durchschnitt nicht mehr gross ändert, wurden alle schlechten Sätze ausgefiltert. So verlagert sich das Problem zu der Frage, was eine grosse Veränderung des Durchschnitts ist.

Für die Zwecke dieser Arbeit sind wir nicht davon überzeugt, dass einer dieser alternativen Ansätze uns merklich bessere Resultate liefern wird. Weiter macht es unser Ansatz eines einfachen Minimalwerts für einen Benutzer sehr einfach, diesen mit vorhersagbaren Resultaten anzupassen. So können wir dem Benutzer die Möglichkeit geben, den Tradeoff zwischen fälschlicherweise gefilterten und fälschlicherweise ungefilterten Sätzen selbst zu bestimmen.

3.3 Wortalignierung

Für die Task-ID 2020 (Audio: https://dindialaekt.ch/data/transcribe/SDS_CD3_2_17_speaker1_2.mp3) gibt es folgende Transkriptionen (für dieses Beispiel gekürzt):

- Im Winter momè jo nömme graase dä goopme uf Püni ue
- Im Winter mue me ja nüme grase, de goht me uf d Bühni ue
- Im winter mome jo nümme grase, denn gohtmä uft Bühni ue
- im Winter mueme ja nümme grase, de got me ga d Bühni ue

Ziel der Wortalignierung ist es, aus den vorhandenen Transkriptionen alle Schreibvarianten für die jeweils gleichbedeutenden Ausdrücke zu finden. Unter einem Ausdruck verstehen wir das kleinste Wort-N-Gramm, das direkt mit einem einzelnen Wort aligniert werden kann. Gross- Kleinschreibung und Satzzeichen ignorieren wir hierbei. So gehören 'graase' und 'grase' zusammen, sowie 'jo' und 'ja'. Für die Wörter 'im' und 'winter' gibt es keine alternativen Schreibvarianten.

Gewisse Ausdrücke werden von manchen Leuten zusammen oder getrennt geschrieben. Hier gehören 'momè', 'mue me', 'mome' und 'mueme' zusammen. Interessant ist hier, dass das Bigramm 'mue me' ein einzelner Ausdruck ist.

Das Wort 'ga' im vierten Satz ist nicht gleichbedeutend mit den Ausdrücken 'uf', 'uf d' und 'uft'. Wie wir dieses ausfiltern, wird im Kapitel Filterung des Alignments behandelt.

Konkret wollen wir für dieses Beispiel Output in folgendem Format:

```
[
  ["im"],
  ["winter"],
  ["momè", "mue me", "mome", "mueme"],
  ["jo", "ja"],
  ["nömme", "nüme", "nümme"],
  ["graase", "grase"],
  ["dä", "de", "denn"],
  ["goopme", "goht me", "gohtmä", "got me"],
  ["uf", "uf d", "uft"],
  ["püni", "bühni"],
  ["ue"]
]
```

Im Bereich des Natural Language Processing bezeichnet der Begriff Bitext-Wortalignierung (oder einfach Wortalignierung) den Vorgang, einen bipartiten Graph zwischen zwei Bitexten aufzubauen, bei dem die Knoten die Wörter und die Kanten die Relation "ist eine Übersetzung von" darstellen.

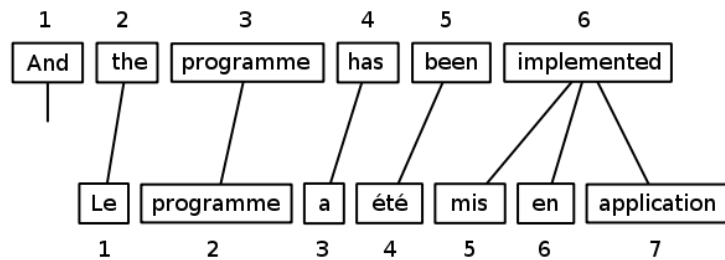


Abbildung 6: Wortalignierung als bipartiter Graph, von Redienss (CC BY-SA 3.0)

Obwohl wir es mit einem Wortalignierungsproblem zu tun haben, verwenden wir Satzaligner und keine Wortaligner. Satzaligner werden klassischerweise verwendet, um in parallelen Korpora die Relation zwischen Sätzen herauszufinden. Bei Übersetzungen kann es oft passieren, dass ein einzelner Satz in zwei (oder mehrere) Sätze übersetzt wird oder mehrere Sätze zu einem zusammengefasst werden. Der entscheidende Unterschied ist, dass Satzaligner grundsätzlich davon ausgehen, dass die Reihenfolge der Sätze in beiden Texten stabil bleibt, wohingegen die Wortstellung innerhalb eines Satzes in zwei unterschiedliche Sprachen sehr unterschiedlich sein kann. Da in unserem Fall die zu alignierenden Sätze nicht Übersetzungen voneinander sind, sondern Transkriptionen derselben Audiodatei, können wir davon ausgehen, dass die Wortreihenfolge in allen Sätzen identisch ist. Normale Wortaligner sind also grundsätzlich zu flexibel für unsere Transkriptionsdaten.

Unser Problem gleicht also eher einem Satzalignierungsproblem. Um Satzaligner für Wortalignierungsprobleme zu verwenden, können Sätze als Abschnitte, Wörter als Sätze und Buchstaben als Wörter angesehen werden. Konkret heisst das, dass wir die Wörter in einem Satz mit Zeilenumbrüchen ('\n') und die Buchstaben in einem Wort mit Leerschlägen trennen müssen, bevor wir einen Satzaligner verwenden.

Satzaligner benutzen eines oder mehrere folgender Informationen: Wörterbuch, Textlänge, Textähnlichkeit. Da es in unserem Fall keine Wörterbücher gibt, können alle Tools, die ein Wörterbuch benötigen, ausgeschlossen werden. In http://lium3.univ-lemans.fr/mtmarathon2010/ProjectFinalPresentation/SentenceAlignment/sentence_alignment.pdf werden einige Satzaligner vorgestellt und verglichen (Abdul Rauf, et al., 1012). Die Resultate hierbei sind generell sehr ähnlich, weshalb wir keine fundamental unterschiedlichen Ergebnisse erwarten. Wir haben zwei der vorgestellten Aligner (Bleualign und Hualign) ausprobiert.

Bleualign (Sennrich, et al., 2010), (Sennrich, et al., 2011) ist ein Satzalignierungstool, das versucht, den BLEU Score zwischen den alignierten Sätzen zu maximieren.

Hualign (Varga, et al., 2007) (wenn ohne Wörterbuch verwendet) aligniert in einer ersten Iteration anhand der Satzlänge. Aus der so gewonnenen Information wird ein Wörterbuch generiert, das in einer zweiten Iteration zur Verbesserung der Alignierung verwendet wird.

Fastalign (Dyer, et al., 2013) ist ein Wortalignierer. Er bietet jedoch eine Option an, die diagonale Alignierungen bevorzugt. Da sich jedoch das Input- und Outputformat stark von demjenigen von Bleualign und Hualign unterscheidet und wir davon ausgehen müssen, dass das Resultat nicht fundamental besser ist, haben wir hier Fastalign nicht in Betracht gezogen.

Vorgehen:

Der Aligner bildet für ein Satzpaar die jeweiligen Ausdruckspaare. Diese können wir wie oben beschrieben als Graph interpretieren. So ist es möglich, Ausdruckspaare, die in unterschiedlichen Alignierungen entstanden sind, in einen einzelnen Graphen zu kombinieren.

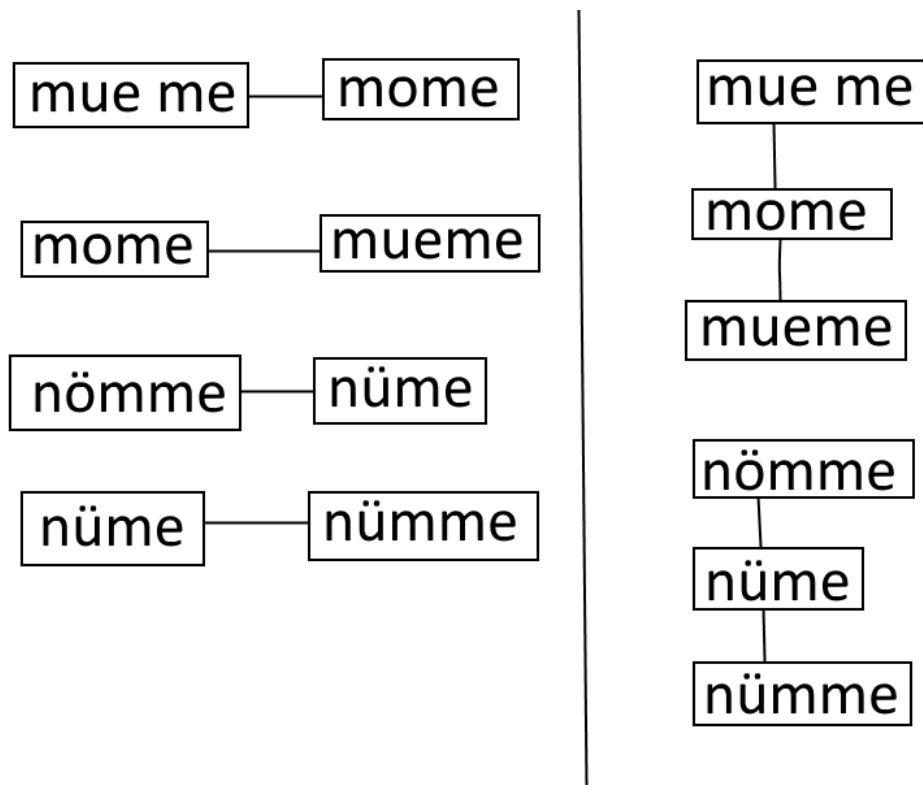


Abbildung 7: Kombination von Ausdruckspaaren in einen einzelnen Graphen

In einem so entstandenen Graphen können wir die Komponenten (auch Zusammenhangskomponenten genannt: ein maximaler zusammenhängender Teilgraph) als Gruppen gleichbedeutender Ausdrücke interpretieren.

