Vergleich der Silbenmodelle des Russischen mit Hilfe eines Natural Language Processing-Ansatzes

Wissenschaftliche Hausarbeit
zur Erlangung des akademischen Grades
Bachelor of Arts (B.A.)
der Universität Hamburg

vorgelegt von
Igor Fischer
geboren in
Temirtau (Kasachstan)

Hamburg 2014

Inhaltsverzeichnis

Einleitung

„Syllables govern the world“, sagte einmal der irisch-britische Schriftsteller George B. Shaw. Auch wenn er damit wohl die Macht von Wörtern beschreiben wollte, hat er doch zwei Tatsachen ausgedrückt: Wörter bestehen aus Silben und Silbe als Begriff ist nicht nur für akademische Kreise reserviert. Goldsmith (2011, S. 164) schreibt zur Rolle in der Lingustik: „The syllable is one of the oldest constructs in the study of language, and most studies of phonology have found a place for the syllable within them.“ Trotz der langen Geschichte bleibt der Begriff umstritten. In der Russistik gibt es zumindest 3 rein russistische Silbenmodelle. An der Theorie fehlt es allerdings nicht, aber an der Computerisierung.

Die vorliegende Arbeit soll einen Beitrag zum Aufheben dieses Umstandes leisten. Tatsächlich soll nicht noch ein theoretischer Beitrag zur Silbe geleistet werden, sondern die Russistik um einen praktischen Ansatz bereichert werden. Die konkreten Subziele sind zum einen, dem Forscher die Wahl zu lassen, unter welchem Modell er einen Text oder eine Aufgabe betrachten will. Andererseits soll alles leicht modulierbar und ergänzbar sein, damit der Forscher die Anwendung seinen Zielen anpassen kann.

Dies soll mit einem hier entwickelten Programm erreicht werden. Dieses soll aus state-of-art Elementen geschaffen sein, die es mittelfristig zu mehr machen können als einer Momentaufnahme. Zum einen wird Python als Sprache verwendet, die in Data Science und akademischer Welt sehr erfolgreich ist. Zum anderen basiert das Programm auf pandas, einer Programmbibliothek für Python. Dieses Projekt von 2008 ist mit 5 Jahren sehr jung, doch in Kombination mit Python macht es durch die Zugänglichkeit anderen Programmierprachen wie R, SAS oder C++ in der Finanzindustrie bereits jetzt Konkurrenz: „Institutions have found that Python is well-suited both as an interactive analysis environment as well as enabling robust systems to be developed often in a fraction of the time it would have taken in Java or C++“(McKinney, 2012, S. 329). Python ist nicht zuletzt prädestiniert für sprachliche Aufgaben durch andere Bibliotheken wie NLTK (Natural Language Toolkit) oder scikit-learn (Machine-Learning-Algorithmen).

Die Struktur dieser Arbeit spiegelt auch die Etappen der Entwicklung des Programms wider: Zunächst sollen in Kapitel 1 bestimmte Rahmenbedingungen und nützliche Begriffe geklärt werden. Dies ist der kürzeste Teil der Arbeit, soll aber durch seine Kürze der Absicht der Arbeit dienen, die Silbe sehr praktisch zu untersuchen. Anders ausgedrückt: Theorie soll nur so weit einfließen, wie es für die Umsetzung und Verständnis des Programms notwendig ist.

Kapitel 2 bietet eine kurze Vorstellung der behandelten Silbenmodelle und stellt ebenfalls die notwendigste Theorie vor, um sie zu verstehen. Zu keinem Zeitpunkt soll ein Modell dem anderen bevorzugt werden, höchstens Umstände thematisiert werden, die eine Implementierung schwierig machen.

In Kapitel 3 soll diese theoretische Grundlage der Silbe mit einer technischen Grundlage der Algorithmen angereichert werden. Hier soll ebenfalls ein wichtige Zwischenstation des Programms erklärt werden, nämlich die Transkription hin zur Lautebene. Da dies für genaue Silbenstatistiken sehr wichtig ist, wurde hier besonders viel Arbeit investiert. Der Verfasser zählt dies ebenfalls als wichtige Leistung der vorliegenden Arbeit. Ausgehend von der „Heimat“ dieser Arbeit (die Linguistik), soll die Code- und Mathematik-Seite in den Hintergrund rücken und eher durch Pseudocode und Abbildungen abstrahiert dargestellt werden.

Das Kapitel 4 zur statistischen Auswertung soll den Leser für einige Probleme sensibilisieren, die mit den Daten verbunden sind. Das ist wichtig, da das Programm prinzipiell jedem ermöglichen soll, eine Auswertung vorzunehmen, die Daten aber tückisch sein können. Gleichzeitig sollen aber einige Vorschläge gemacht werden, um die Probleme zu lösen. Es soll eine Technik Anwendung finden, die sich sehr gut für diese Art von Daten eignet.

