IP5 - NLP Schweizerdeutsch

Fabio Strappazzon, 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](#_Toc504034148)

[1.1 Zusammenfassung 4](#_Toc504034149)

[1.2 Projektkontext 4](#_Toc504034150)

[1.3 Problemstellung 4](#_Toc504034151)

[1.4 Vorgehen 4](#_Toc504034152)

[2 Daten 7](#_Toc504034153)

[2.1 Charakteristiken der Daten 7](#_Toc504034154)

[3 Alignierung von schweizerdeutschen Transkriptionen 9](#_Toc504034155)

[3.1 Satzbewertung 9](#_Toc504034156)

[3.2 Ausfilterung 11](#_Toc504034157)

[3.3 Wortalignierung 15](#_Toc504034158)

[3.4 Filterung des Alignments 16](#_Toc504034159)

[3.5 Fehlerkorrektur / Interpolation / Verbesserung 25](#_Toc504034160)

[3.6 Bewertung des Ergebnisses 28](#_Toc504034161)

[4 Ergebnis 29](#_Toc504034162)

[5 Code 30](#_Toc504034163)

[5.1 Best\_word 30](#_Toc504034164)

[6 How to’s 31](#_Toc504034165)

[6.1 How to load and access Data 31](#_Toc504034166)

[6.2 How to rate a transcription group 31](#_Toc504034167)

[6.3 How to directly get the good transcriptions 31](#_Toc504034168)

[6.4 How to align every sentence to the others 31](#_Toc504034169)

[6.5 How to align a sentence to the others 32](#_Toc504034170)

[6.6 How to align a list of groups of sentences 32](#_Toc504034171)

[6.7 How to improve a sentence 32](#_Toc504034172)

[6.7.1 Normal 32](#_Toc504034173)

[6.7.2 Mit experimental bad\_word\_detection 32](#_Toc504034174)

[6.8 How to print a graph 33](#_Toc504034175)

[6.9 How to export as a graph 33](#_Toc504034176)

[6.10 How to export as a list 33](#_Toc504034177)

[6.11 How to import a graph 33](#_Toc504034178)

[7 Schlussfolgerung 34](#_Toc504034179)

[8 Literaturverzeichnis 34](#_Toc504034180)

# Einführung

## Zusammenfassung

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

## Projektkontext

Im Rahmen des vom Schweizerischen Nationalfods geförderten Agora-Project «Citizen Linguistics: Locate that dialect!» ist die Website [www.dindialaekt.ch/](http://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.

## 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?**

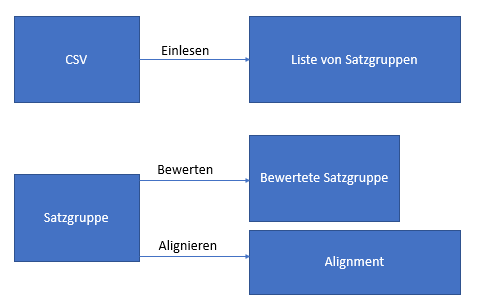
Natural Language Processing (NLP) ist ein etabliertes Gebiet. Für viele Problemstellungen gibt es Tools und Algorithmen. Viele dieser Tools haben jedoch gewisse Voraussetzungen. Beispielsweise benötigen sie einen Textkorpus einer gewissen Grösse oder sie benötigen Wörterbücher der zu untersuchenden Sprachen. 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 and 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.

## Vorgehen

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



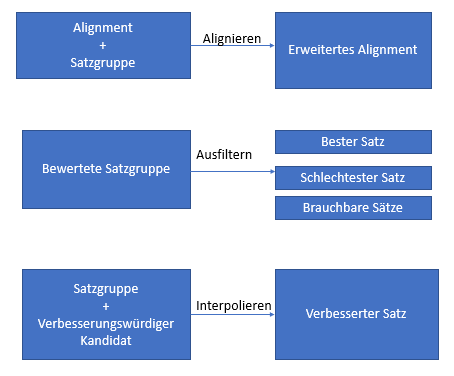


Abbildung 1: Alle Transformationen

Sätze einer Satzgruppe können nach Qualität bewertet werden. Die 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 List 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.