Da wir meist mehr als zwei zusammengehörige Sätze haben, müssen wir eine Strategie für die Paarbildung der Sätze finden.

1 zu N: Ein Satz wird als Hauptsatz ausgewählt und dann mit allen anderen Sätzen gepaart. Sinnvollerweise ist dieser Hauptsatz nach seiner Bewertung durch BLEU zu wählen. Bei n Sätzen in einer Satzgruppe erhalten wir so $n-1$ Paarungen.

N zu N: Jeder Satz wird mit jedem gepaart. Nach dem Handshake Problem gibt es bei der grössten Satzgruppe mit 14 Sätzen $14 * \frac{14-1}{2} = 91$ und bei der durchschnittlichen Satzgruppe mit 5 Sätzen $5 * \frac{5-1}{2} = 10$ Paarungen.

Da wir uns schlussendlich nur für die Komponenten interessieren, sollte die gewählte Strategie keine Rolle spielen. Idealerweise resultieren beide in den gleichen Komponenten und unterscheiden sich nur in den Kanten innerhalb einer Komponente. Da aus dem Alignment jedoch auch fehlerhafte Ausdruckspaare entspringen können, sind wir gezwungen, manche davon auszufiltern. Dieser Vorgang wird im nächsten Kapitel beschrieben. Dies hat auf die beiden Strategien unterschiedliche Auswirkungen.

Haben wir beispielsweise drei Ausdrücke A, B und C, welche alle gleichbedeutend sind und vom Aligner auch so erkannt werden, Der Filteralgorithmus erklärt jedoch die Beziehungen zwischen A und B sowie B und C fälschlicherweise für ungültig, so taucht keiner der drei Ausdrücke im Graphen auf, wenn wir das 1 zu N Verfahren mit dem Hauptsatz wählen, der B beinhaltet. Das N zu N Verfahren würde in diesem Fall jedoch noch ein weiteres

Ausdruckspar, nämlich A zu C, generieren. Somit ist das N zu N verfahren resistenter gegenüber zu strengen Filterkriterien. Jedoch unterliegt das N zu N Verfahren dem 1 zu N Verfahren bei zu laschen Filterkriterien sowie grossen Datenmengen.

	Hunalign	Bleualign
1 zu N	[["im"], ["winter"], ["momè", "mome", "me"], ["jo", "ja"], ["nüme", "nümm", "nömme"], ["grase", "grase"], ["denn", "de", "dä"], ["goht me", "goopme", "gohtmä"], ["uft", "uf"], ["püni", "bühni"], ["ue"], ["mue", "mueme"], ["goht", "got"], ["d"]]	[["im"], ["winter"], ["momè", "mue"], ["jo", "ja"], ["nümm", "nömme", "nüme"], ["grase", "grase"], ["denn", "dä", "de"], ["goopme", "goht me"], ["uft", "uf"], ["bühni", "püni"], ["ue"], ["me", "mome"], ["gohtmä", "got", "goht"], ["d"], ["mueme", "mue me"]]
N zu N	[["im"], ["winter"], ["me", "mome", "mueme", "mue", "momè"], ["jo", "ja"], ["nömme", "nümm", "nüme"], ["grase", "grase"], ["de", "denn", "dä"], ["goopme", "got me", "goht me", "gohtmä"], ["uf", "uft"], ["püni", "bühni"], ["ue"], ["goht", "got"], ["d"], ["ga"]]	[["im"], ["winter"], ["mue", "mome", "mueme", "me", "mue me", "momè"], ["jo", "ja"], ["nömme", "nümm", "nüme"], ["grase", "grase"], ["dä", "denn", "de"], ["got me", "goht me", "got", "goht", "goopme", "gohtmä"], ["uft", "uf"], ["püni", "bühni"], ["ue"], ["d"], ["ga"]]

In der obigen Tabelle sieht man das Ergebnis beider Aligner mit beiden Strategien (Als Filtervalue wurde hier 0.6 gewählt). Auffällig ist, dass keine der Kombinationen mit unserer Musterlösung identisch ist. Trotzdem ist es schwierig, eine klare Aussage zu machen, welches der Resultate am besten ist.

3.4 Filterung des Alignments

Das Ergebnis dieses Prozesses ist eine Liste von Gruppen von Ausdrücken. Die Ausdrücke einer Gruppe sollten hierbei gleichbedeutend sein. Da wir wissen, dass die Ausdrücke das Resultat von Transkriptionen derselben Audiodatei sind, sollten die Ausdrücke bis auf die Schreibweise identisch sein. Fehlerhafte Transkriptionen, schlechte Alignment-Ergebnisse und fehlerhaftes Zusammenbauen der Ausdrucksgruppen verschlechtern jedoch das Ergebnis. Deshalb müssen wir einen Weg finden, das generierte Alignment qualitativ zu bewerten, um dann ungenügende Ergebnisse auszufiltern.

Für dies haben wir verschiedene Algorithmen in Betracht gezogen.

Algorithmen

Bleu-Score (Chen, et al., 2014)

BLEU ist für die Bewertung längerer Texte gedacht und liefert beim Vergleich einzelner Wörtern kein sinnvolles Ergebnis. Siehe auch 3.1 Satzbewertung

Levenshtein-Distanz (Nádovrník, 2012) (Levenshtein, 1966)

Auch Editierdistanz genannt, berechnet die Anzahl an Operationen (Einfügen, Löschen, Ersetzen von Buchstaben) die es braucht, um von Wort A auf Wort B zu kommen.

Beispiel: Geld und Gäld haben eine Distanz von 1.

Gewichtete Levenshtein-Distanz (Su, 2017)

Besser geeignete Version von der Levenshtein Distanz für das Vergleichen von Phonetisch gleich Klingenden Wörtern. So kann man hier mit der Gewichtung der Operationen auf den Buchstaben eine fairere Editierdistanz berechnen. Beispiel: Gäld und Geld klingen im Schweizerdeutschen sehr ähnlich, würden aber mit der normalen Version eine Distanz von 1 aufweisen. Jedoch hier bei der Gewichtung wird die Substitution von gewissen Zeichenpaaren billiger gemacht, wenn sie akustisch ähnlich sind.

<https://github.com/nadvornix/python-fizzle>

<https://pypi.python.org/pypi/weighted-levenshtein/0.1>

Damerau-Levenshtein-Distanz (Fairchild, 2017)

Die Operationen der Levenshtein Distanz wird um Transponieren erweitert, diese ermöglicht das Vertauschen von zwei nebeneinanderstehenden Buchstaben.

<https://github.com/gfairchild/pyxDamerauLevenshtein>

Soundex (Russell, 1918)

Errechnet die Lautähnlichkeit von zwei Strings. Gedacht um englische Namen zu vergleichen. Beispiel: Smith und Smiff.

Schlechte Unterstützung für Deutsch, keine für Schweizerdeutsch.

<https://pypi.python.org/pypi/soundex>

Metaphone (Beider, et al., 2010)

Generiert Keys basierend auf Lautähnlichkeit eines Wortes. Im Englischen kann man so zwei Wörter vergleichen. Beispiel: Programming und Programmer resultieren beide in PRKRM.

Double Metaphone (Somerville, 2010)

Generiert Keys auf Lautähnlichkeit eines Wortes wie Metaphone, unterstützt jedoch auch Slawisch, Germanisch, Keltisch, Griechisch, Französisch, Italienisch, Spanisch, Chinesisch und andere Herkünfte. Generiert einen Primary und Secondary Key, wir beschränken uns auf die Verwendung des Primary Keys, da es sich bei unseren Wörtern generell nicht um Namen handelt.

<https://github.com/dracos/double-metaphone>

Entscheid über den Algorithmus

Sondex und Metaphone unterstützen die deutsche Sprache zu wenig gut. BLEU und vergleichbare Metriken sind prinzipiell für ganze Dokumente konzipiert und können nur dank Smoothing Functions auf Satzebene angewendet werden.

Wir verwenden als Grundlage die gewichtete Damerau-Levenshtein-Distanz.

Diese normalisieren wir auf die Länge der jeweiligen Wörter. Somit erhalten wir einen Wert zwischen 0 und 1, der besagt, wie gross der Unterschied der beiden Wörter ist. Haben zwei Wörter eine Distanz von 0 sind sie identisch, haben sie eine Distanz von 1 sind sie komplett unterschiedlich (im Unterschied zu BLEU Score, welcher die *Ähnlichkeit* ausdrückt). Um aufgrund dieser Distanz einen binären Entscheid zwischen Filtern oder nicht Filter treffen zu können, müssen wir also wieder einen Grenzwert definieren.