Formales

Im Grunde folgt die Arbeit den Konventionen üblicher Arbeiten. Betonte Wörter werden kursiv geschrieben, ebenfalls auch neu eingeführte Konzepte. Da die Software und die Literatur zu einem Großteil Englisch ist, folgt die Arbeit angelsächsischen Konventionen in Bezug auf Zahlen: Dezimaltrenner „.“, Tausendertrennzeichen „,“, wobei der letztere nur bei $\geq$5 Zahlen Anwendung findet.

Der oft gemachte Unterschied zwischen Tokens und Types soll kurz an einem Beispiel veranschaulicht werden: Nehmen wir eine Aufzählung [a,b,a,a,b,c]. In ihr finden sich 6 Tokens, aber nur 3 Types {a,b,c}, gleichzeitig entspricht {a,b,c} dem Set von der Aufzählung. Dieser wichtige Unterschied soll in dieser Arbeit häufig gemacht werden.

Es finden sich größtenteils Beispiele, die nicht in IPA-Umschrift sind. Sie wird nur verwendet, wenn tatsächlich ein für Silbengrenzen wichtiger Unterschied zwischen Graphemen und Lauten herrscht. Der Grund ist einfach: Das Programm arbeitet zu keiner Zeit mit der IPA, sondern mit einer pseudokyrillischen Form (што, мяхкий, чуство), die in eine spezielle Form von SAMPA[1] übertragen wird. Die Argumente dafür werden im Laufe der Arbeit vorgestellt.

[1] http://www.phon.ucl.ac.uk/home/sampa/russian.htm

1. Theoretische Grundlagen zur Silbe und NLP

1.1. Silbentermini

Die grundlegenden Silbenbegriffe sollen hier in Anlehnung an Duanmu (2009, S. 5–10) wiedergegeben werden. Eine Silbe besteht immer aus einem *Nucleus*, im Russischen ist es immer ein Vokal, in anderen Sprachen können Sonore diese Rolle einnehmen wie im Englischen oder Italienischen (Iacoponi und Savy, 2011). Die Konsonanten vor einem Nucleus nennt man *Onsets*, die Konsonanten nach dem Vokal bilden die *Coda*. Coda und Nucleus bilden zusammen die Gruppe namens *Reim* (rime oder rhyme im Englischen), trivialerweise weil dieser Teil für Reime in den Sprachen verantwortlich ist. Einfache Onsets und Codas bestehen aus einem Konsonanten wie in Abbildung 1 (a), komplexe haben mehrere (Abbildung 1(b)). Die ganze Silbe wird mit dem griechischen Buchstaben σ „Sigma“ bezeichnet. Die ersten beiden Strukturen bilden *geschlossene Silben* ab, da die Silbe mit einem Konsonanten endet, Beispiel (c) bildet dagegen eine *offene Silbe* ab. Das Silbenskelett bezieht sich auf die Kategorien Konsonant und Vokal. Erstes Beispiel hat ein Silbenskelett CVC, (b) CCVC, (c) CV.

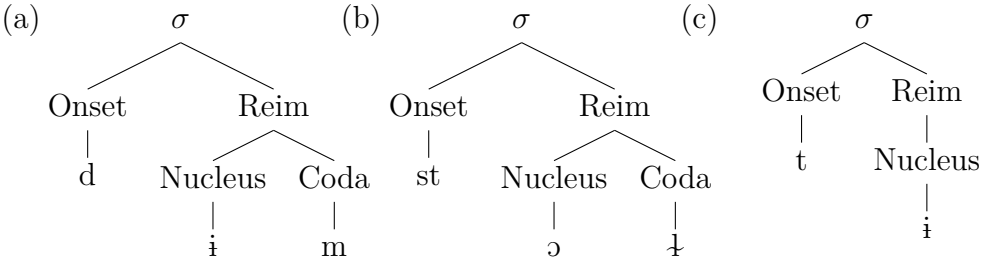

Abbildung 1: Überblick über Silbenstrukturen. Die Wörter sind (a) дым, (b) стол, (c) ты.

In Zusammenhang mit der Silbe werden noch weitere Begriffe erwähnt wie *schwere* und *leichte Silbe*. Die erstere besteht aus einem mehrteiligen Reim wie in Abbildung 1(a) und (b), die letztere besitzt dagegen nur einen Slot im Reim Abbildung 1(c). Eine *Mora* ist jeweils ein Slot im Reim, d.h. (a) und (b) haben zwei Moren. Diese Begriffe werden allerdings keine Rolle in dieser Arbeit einnehmen anders als die oben erwähnten.

Formal werden Grenzen durch einen Punkt an der Silbengrenze markiert wie in хо.ро.ший. Innerhalb von qualitativen Studien ist es üblich mit Baumstrukturen ähnlich wie in Abbildung 1 und Abbildung 2 auf der nächsten Seite zu arbeiten. Wegen dem quantitativen Charakter der Arbeit wird die vereinfachte Variante mit Punkt gewählt.