## Ergebnis

todo

# 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 höwe héts aube ghéésse; <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 zämetribe 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.

## Charakteristiken der Daten

Stand 2017-07-08

2848 Transkriptionen zu 554 verschiedenen Audiodateien.

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 för d' Härdöpfel understuen fascht alli hän da un dert än Härdöpfelblätz. Ds Grosis ämel o.»

«Es isch die Ziit, wo d'Liit afend üsziehn fer d'Härdöpfu 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: Fehlerhafte Sätze aufgeschlüsselt nach Anzahl Sätze in Satzgruppe.

In der Grafik erkennt man einige Dinge, zum Beispiel, dass bei nur einem Input Satz 40% dieser nicht brauchbar sind, da Wörter darin fehlen und mit Sternen ersetzt wurden. Spannend ist auch, dass bei 8 Input Sätzen die Qualität zunimmt.

# Alignierung von schweizerdeutschen Transkriptionen

## 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 Alignierungsergebnis. 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 geschtossed.
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 usenangèrègschüttlet 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.

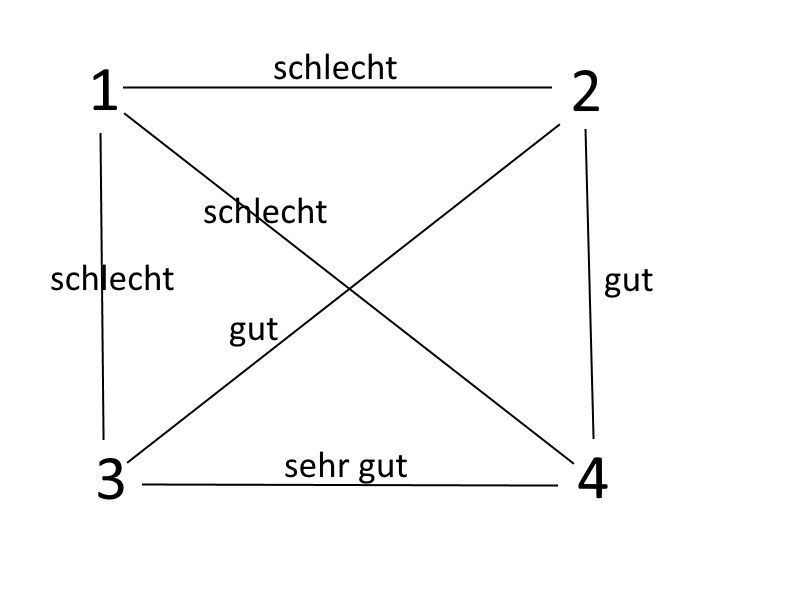
Wenn wir in der Lage sind, die Ähnlichkeit zwischen den Sätzen zu berechnen, können wir also relativ gut entscheiden, wie «gut» eine Transkription ist. 

Abbildung 2: 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 Dokumentebene konzipiert. Auf der Satzebene, wie wir sie hier verwenden, liefern sie etwas schlechtere Ergebnisse. Für BLEU wurden deshalb einige sogenannte Smoothing Funktions 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.