Wir haben zusätzliche Gewichtungen für Buchstabenersetzungen definiert (Standard ist 1): 'y' wird im Schweizerdeutschen oft äquivalent mit 'i' verwendet. Die Vokale 'e' und 'ä' sowie einige Varianten hiervon mit anderen diakritischen Zeichen werden sehr ähnlich betont.

Ebenso die zusammengehörigen Plosivlaute 'g' und 'k', 'd' und 't' und 'b' und 'p'.

Dies ist die vollständige Gewichtungstabelle:

Buchstabe A	Buchstabe B	Gewichtung
i	y	0.1
e	ä	0.2
e	é	0.2
e	è	0.2
e	a	0.5
ä	é	0.2
ä	é	0.2
ä	ë	0.2
e	ë	0.2
a	ë	0.2
é	ë	0.2
b	p	0.8
n	m	0.8
d	t	0.8
g	k	0.8

Dies gilt in beide Richtungen, von A -> B und von B -> A.

Als Unterstützung verwenden wir aber noch Double-Metaphone. Mit Double-Metaphone prüfen wir die phonetische Ähnlichkeit der Wörter. So sind «dängelet» und «tengälät» für Double-Metaphone phonetisch sehr ähnlich.

Wir kombinieren diese beiden Algorithmen und kommen so auf bessere Ergebnisse.

Unsere Algorithmen filtern wie folgt die Alignments aus:

Metaphone Bewertung	Levenshtein Bewertung	Resultat
OK	OK	NICHT AUSGEFILTERT
OK	FAILED	NICHT AUSGEFILTERT
FAILED	OK	NICHT AUSGEFILTERT
FAILED	FAILED	AUSGEFILTERT

Hier ein paar Beispiele:

Params: Levenshtein Filter (0.4)

Wortpaar mit Levenshteinvalue und Metaphone Bewertung	Wird ausgefiltert
('sägäslì', 'sägessli', 0.275, False)	Nein, da Levenshtein OK
('de', 'die', 0.3333333333333333, True)	Nein, da beide OK
('wèrdèt', 'wäärded', 0.42857142857142855, True)	Nein, da Metaphone OK
('ëm', 'am', 0.1, True)	Nein, da beide OK
('?ù', 'scho', 1.0, False)	Ja, da beide FAILED
('ùn', 'und', 0.6666666666666666, False)	Ja, da beide FAILED

Code dahinter

```
1: lv = normalized_dl_distance(key, value) > alignment_filter_value
2: meta = doublemetaphone(key) == doublemetaphone(value)
3: if lv and not meta:
4:     continue
```

Zeile 1:

Wir berechnen die normalisierte Levenshteindistanz schauen dann, ob sie über der übergebenen Filtervalue liegt.

Zeile 2:

Wir berechnen den Double-Metaphone Key des Wortpaares und vergleichen ob er identisch ist.

Zeile 3/4:

Wenn sich Levenshtein und Double-Metaphone einig sind, sprich sie beide das Wortpaar als schlecht bewerten, wird keine Verbindung der Wörter im Graphen erstellt.

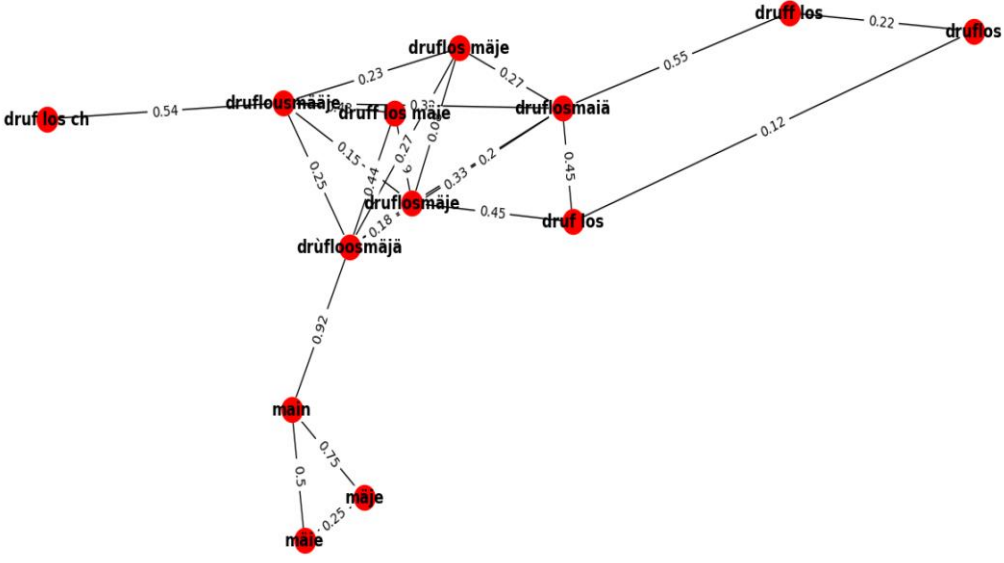
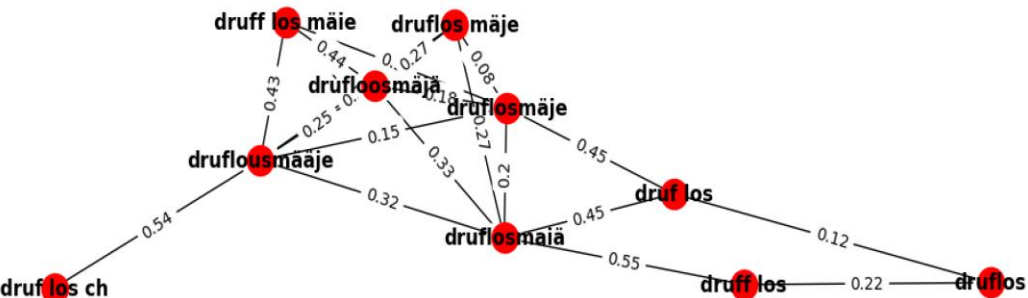
Entscheid über Parameter des Filters

Um eine Filtervalue festzulegen, haben wir für verschiedene Abstufungen der Filtervalue den Graphen von Task-ID 2048 generiert und davon den Teilgraphen zu «drauflos mähen» betrachtet.

Für die Task-ID 2048 (Audio: https://dindialaekt.ch/data/transcribe/SDS_CD4_1_14_speaker1_4.mp3) gibt es folgende Transkriptionen:

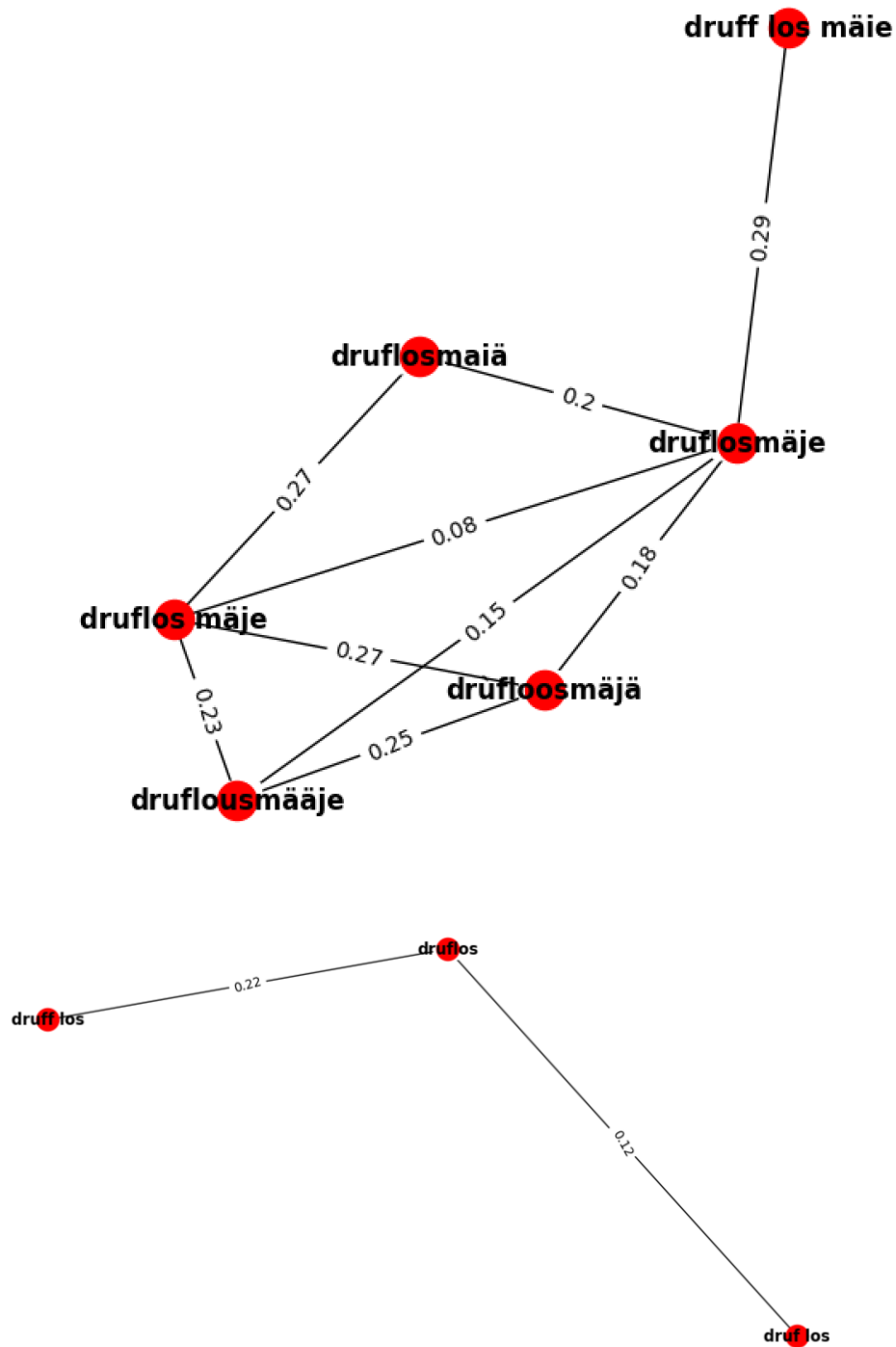
- D sägässli diä wärdid am oobet voräne schon tänglet ins grüschtet Dass me denn moore dess a morgete cho druflosmäje
- *** Ds Zägäslì diä wärdid am Obed vor alle scho dängläd u grüschtet dasme denn am More desse Morgete chan druflosmäje.
- Diä sägässli werded am obed scho grüschtet und dängläd dass me de more desse murgete cho druf los ch main
- Diä sage wered em obed scho grüschtet und tängelet dass me de more desse murgete cho ***
- d sägässli die wäred dänn am oobed vorane scho dängälät u grüschtet dassme dänn am moore disse murgete cha druff los mäie
- T Sägäslì de wèrdèt dä ëm Aabëd forhanä ?ù tängëlèt ùn ggry?tët. das mä dän ëm moorä det ëm Morgëd dä cha drufloosmäjä.
- und Sägesse werded dänn scho am Obed voranne dängelet und grüschtet das mer se more des morge de cha druflos mäje
- d Sägessli die wäarded am Obed vorane scho tängelet und grüschtet das me dennè moore dissemorgete cha druflosmäje