## 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.

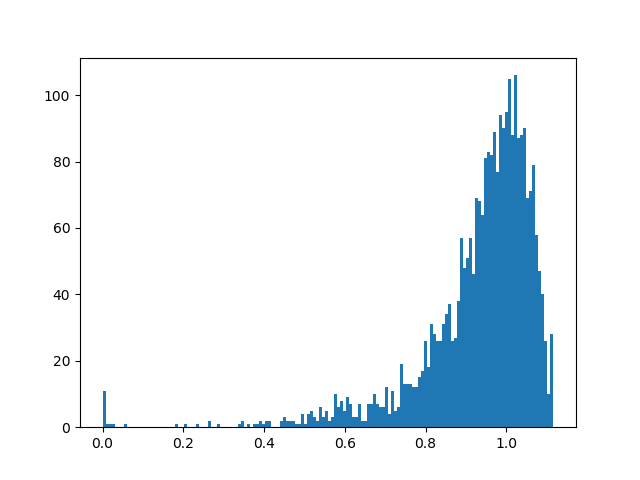


Abbildung 3: Histogramm der Bewertungen aller Sätze

Aus Abbildung 2 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 | élk ék ékl ékl 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 gschtoosse |
| 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 interessierts würkli 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 Früestück rüschte und das fueret denn e chli und de mögeds de die eiwer Lüüt ender öppe bis zum Zmittag 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 träit es Bürdeli ei jede träit e Teil wärs eppis zum Verchräämere, es häts e jede feil. |
| 0.6588 | Ein jeder wollte mehr Land besitzen, .. und sie wurden sich nicht einig. 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 Herbscht emene \*\*\* übergä. dä miends die Gäischtleche Bihäärdig und de Tokter is Aesse iilade und dem Eerebrediger de Chääs psoorge | de Bärgamme müesti allig im Herbscht emene Wiil d'Chilbi vergä, de miends di Geischtliche, 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 angers - jo das wei mer denn luege, ja | oderes Bögli für è Chääs üüseznäh und det hät hans fäärè gseit das sig z churzes damol luegtemer de scho z gsee fürnes anders |
| 0.65 | jetze chumensch güetlech witer und lüegen uber de abrosch uf ds Dach u gseh - da isch ds zennusch hüt üsgspreiteti gsi | Jetzt bechums güet för hütte, und lüegend öber 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-Iise 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 woorde und er häig Rossiise vo bloosser hand abenanggrisse |

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.

## 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.

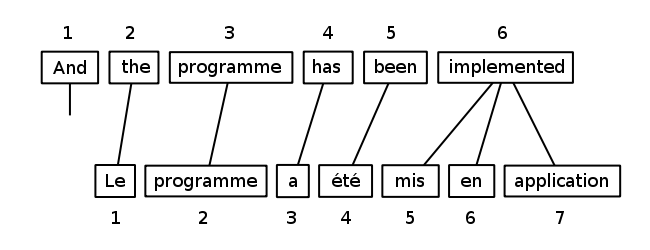


Abbildung 4: Wortalignierung als bipartiter Graph, von Redienss (CC BY-SA 3.0)

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 Hunalign) ausprobiert.

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

Hunalign (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 Hunalign 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.

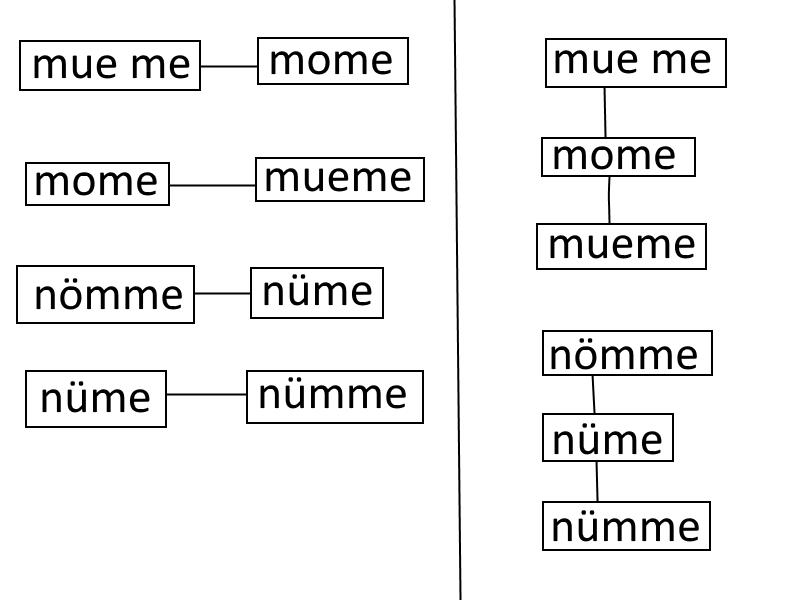


Abbildung 5: Kombination von Ausdruckspaaren in einen einzelnen Graphen

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 wir als Hauptsatz ausgewählt und dann mit allen anders 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 und bei der durchschnittlichen Satzgruppe mit 5 Sätzen 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 Ausdruckspaar, 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.