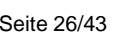
—
Folgend die Graphen und ihre Filtervalue. Ein Teil-Graph stellt eine Gruppe von alignierten Wörtern dar.

Filter- value für Le- vensht ein	Graphauschnitt
1	 <p>Network graph showing nodes and edges for filter value 1. The graph is a complex network with many nodes and edges. The nodes are labeled with names like 'druff los ch', 'druff los mäie', 'druff los mäje', 'druff los maia', 'druff los', 'druff los mäjä', 'main', 'mäje', and 'mäje'. The edges are labeled with numerical values representing weights or distances. The graph shows a dense cluster of nodes in the upper right and a more sparse structure in the lower left.</p>
0.8	 <p>Network graph showing nodes and edges for filter value 0.8. The graph is a simplified version of the one for filter value 1, with fewer nodes and edges. The nodes are labeled with names like 'druff los ch', 'druff los mäie', 'druff los mäje', 'druff los maia', 'druff los', 'druff los mäjä', 'main', 'mäje', and 'mäje'. The edges are labeled with numerical values representing weights or distances. The graph shows a dense cluster of nodes in the upper right and a more sparse structure in the lower left.</p>

The graph illustrates a network of relationships between different 'druflos' entities. The nodes are represented by red circles, and the edges are black lines with numerical weights. The graph is divided into two main clusters by a thick vertical line. The left cluster includes nodes like 'druflos mäje', 'druflosmäjä', 'druflosmäiä', 'druflosmäje', 'druflos', and 'druff los mäie'. The right cluster includes 'druflosmäje' and 'druff los mäie'. Weights on edges range from 0.08 to 0.35.

0.3





Durch die Beispiele kann man gut sehen, dass es ein Abwägen zwischen Quantität und Qualität ist. Bei einer Levenshtein Filtervalue von 0.6, hat man «druf los» und «druflosmäje» in einer Gruppe, da diese nicht die gleiche Bedeutung haben, sprechen wir hier von einer schlechten Alinierungsqualität. Da sich bei grossen Datenmenge Fehler sehr schnell weiterverbreiten und so riesige Alignierungsgruppen entstehen, muss man den Filterwert den Daten anpassen. Um die Qualität der Alignierungsgruppen hoch zu halten verwenden wir eine strenge Filtervalue von 0.25.

Filtervalue von 1 nimmt alle Wörter und eine Filtervalue von 0 heisst, dass die komplette Entscheidung bei Double-Metaphone liegt.

3.5 Fehlerkorrektur / Interpolation / Verbesserung

Ein Ziel unserer Arbeit war das Interpolieren von Wörtern aus schlechten Sätzen mit Wörtern aus guten Sätzen. Beim Interpolieren betrachten wir nur Sätze, welche nicht durch unseren Filter ausgefiltert wurden (Siehe Kapitel 3.2 Ausfilterung). Schlechte Sätze definieren sich durch ihre Auslassungen (**).

Aufbau unserer Daten

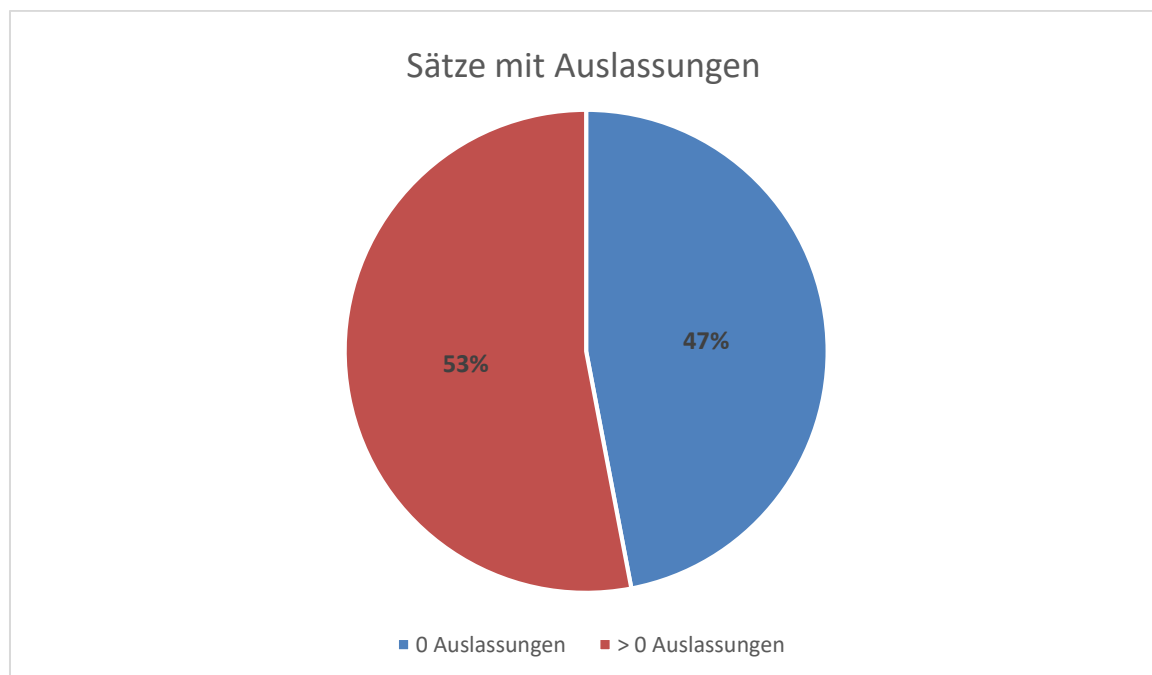


Abbildung 8: Prozentuale Verteilung der Auslassungen

Unsere Daten bestehen aus 47% auslassungsfreien Sätzen und 53% aus Sätzen mit Auslassungen (**).

Zwei Beispielsätze, wie unsere Transkriptionen aussehen können:

«D sägässli diä wärdid am oobet voräne schon tänglet ins grüschtet Dass me denn moore dess a morgete cho druflosmääje»

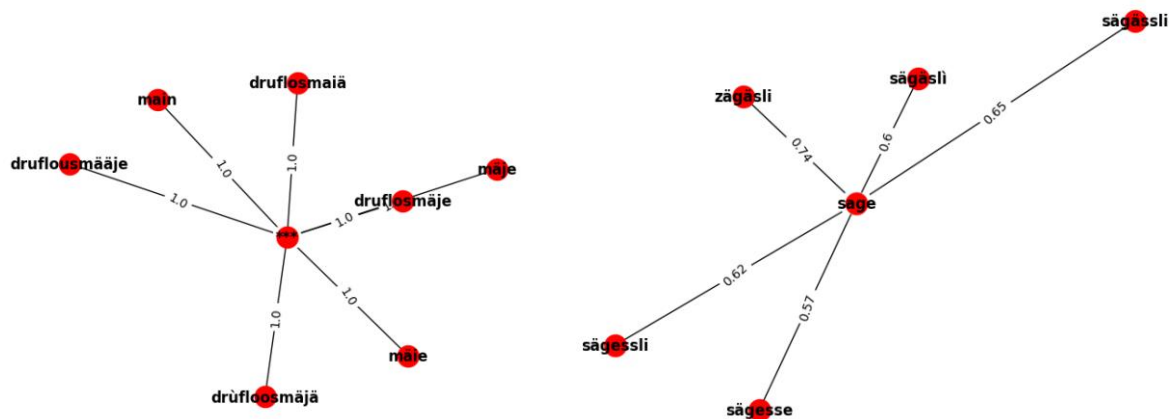
«Diä sage wered em obed scho grüschtet und tängelet dass me de more desse murgete cho ***»

Die Zweite beinhaltet eine Auslassung (**), da der Benutzer den Ausdruck «druflosmääje» nicht verstanden hat.

Diese Auslassungen wollen wir aus vollständigen Sätzen interpolieren.

Vorgehen:

Aus einer Satzgruppe wählt man den zu verbessernden Satz aus und übergibt diesen, sowie die Satzgruppe an unseren Algorithmus. Dieser aligniert dann alle anderen Sätze aus der Satzgruppe and den zu verbessernden Satz. Dies gibt uns einen Graphen, wo jeweils die Wörter des zu verbessernden Satzes die Stämme der Teil-Graphen des Graphen sind. Hier zwei dieser Teil-Graphen.



Hier ist bereits sehr gut zu sehen, dass im Teil-Graphen wo die *** den Stamm bilden, die *** zu «druflosmäaje» aligniert werden.

Normales interpolieren

bad_words = ["***", "****", "*****", "??", "???"]

Experimental

bad_words = ["***", "****", "*****", "??", "???"], jedoch fügen wir hier noch jeweils das schlechteste Wort aus einer Alignierungsgruppe hinzu. Dies Resultiert in folgender bad_words Liste für Task-ID 2048. ['***', '****', '*****', '??', '???'', 'sage', 'ggry?tët.', 'das', 'morged', 'cho']

Nach dem generieren dieses Alignments suchen wir für alle bad_words im zu verbessernden Satz eine Alternative.

Die Alternative bestimmen wir, indem wir aus der Gruppe der alignierten Wörter das Beste (siehe Kapitel Code, Best_word) nehmen. Die Alternative ersetzt das Wort aus bad_words, zurückgegeben wird der verbesserte Satz. Hier ein Beispiel, Gelb markiert die Änderungen.

Eingabe	Ausgabe
Normal	
Diä sage wered em obed scho grüschtet und tängelet dass me de more desse murgete cho ***	Diä sage wered em obed scho grüschtet und tängelet dass me de more desse murgete cho druflosmäje
Experimental	
Diä sage wered em obed scho grüschtet und tängelet dass me de more desse murgete cho ***	Diä sägässli wered em obed scho grüschtet und tängelet dass me de more desse murgete cha druflosmäje

Wie man sieht verbessern beide Versionen die ***, jedoch bei Experimental werden noch andere Worte mit Alternativen ersetzt. Dies kann zu besseren Ergebnissen führen.

Grenzen unseres Algorithmus

Beim Bestimmen des besten Wortes aus einer Alignierungsgruppe verwenden wir die Levenshtein-Distanz (siehe Kapitel Code, Best_word). Wir nehmen das Wort, welches am Ähnlichsten zu allen anderen Worten aus der Alignierungsgruppe ist. Dies ist jedoch ein Problem, wenn die Mehrheit der alignierten Wörter aus Auslassungen (***) besteht, da dann eine Auslassung mit einer anderen interpoliert wird. In der untenstehenden Grafik sieht man, dass dies bei ca. 17% aller Satzgruppen der Fall ist. Dies kann man nur durch mehr vollständige Transkriptionen verbessern. Somit kann man sagen, dass wir pro Satzgruppe mehr wie 50% vollständige Sätze für das Interpolieren benötigen.

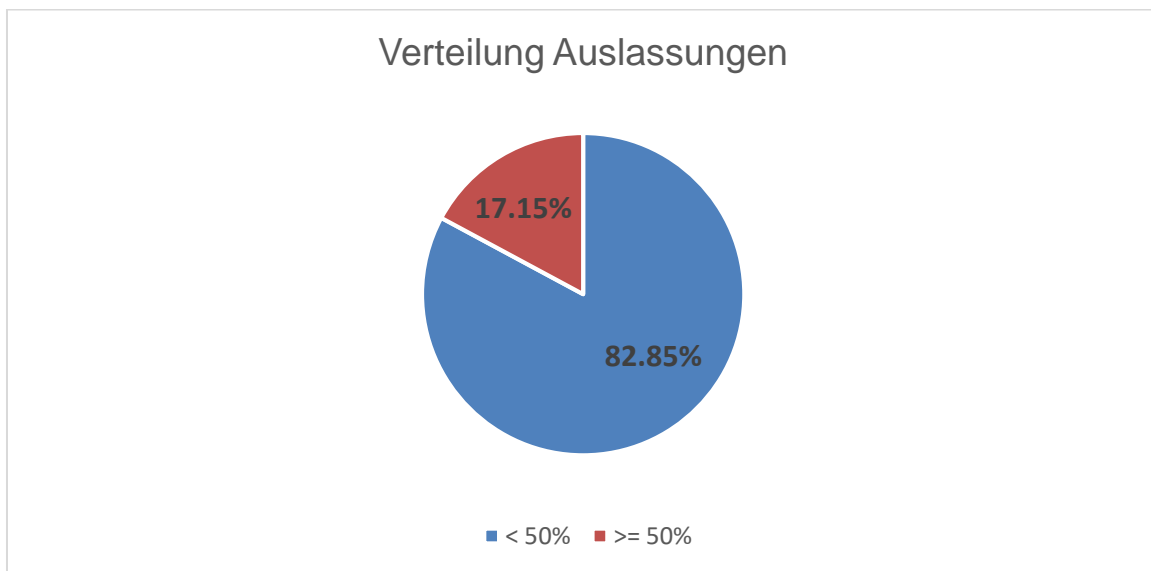


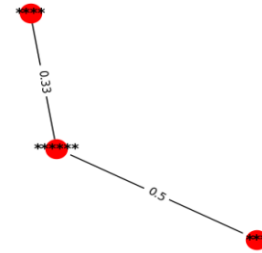
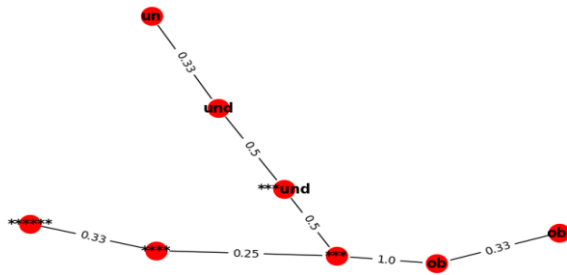
Abbildung 9: Verteilung Auslassungen über/unter 50%

Zwei Beispielsätze, wie eine schlechte Transkriptionsgruppe aussehen können:

«'De Schwigersoo waasch de Miriam iren isch den graad echli *** dän hends gfrooget über nomoll en Tälller wett oder ***** und do sägi jo Bänno und dän ischer ***** Grad dServiertochter choo umprocht, däsch grad verchlupft so schnäll heyers no niè übercho'»

«"de Schwigersoo, weisch de Miriam iren, ja, isch de graad e chli *** böös gsi dänn hends gfrooget, ob er no mol en Tälller wett oder un de isch cho ***** das hett i o ***und denn isch ***** gad d Serviertochter choo und brocht de isch grad vechlopft und so schnäll hei er's no nie übercho"»

Wenn man nun die Alignments generiert, erkennt man, dass Auslassungen mit anderen Auslassungen aligniert wurden. Beim Bestimmen des besten Wortes aus einer Alignierungsgruppe werden nun andere Auslassungen genommen.



Eingabe	Ausgabe
Normal	
De Schwigersoo waasch de Miriam iren isch den graad echli *** dän hends gfrooget über nomoll en Tälller wett oder ***** und do sägi jo Bänno und dän ischer ***** Grad dServiertochter choo umprocht, däsch grad verchlupft so schnäll heyers no niè übercho	De Schwigersoo waasch de Miriam iren isch den graad echli **** dän hends gfrooget über nomoll en Tälller wett oder ***** und do sägi jo Bänno und dän ischer ***** Grad dServiertochter choo umprocht, däsch grad verchlupft so schnäll heyers no niè übercho
Experimental	
De Schwigersoo waasch de Miriam iren isch den graad echli *** dän hends gfrooget über nomoll en Tälller wett oder ***** und do sägi jo Bänno und dän ischer ***** Grad dServiertochter choo umprocht, däsch grad verchlupft so schnäll heyers no niè übercho	De Schwigersoo waasch de Miriam iren isch den graad echli **** dän hends gfrooget über nomoll en Tälller wett oder ***** und de sägi jo Bänno und dän ischer ***** Grad dServiertochter choo umprocht, däsch grad verchlupft so schnäll heyers no niè übercho

3.6 Bewertung des Ergebnisses

Um verschiedene Parameter und Tools miteinander vergleichen zu können, muss es einen Weg geben, das generierte Alignment qualitativ zu bewerten. Was ist ein «qualitativ gutes» Alignment? Ein «qualitativ gutes» Alignment ist, wenn genau nur alle gleichbedeutenden Worte in der gleichen Alignierungsgruppe vorzufinden sind.