## 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:**

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

**Levenshtein-Distanz**

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**

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**

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

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

**Soundex**

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**

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**

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' wir 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 | Buchtstabe 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 | Levensthein Bewertung | Resultat |
| OK | OK | NICHT AUSGEFILTERT |
| OK | FAILED | NICHT AUSGEFILTERT |
| FAILED | OK | NICHT AUSGEFILTERT |
| FAILED | FAILED | AUSGEFILTERT |

**Hier ein paar Beispiele:**

Params: Levensthein Filter (0.4)

|  |  |
| --- | --- |
| Wortpaar mit Levenstheinvalue und Metaphone Bewertung | Wird ausgefiltert |
| ('sägäslì', 'sägessli', 0.275, False) | Nein, da Levensthein OK |
| ('de', 'die', 0.3333333333333333, True) | Nein, da beide OK |
| ('wèrdët', '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 Levenstheindistanz 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 Levensthein und Double-Metaphone einig sind, sprich sie beide das Wortpaar als schlecht bewerten, wird keine Verbindung der Wörter im Graphen erstellt.

Zeitpunkt der Bewertung des Alignments

**Pro Satzpaar**

Zwei Sätze werden aligniert und dann werden die daraus resultierenden Wortpaare bewertet. Schlechte Paare werden in diesem Schritt direkt verworfen, siehe Algorithmen oben, die guten in einem Graphen abgelegt. Die Wörter sind die Nodes.

**Pro Satzgruppe**

Alle Sätze einer Satzgruppe werden miteinander aligniert, ein Graphen mit alignierten Wörter wird kreiert. Alle Wörter werden vor dem Einfügen in den Graphen bewertet und ausgefiltert.

**Global**

Alle Sätze werden miteinander aligniert, ein Graphen mit alignierten Wörter wird kreiert. Alle Wörter werden vor dem Einfügen in den Graphen bewertet und ausgefiltert.

**Tester**

**TODO**

**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 druflousmääje
* \*\*\* Ds Zägäsli diä wärdid am Obed vor alle scho dängläd u grüschtet dasme denn am More desse Morgete chan druflosmaiä.
* 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 wèrdët dä ëm Aabëd forhanä ?ù tängëlët ùn ggry?tët. das mä dän ëm moorä det ëm Morgëd dä cha drùfloosmäjä.
* und Sägesse werded dänn scho am Obed voranne dängelet und grüschtet das mer se more des morge de cha druflos mäje
* d Sägessli die wäärded 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.

|  |  |
| --- | --- |
| **Filtervalue für Levenshtein** | **Graphauschnitt** |
| **1** |  |
| **0.8** |  |
| **0.6** |  |
| **0.4** |  |
| **0.3** |  |
| **0.2** |  |
| **0.0** |  |

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 in «druf los» und «druflosmäje» in einer Gruppe. Wir stellten uns dann die Frage ob dies für unsere Aufgabe Sinn macht und kamen zum Schluss, dass uns Qualität viel wichtiger ist wie Quantität. Da man in einer Alignierungsgruppe nur das selbe Wort, in anderen Schreibweisen finden möchte und nicht etwas mit einer anderen Bedeutung («druf los» und «druflosmäje» ist nicht das Gleiche). Deshalb haben wir uns dazu entschieden, die Filtervalue für Levenshtein auf streng zu setzen, sprich um die 0.25 herum. Natürlich kann man diese jederzeit anpassen.

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

## Fehlerkorrektur / Interpolation / Verbesserung

Ein Ziel unserer Arbeit war das Interpolieren von Wörtern aus schlechten Sätzen mit Wörtern aus guten Sätzen.

Mehr wie die Hälfte aller Sätze (53%) weisen fehlende Wörter auf. Wenn man diese Sätze nun einfach ignorieren würde, müsste man viel mehr Daten sammeln um auf die gleiche Anzahl an zueinander alignierten Wörter zu kommen. Wir versuchen mit unserem Interpolationsvorgehen, dieses verschwendete Potential zu retten.

**Aufbau unserer Daten**

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 druflousmääje»

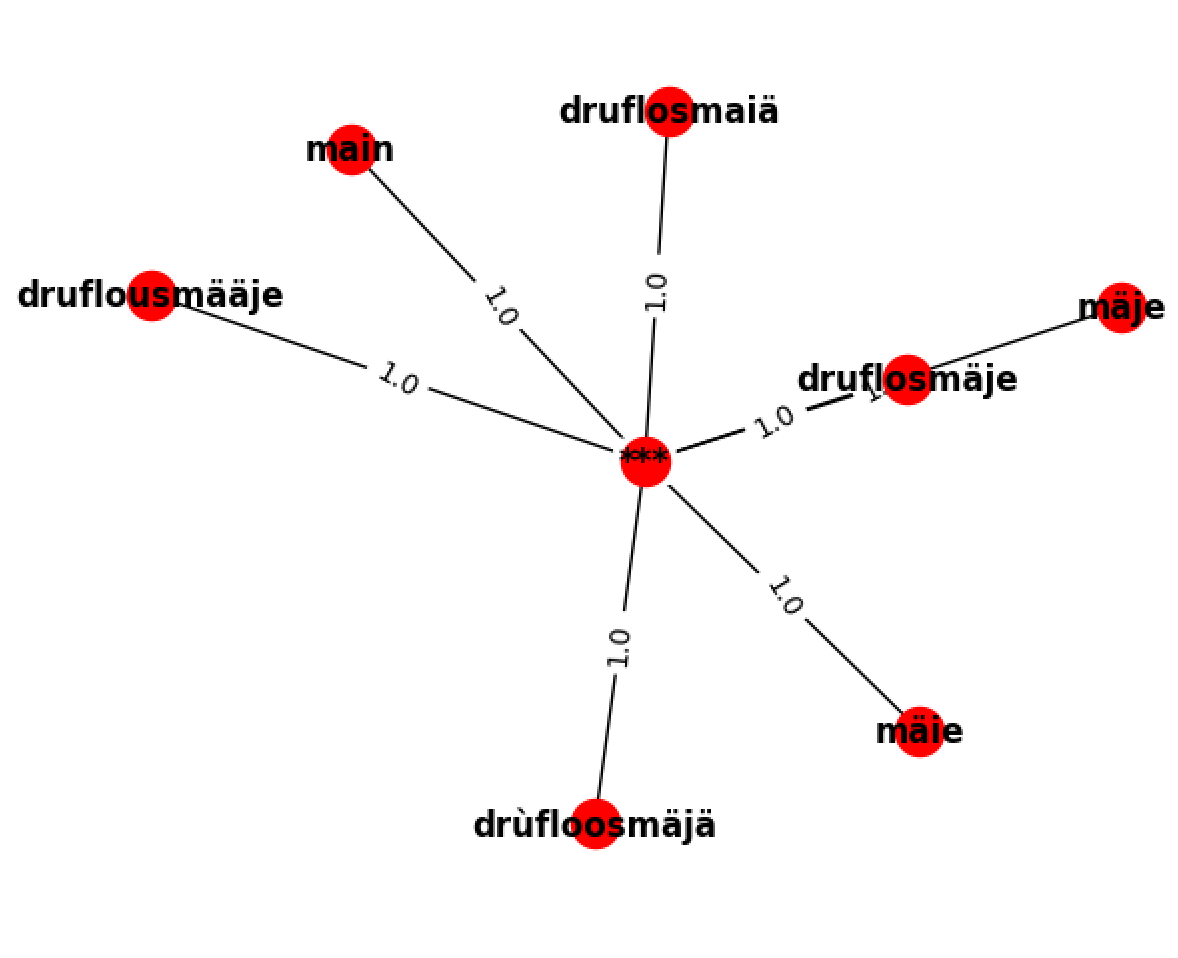