Beispiele

«qualitativ gutes» Alignment (2 Gruppen, alle Wörter vorhanden)

["geissziger", "gäissziger", "gäiss ziger", "gaisziger", "geiss ziger", "gäis ziger", "gäisziger"],
["laschtwagefaarer", "laschtwagafahrer", "laschtwagefahrer", "laschtwagefahrer"]

«qualitativ schlechtes» Alignment (2 Gruppen, nicht alle Wörter vorhanden)

["geissziger", "gäissziger", "gaisziger", "gäisziger"],
["laschtwagafahrer", "laschtwagefahrer"]

«qualitativ schlechtes» Alignment (3 Gruppen, alle Wörter vorhanden)

["geissziger", "gäissziger", "gaisziger", "gäisziger"],
["gäiss ziger", "geiss ziger", "gäis ziger"]
["laschtwagefaarer", "laschtwagafahrer", "laschtwagefahrer", "laschtwagefahrer"]

«qualitativ schlechtes» Alignment (1 Gruppe, alle Wörter vorhanden)

["geissziger", "gäissziger", "gäiss ziger", "gaisziger", "geiss ziger", "gäis ziger", "gäiszi-
ger", "laschtwagefaarer", "laschtwagafahrer", "laschtwagefahrer", "laschtwagefahrer"]

Um ein «qualitative gutes» Alignment zu finden, befassen wir uns mit folgenden Ansätzen:

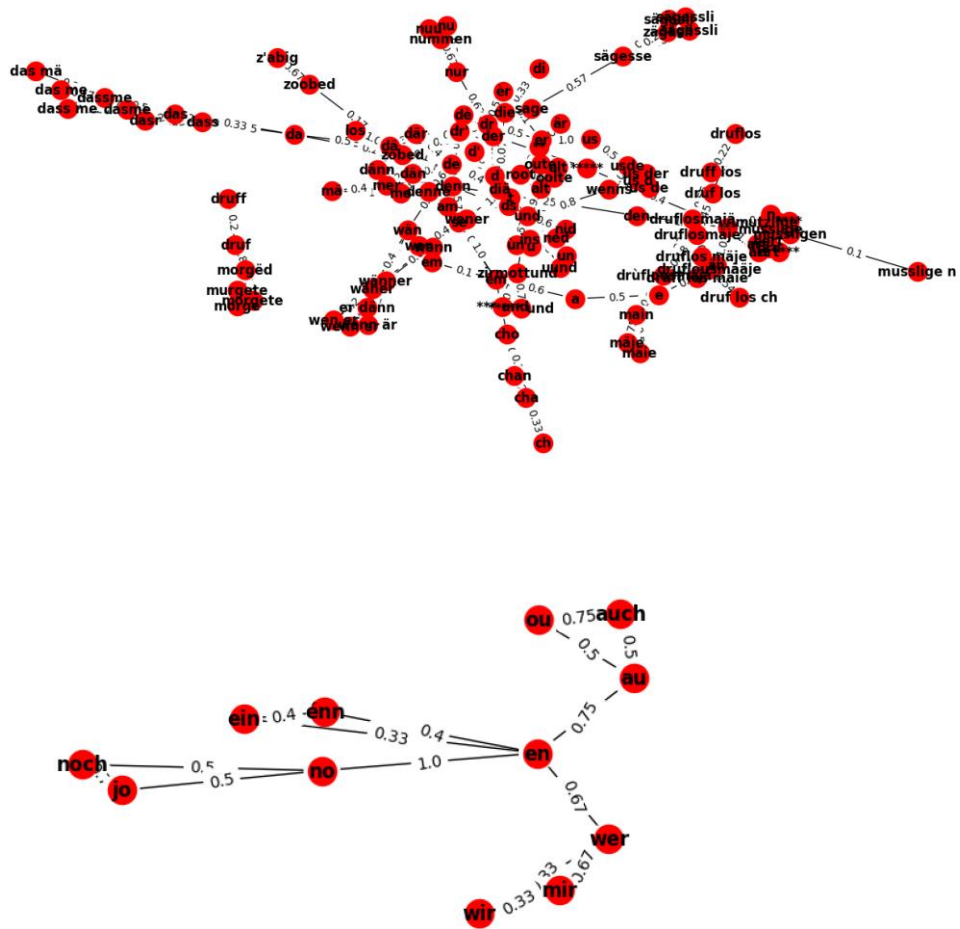
- Bewertung durch Betrachtung der Graphen von Auge
- Bewertung durch Berechnung der Wortdistanz der Alignments
- Bewertung anhand eines Goldstandards

Bewertung durch Betrachtung der Graphen von Hand

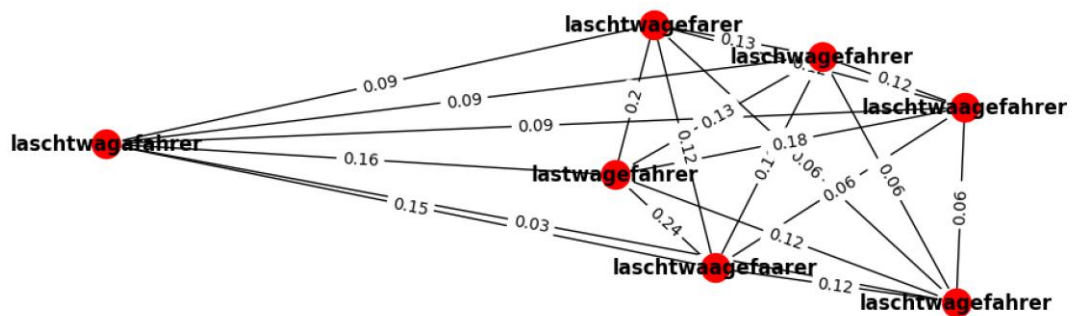
Ein plausibler Ansatz ist das Betrachten des erstellten Graphen von Hand aus. Mit dieser Methode konnten wir relativ schnell komplett schlechte Filterwerte ausschliessen, da diese die Graphen markant verändern.

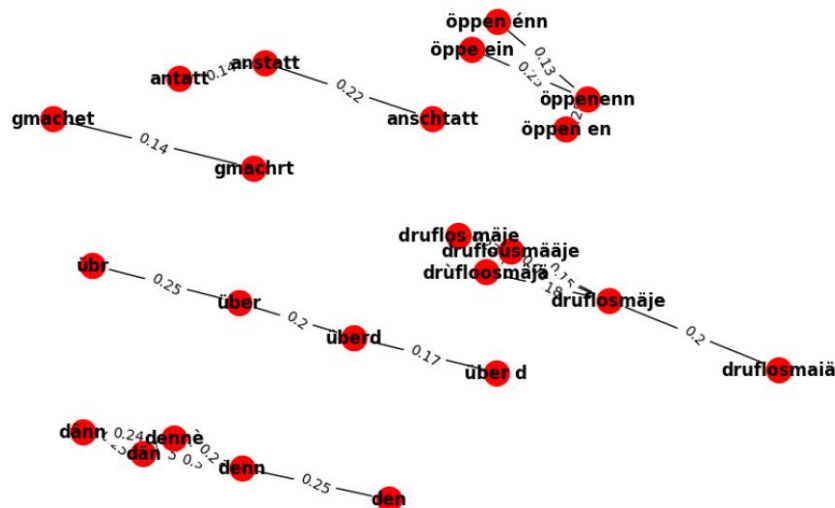
Hier ein paar Beispiele:

Schlechte Parameter



Bessere Parameter





Jedoch wird diese Methode sehr schnell unübersichtlich bei grossen Graphen, weshalb wir einen anderen Ansatz verfolgen müssen.

Bewertung durch Berechnung der Wortdistanz der Alignments

Hier ist die Idee, dass wenn die aufsummierte Wortdistanz einer Alignierungsgruppe tief ist, das Alignment auch gut sein muss.

Beispiele:

Schlechtes Alignment

["geissziger", "gäissziger", "gäiss ziger", "gaisziger", "geiss ziger", "gäis ziger", "gäiszi-ger", "laschwaagefaarer", "laschtwagafahrer", "laschtwagefahrer", "laschwaagefahrer"]

Dieses Alignment bekommt eine Bewertung von 0.4752772511208341

Gutes Alignment

["geissziger", "gäissziger", "gäiss ziger", "gaisziger", "geiss ziger", "gäis ziger", "gäisziger"]
0.08962962962962966

["laschwaagefaarer", "laschtwagafahrer", "laschtwagefahrer", "laschwaagefahrer"]
0.08363970588235294

Mit dieser Methode werden jedoch effektiv kleine Alignierungsgruppen besser Bewertet, sprich, je kleiner unser generierter Graph ist, desto besser ist die Score.

Bewertung anhand eines Goldstandards

Aus allen Transkriptionsgruppen wählen wir 5 Gruppen, welche in etwa 5 Sätze pro Gruppe haben, aus. Gruppen mit folgenden IDs [2048, 2095, 2374, 1930, 1929].

Von diesen 5 Gruppen erstellen wir von Hand ein perfektes Alignment und in speichern dieses in Graphenform ab.

Die Idee dahinter ist, dass wir dann die generierten Alignments mit dem Goldstandard vergleichen können und so «Precision», «Recall» und «Accuracy» zu berechnen.

Wir vergleichen ob ein Wort welches im Goldstandard vorkommt auch im kreierten Alignment vorzufinden ist. Weiter vergleichen wir ob die Anzahl der Alignierungsgruppen gleich der Anzahl Gruppen im Goldstandard ist. «Precision» und «Recall» sind für Binäre Klassifikation gedacht und können bei n-Klassifikationen nur unter der Bedingung verwendet werden, dass alle Gruppen bekannt sind. Da die Anzahl der Gruppen jedoch auch variiert, ist es für uns nicht möglich diese Werte zu berechnen. Ohne «Precision» und «Recall» kann man auch keine «Accuracy» berechnen.