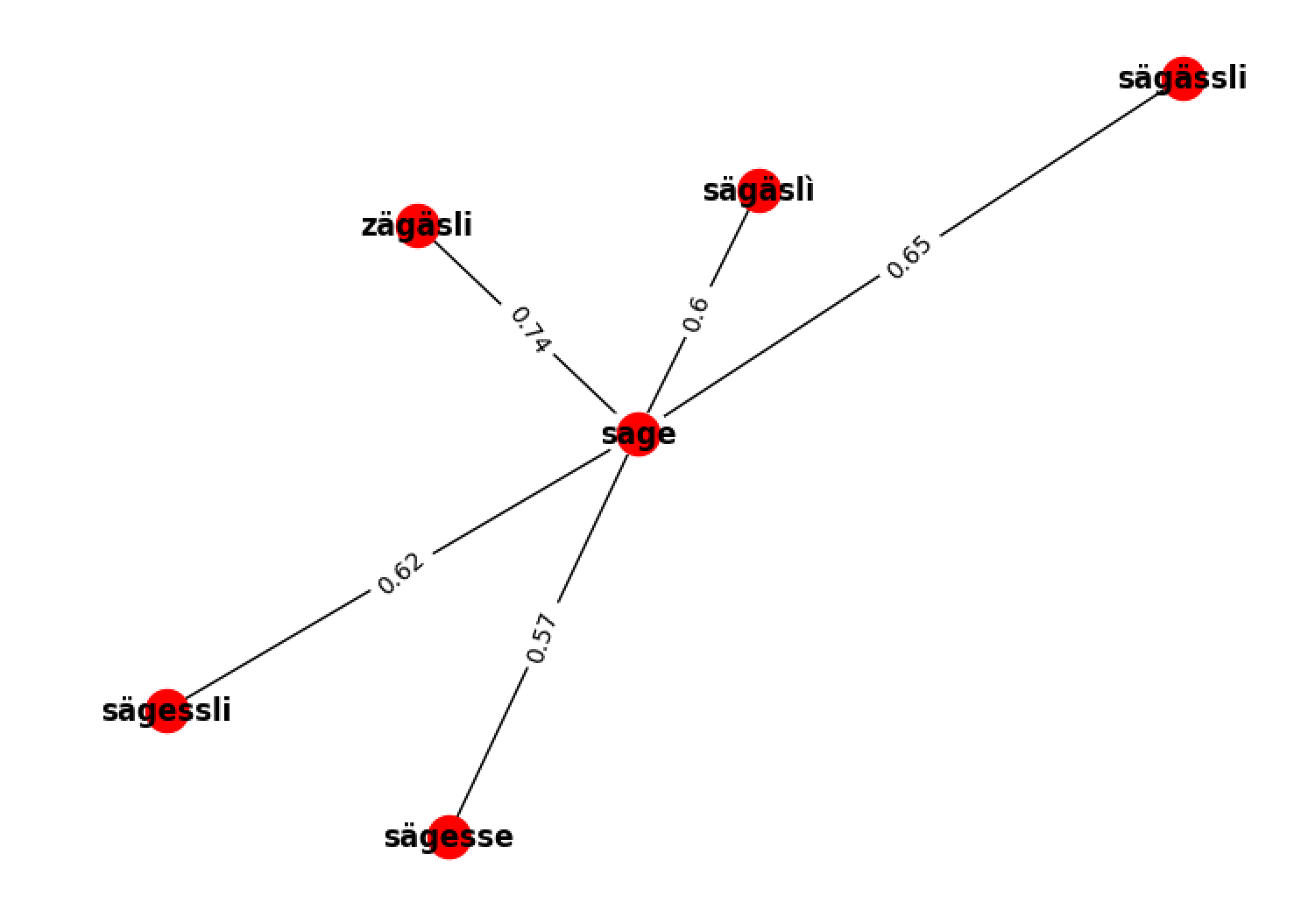
«Diä sage wered em obed scho grüschtet und tängelet dass me de more desse murgete cho \*\*\*»

Die Zweite beinhaltet eine Auslassung (\*\*\*), da der Benutzer den Ausdruck «druflousmääje» 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 aligned dann alle anderen Sätze aus der Satzgruppe and den zu verbesserden 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 «druflousmääje» 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', 'morgëd', '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 Wörter mit Alternativen ersetzt. Dies kann zu besseren Ergebnissen führen.

## Bewertung des Ergebnisses

Um verschiedene Parameter und Tools miteinander vergleichen zu können, muss es einen Weg geben, das generierte Alignment qualitativ zu bewerten.

Wir haben einen Goldstandard für die Alignierung von Wörtern aus Task 1 erstellt, mit diesem werden wir die verschiedenen Methoden zur Alignierung messen und so vergleichen können.

.

Wir haben zum Testen folgende GOLD\_STANDARD\_SET = [2048, 2095, 2374, 1930, 1929] Satzgruppen genommen.

# Ergebnis

# Code

## Best\_word

Mit Hilfe der Levensthein 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 Levenstheindistanz 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.

# How tos

## 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]

## 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

## 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.

## 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

## 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 bleiden die Gleichen.

## 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.

## How to improve a sentence

Man kann schlechte Sätze verbessern lassen, das heisst man wählt den zu verbesserden 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 |

### Normal

Code

improved = improve\_sentence(group, group[3], experimental\_improve=**False**)

### Mit experimental bad\_word\_detection

Code

improved = improve\_sentence(group, group[3], experimental\_improve=**True**)

## How to print a graph

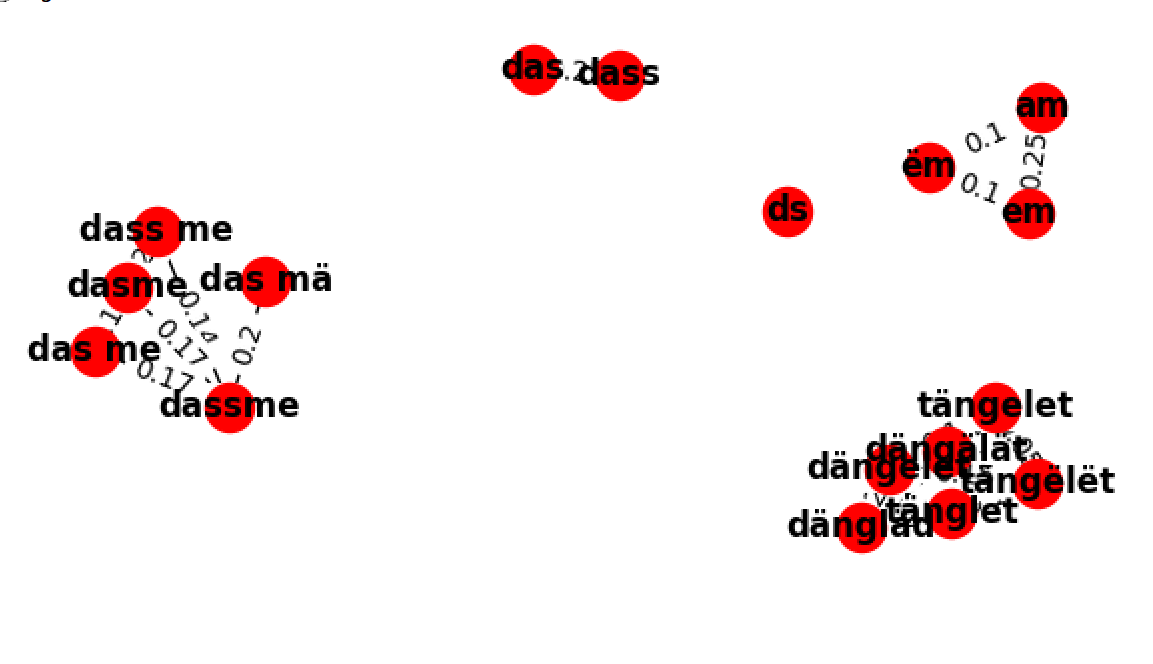
Voraussetzung: Graphen

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



## 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.

## 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.

## How to import a graph

Voraussetzung: Exportierten Graphen im Graph Format.

Code

imported\_graph = import\_as\_graph("dumpedGraph.json")

Importiert den Graphen.

# Schlussfolgerung

# Abbildungsverzeichnis

[Abbildung 1: Alle Transformationen 5](#_Toc504123765)

[Abbildung 2: Relationen der Sätze bezüglich Ähnlichkeit 10](#_Toc504123766)

[Abbildung 3: Histogramm der Bewertungen aller Sätze 12](#_Toc504123767)

[Abbildung 4: Wortalignierung als bipartiter Graph, von Redienss (CC BY-SA 3.0) 16](#_Toc504123768)

# 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.

**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.