Deshalb beschränken wir uns auf die Ausgabe von Kennzahlen, welche einen Anhaltspunkt zur Qualität liefern können.

Die Bewertungsfunktion gibt folgende Ausgaben aus, hier ein Beispiel:

Verwendung von Bleualign, Filtervalue 0.25

Params used: Aligner: bleualign/bleu-champ.exe Filtervalue: 0.25

Goldstandard Word Count: 340

Alignment Word Count: 342

Difference Word Count: 2

Words from Goldstandard not found in Alignment: 15

Words from Alignment not found in Goldstandard: 47

Goldstandard Group Count: 124

Alignment Group Count: 142

Difference Group Count: 18

Verwendung von Hunalign, Filtervalue 0.25

Params used: Aligner: Hunalign/hunalign.exe Filtervalue: 0.25

Goldstandard Word Count: 340

Alignment Word Count: 330

Difference Word Count: 10

Words from Goldstandard not found in Alignment: 22

Words from Alignment not found in Goldstandard: 42

Goldstandard Group Count: 124

Alignment Group Count: 143

Difference Group Count: 19

Graph Edit Distanz (Nicht ausprobiert) (Sanfeliu, et al., 1983)

Die Graph Edit Distanz, misst die Distanz zwischen zwei Graphen g1 und g2 indem sie die Anzahl der Änderungen, um von Graph g1 auf Graph g2 zu kommen zählt. Sie kann auch mit nicht vorkommenden Nodes oder Teilgraphen klarkommen.

Man muss jedoch beachten, dass der Algorithmus NP-Complete ist und so bei grösseren Graphen zu lange brauchen wird.

Schlussfolgerung

Unsere Ansätze liefern zwar einen Anhaltspunkt zur Qualität, können jedoch nicht zur automatischen Bewertung/Verbesserung unserer Parameter verwendet werden.

4 Code

4.1 Best_word

Mit Hilfe der Levenshtein Distanz generieren wir ein Rating der Wörter einer Alignierungsgruppe. Wir geben den Index des besten Wortes zurück.

```
def best_word(words):  
    if len(words) < 2:  
        return words[0]  
    scores = []  
    for i, word1 in enumerate(words):  
        scores.append(0)  
        for word2 in words:  
            if word1 != word2:  
                scores[i] += util.normalized_dl_distance(word1, word2)  
        scores[i] /= len(words) - 1  
    return words[scores.index(min(scores))]
```

Vorgehen:

Für jedes Wort in der übergebenen Alignierungsgruppe berechnen wir die normalisierte Levenshteindistanz zu allen anderen Wörtern in der Gruppe. Diese summieren wir auf und teilen sie schliesslich durch die Anzahl der anderen Wörter. So bekommen wir eine Wertung, die aussagt, wie nahe das Wort an allen anderen Wörtern der Gruppe ist.

5 How tos

5.1 How to load and access Data

Voraussetzungen: CSV File mit den Daten.

Code

```
# Load Data from csv
allTaskByID = rate_sentence_group('../data/TRANSCRIPTIONS.CSV')
```

Mit loadDataFromCSVFile(..) kann man die Transkriptionen in eine Variabel laden.

Dann kann man die TASK_ID übergeben um an eine Transkriptionsgruppe zu kommen.

```
# Access single transcription group
transcription_group = allTaskByID[TASK_ID]
```

5.2 How to rate a transcription group

Code

```
ratings = rate_sentence_group(allTaskByID[2048][0])
# print(ratings)
# [0.9357116953704105, 0.93685434339286, 0.8377555896409601,
1.007514941221935, 0.5174335636926569, 0.9755452596065509,
1.0449534822139588]
```

Man bekommt eine Liste mit bleu_scores für jeden Satz. Hohe Zahlen = Gute Score, Siehe Kapitel 3.1

5.3 How to directly get the good transcriptions

Code

```
# Load Data
group = allTaskByID[2048][0]
good_transcriptions = get_good_transcriptions(group)
```

get_good_transcriptions filtert Ausreisser aus und gibt die Sätze mit ihren Bleu_Scores zurück.

5.4 How to align every sentence to the others

Code

```
aligned_graph = align_every_sentence_to_the_others(group,
aligner=align.ALIGNER_BLEUALIGN, filtervalue=0.25)
```

Beim Alignen der Sätze kann man gewisse Optionen wählen.

Betreffend Filtervalue siehe: Kapitel 3.2

Betreffend Aligner siehe: Kapitel 3.3

5.5 How to align a sentence to the others

Code

```
align_to_this_ID = 0
aligned_graph = align_a_sentence_to_the_others(group, id_of_sentence_to_be_aligned_to=align_to_this_ID, aligner=align.ALIGNER_BLEUALIGN, filtervalue=0.25)
```

In Addition zu 2.4, muss man hier den Satz, zu dem aligned werden soll noch angeben. Die Optionen für Aligner und Filtervalue bleiben die Gleichen.

5.6 How to score an alignment

Code

```
print("Load Goldstandard")
gs_graph = nx.json_graph.node_link_graph(util.load_json("GoldStandard/goldstandard.json"))

print("Align Goldstandard Group")
aligned_graph = create_graph_over_list_of_groups(options.GOLD_STANDARD_SET,
aligner=align.ALIGNER_BLEUALIGN,
filtervalue=0.25)

print("Calculate Scores and print them")
calculate_alignment_score(gs_graph, aligned_graph, True)
```

Um ein Alignment zu bewerten, muss man einen Graphen, gegen den man Bewerten will laden. In obigem Beispiel ist es unser Goldstandard Graph.

In einem zweiten Schritt muss man ein Alignment über die selben Gruppen wie im obigen Graphen machen.

Diese beiden Graphen übergibt man dann an `calculate_alignment_score(...)`

5.7 How to align a list of groups of sentences

Code

```
aligned_graph = create_graph_over_list_of_groups(allTaskByID,
aligner=align.ALIGNER_BLEUALIGN, filtervalue=0.25)
```

Das erste Argument der Funktion ist eine Liste von Satzgruppen, sprich man kann direkt mehrere Tasks auf einmal in den gleichen Graphen alinieren lassen.

Es wird auch ein Fortschrittsbalken angezeigt, was bei grösseren Datenmengen eine Übersicht für den Fortschritt zeigt.

5.8 How to improve a sentence

Man kann schlechte Sätze verbessern lassen, das heisst man wählt den zu verbesserten Satz aus und übergibt ihn der Funktion. Als Rückgabe erhält man den verbesserten Satz.

Hier ein Beispiel mit Task_2048 und dem zu verbessernden Satz Nummer 4. Gelb Markiert die Änderungen.

Eingabe	Ausgabe
Normal	
Diä sage wered em obed scho grüschtet und tängelet dass me de more desse murgete cho ***	Diä sage wered em obed scho grüschtet und tängelet dass me de more desse murgete cho druflosmäje
Experimental	
Diä sage wered em obed scho grüschtet und tängelet dass me de more desse murgete cho ***	Diä sägässli wered em obed scho grüschtet und tängelet dass me de more desse murgete cha druflosmäje

5.8.1 Normal

Code

```
improved = improve_sentence(group, group[3], experimental_improve=False)
```

5.8.2 Mit experimental bad_word_detection

Code

```
improved = improve_sentence(group, group[3], experimental_improve=True)
```

5.9 How to print a graph

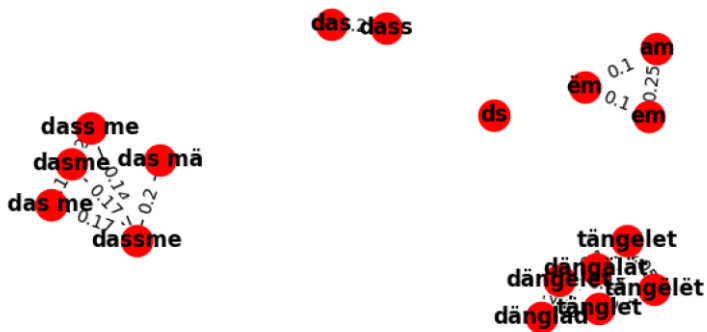
Voraussetzung: Graphen

Code

```
print_graph_with_edges(aligned_graph)
```

Dies wird mit matplotlib einen Graphen in einem zusätzlichen Fenster öffnen. Achtung, dies geht bei sehr grossen Graphen sehr lange.

Beispiel



5.10 How to export as a graph

Voraussetzung: Graphen

Code

```
export as graph(aligned_graph, "dumpedGraph.json")
```

Exportiert den erstellten Graphen in einem JSON Format. Dies kann mit anderen Graphen Libraries gelesen werden.

5.11 How to export as a list

Voraussetzung: Graphen

Um in ein lesbareres Format zu exportieren, kann man folgende Funktion aufrufen.

Code

```
export as list(aligned_graph, "dumpedGraph.json")
```

Die speichert es als Liste in einem JSON File ab.

5.12 How to import a graph

Voraussetzung: Exportierten Graphen im Graph Format.

Code

```
imported_graph = import as_graph("dumpedGraph.json")
```

Importiert den Graphen.

6 Schlussfolgerung

Satzbewertung

Sätze innerhalb einer Satzgruppe können mit dem selben Ansatz bewertet werden, wie er auch für die Bewertung von Maschinenübersetzungen verwendet wird. Je ähnlicher eine Maschinenübersetzung einer menschlichen Übersetzung ist, desto besser ist sie. Für unser Problem bedeutet das, je ähnlicher eine Transkription den anderen Transkriptionen ist, desto besser ist sie. Es lassen sich die selben Werkzeuge anwenden. BLEU, welcher bei der Bewertung von Maschinenübersetzungen ein Industriestandard ist, funktioniert auch mit unseren Daten. Für optimale Ergebnisse sollte eine Smoothing Function verwendet werden.

Ausfilterung unbrauchbarer Sätze

Die Ausfilterung beruht auf der Bewertung. Es wird ein simpler Grenzwert festgelegt. Als Standard empfehlen wir den Wert 0.5. Je nach Benutzerbedürfnissen kann er aber zwischen 0.3 für mehr, dafür qualitativ schlechtere und 0.7 für weniger, dafür qualitativ bessere Sätze frei gewählt werden.

Alignierung

Für die Wortalignierung verwenden wir Satzaligner, da diese genauer unseren Einschränkungen entsprechen, da die Anordnung der Wörter stabil ist. Wir haben die zwei etablierten Aligner Hunalign und Bleualign verwendet und mit beiden ähnliche Resultate erzielt. Zur Erweiterung eines einfachen Alignments zwischen zwei Sätzen bauen wir einen Graphen mit Wörtern als Knoten auf und betrachten dann die daraus entstehenden Komponenten als Gruppen gleichbedeutender Ausdrücke.

Da die durch den Aligner gebildeten Ausdruckspaare nicht zu einhundert Prozent zuverlässig sind, ist es nötig, jedes alignierte Ausdruckspaar auf Ähnlichkeit zu überprüfen und allenfalls zu verwerfen. Hierfür verwenden wir die Damerau-Levenshtein Distanz kombiniert mit Double-Metaphone. Der Grenzwert für die Damerau-Levenshtein Distanz hat grossen Einfluss auf das Gesamtergebnis. Bei grossen Datenmengen empfehlen wir einen Wert von 0.25.

Fehlerkorrektur / Interpolation

Wir können Auslassungen (***) in einem Satz aus anderen Sätzen interpolieren, indem wir ihn mit den anderen Sätzen alignieren und dann aus der entsprechenden Wortgruppe ein anderes Wort auswählen. Das Verfahren funktioniert jedoch nur, solange grundsätzlich genügend gute Transkriptionen vorhanden sind.

Bewertung des Alignments

Wir liefern Kennzahlen, die es ermöglichen, die Auswirkungen verschiedener Parameter auf das Ergebnis einzuschätzen. Für eine absolute Bewertung ist jedoch immer noch Augenmass gefordert, da keiner der uns bekannten Ansätze ein verlässliches Ergebnis liefert.

Software

Wir haben ein Python Framework konstruiert, das es ermöglicht, alle von uns vorgestellten Verfahren auf die bestehenden und zukünftigen Daten anzuwenden. Es kann mit geringem Aufwand um weitere Algorithmen und Tools erweitert werden.

Welche Tools funktionieren?

Es gibt keine NLP-Tools, die für genau unsere Zwecke konzipiert sind. Es ist jedoch möglich, Tools aus anderen Gebieten anzuwenden. BLEU kann problemlos zur Satzbewertung verwendet werden. Satzaligner wie Hunalign und Bleualign können mit etwas Mehraufwand als Wortaligner verwendet werden, solange die Wortstellung in den Sätzen stabil bleibt. Levenshtein kann für den Vergleich einzelner Wörter verwendet werden. Wir empfehlen eine eigene Gewichtung für einige der Buchstabenpaare. Double-Metaphone alleine funktioniert nur unzulänglich mit schweizerdeutschen Wörtern, kann jedoch gut in Kombination mit Levenshtein verwendet werden. Wir haben viele weitere Tools aufgelistet, die wir nicht genauer

testen konnten. Wir erwarten, dass die meisten Ergebnisse liefern, die mit den Ergebnissen der von uns verwendeten Tools vergleichbar sind.

Wie viele Transkriptionen werden benötigt?

Die Menge der benötigten Transkriptionen hängt primär von ihrer Qualität ab. Zwei bis drei zusammengehörige Transkriptionen sind bereits genug, wenn alle qualitativ gut sind. Pro Satzgruppe erwarten wir kaum mehr als einen unbrauchbaren Satz. Für die Interpolation sind so viele Transkriptionen nötig, dass eine Mehrheit davon ein gegebenes Wort korrekt transkribiert. Dies ist deshalb schwierig, da die meisten Benutzer mit denselben Wörtern Mühe haben.

7 Abbildungsverzeichnis

Abbildung 1: Alle Transformationen	5
Abbildung 2: Verteilung der Transkriptionen	7
Abbildung 3: Prozentuale Verteilung der Auslassungen	8
Abbildung 4: Relationen der Sätze bezüglich Ähnlichkeit	10
Abbildung 5: Histogramm der Bewertungen aller Sätze	12
Abbildung 6: Wortalignierung als bipartiter Graph, von Redienss (CC BY-SA 3.0)	16
Abbildung 7: Kombination von Ausdruckspaaren in einen einzelnen Graphen	17
Abbildung 8: Prozentuale Verteilung der Auslassungen	27
Abbildung 9: Verteilung Auslassungen über/unter 50%	29

8 Literaturverzeichnis

- Abdul Rauf, Sadaf, et al. 1012.** Evaluation of Sentence Alignment Systems. 1012.
- Banerjee, Satanjeev und Lavie, Alon. 2005.** METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*. 29. 6 2005, S. 65-72.
- Beider, Alexander und Morse, Stephen P. 2010.** Phonetic Matching: A Better. *Association of Professional Genealogists Quarterly*. 2010.
- Chen, Boxing und Cherry, Colin. 2014.** A Systematic Comparison of Smoothing Techniques for Sentence-Level BLEU. *Proceedings of the Ninth Workshop on Statistical Machine Translation*. 6 2014, S. 362-367.
- Doddington, George. 2002.** Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. *Proceeding HLT '02 Proceedings of the second international conference on Human Language Technology Research*. 24. 4 2002, S. 138-145.
- Dyer, Chris , Chahuneau, Victor und Smith, Noah A. 2013.** A Simple, Fast, and Effective Reparameterization of IBM Model 2. *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2013.
- Fairchild, Geoffrey. 2017.** pyxDamerauLevenshtein. *GitHub*. [Online] 26. 9 2017. [Zitat vom: 19. 1 2018.] <https://github.com/gfairchild/pyxDamerauLevenshtein>.
- Han, Aaron L.-F, Wong, Derek F. und Chao, Lidia S. 2012.** LEPOR: A Robust Evaluation Metric for Machine Translation with Augmented Factors. 12 2012, S. 441-450.
- Jenks, George F. 1967.** The Data Model Concept in Statistical Mapping. *International Yearbook of Cartography* 7. 1967, S. 186–190.
- Levenshtein. 1966.** Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*. 11 1966, S. 707-710.
- Nádovrník, Jiří. 2012.** python-fizzle. *GitHub*. [Online] 18. 12 2012. [Zitat vom: 19. 1 2018.] <https://github.com/nadvornix/python-fizzle>.
- Papineni, Kishore, et al. 2002.** BLEU: a Method for Automatic Evaluation of Machine Translations. *Proceeding ACL '02 Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. 12. 7 2002, S. 311-318.
- Rodichevski, Alexandre.** Approximate string-matching algorithms, part 1. *Morfoedro, a portal on arts and culture*. [Online] [Zitat vom: 19. 1 2018.] <http://www.morfoedro.it/doc.php?n=222&lang=en>.
- Russell, Robert C. 1918.** *Soundex. US1261167 A USA*, 2. 4 1918. Grant.
- Sanfeliu, Alberto und Fu, King-Sun. 1983.** A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man and Cybernetics*. 1983.

Sennrich, Rico und Volk, Martin. 2011. Iterative, MT-based sentence alignment of parallel texts. *NODALIDA 2011, Nordic Conference of Computational Linguistics*. 2011.

—. **2010.** MT-based Sentence Alignment for OCR-generated Parallel Texts. *Proceedings of AMTA 2010*. 2010.

Somerville, M. 2010. Double-Metaphone. *GitHub*. [Online] 3. 11 2010. [Zitat vom: 19. 1 2018.] <https://github.com/dracos/double-metaphone>.

Su, David. 2017. Weighted Levenshtein library. *readthedocs*. [Online] 23. 10 2017. [Zitat vom: 19. 1 2018.] <http://weighted-levenshtein.readthedocs.io/en/master/>.

Varga, Dániel , et al. 2007. Parallel corpora for medium density languages. *AMSTERDAM STUDIES IN THE THEORY AND HISTORY OF LINGUISTIC SCIENCE SERIES 4*. 2007.

Ehrlichkeitserklärung

Hiermit erklären wir, die vorliegende Projektarbeit selbständig, ohne Hilfe Dritter und nur unter Benutzung der angegebenen Quellen verfasst zu haben.

19.01.2018 Fukuoka, Japan



Ort und Datum

Unterschrift Matthias Ernst

Ort und Datum

Unterschrift Fabio Strappazon