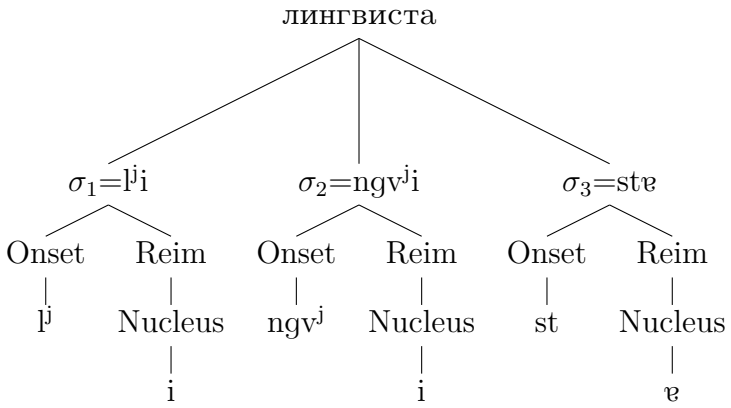

Abbildung 2: Trennung eines kompletten Wortes mit Hilfe der Baumstruktur. Segmentierung nach dem Modell der offenen Silbe.

1.2. Psycholinguistische Experimente

Um die Rolle der Silbe zu illustrieren, sollen hier zwei Arbeiten vorgestellt werden. Die erste Arbeit von Cholin (2011) benutzte ein Priming-Experiment, um den Beginn der Artikulation zu messen. Dabei fand sie mittels englischer und niederländischer Wörter, dass häufigere Silben auch einen schnelleren Beginn der Aussprache vorweisen. Dies ließ sie zum Schluss kommen, dass mental ein internes *syllabary*, d.h. ein Platz, wo die häufigsten Silben aufbewahrt werden, existiert und uns erlaubt, sie schneller zu reaktivieren. Unterschiede wurden allerdings bei dem Zeitpunkt vorgefunden, wann die Aktivierung passiert. Während niederländische Sprecher nur für die erste Silbe eine beschleunigte Aktivierung aufwiesen, wurde dies bei englischen Sprechern in beiden Silben beobachtet. Dieses Ergebnis wurde so interpretiert, dass die letzteren einen größeren prosodischen Fokus hätten.

Kavitskaya und Babyonyshev (2011) verglichen Kinder mit einer Sprachentwicklungsstörung und „normale“ Kinder bei der Aussprache von Pseudo-Wörtern, die einen steigenden Grad von Länge (1-3 Silben) und Silbenkomplexität (CV-CCVCC) aufwiesen und bei denen Cluster nach dem Sonority Sequencing Principle (SSP– genauer nächstes Kapitel) wohlgeformt ([bl,tr,xm]) waren oder nicht ([lb,rt,mx]). Die Studie ergab, dass Kinder mit SLI Probleme hatten, sich komplexe Silben zu merken und sie auszusprechen. So weit wiesen auch russische Kinder ohne Defekt Schwierigkeiten auf; allerdings konnten Probleme bei Kindern mit SLI durch eine Berücksichtigung der Silbenbegriffe Onset und Coda systematisiert werden. Es wurden nämlich mehr Fehler und Vereinfachungen innerhalb der Coda einer Silbe gemacht. SSP spielte bei Fehlern keine signifikante Rolle.

Die vorgestellten Studien zeigen, dass der Begriff der Silbe psycholinguistisch und experimentell vor allem mit neuen Technologien bewiesen werden kann und dies auch statistisch signifikant sein kann. Gleichzeitig sind die Ergebnisse zu sprachspezifisch (Studie 1) oder personenspezifisch (Studie 2). Im nächsten Kapitel unter Abschnitt „Andere Modelle“ werden wir einen interessanten Ansatz

kennenlernen, der wiederum andere Schwächen hat.

1.3. Natural Language Processing (NLP)

Zur einer möglichen Definition von NLP führte Grishman (1989, S. 1) negativ an: “To attempt a unified survey of a field where there is sharp disagreement even about basic approaches may seem foolhardy“. Erschwerend für die Definition kommt hinzu nach Ooi (1998, S. 23), dass Computer in allen Bereichen eine Rolle einnehmen und überspitzt die ganze Linguistik als Computerlinguistik bezeichnet werden kann. Er führt auch an, dass in der Literatur häufig NLP mit Computerlinguistik als quasi-identisch verwendet wird. Ich selbst bin nicht ganz unzufrieden mit einem eher weiten Verständnis des Begriffes, denn es erlaubt die Benutzung verschiedener Techniken aus spezifischeren Bereichen wie Korpuslinguistik, Computerlexikographie und mathematischer Linguistik und Makrobereichen wie Data Science und Computer Science (Informatik).

In NLP muss man zwei Richtungen unterscheiden, die Texte auf ihre Weise anschauen (und analysieren). Eine Richtung ist der auf Regeln und Grammatik beruhende, also ein „wissenbasierter“ Ansatz, der u.a. in Allen (1995) propagiert wird. Der andere Ansatz beruht auf Statistik und Daten – ist insofern „datenbasiert“ wie bei Manning und Schütze (1999) als Standardbuch beschrieben. In neuerer Zeit hat sich kein Ansatz wirklich durchgesetzt. Bird u. a. (2009) benutzen beide Ansätze und erreichen mit beiden hohe Richtigkeitsquoten bei Wortartenbestimmung. Interessant ist zum Beispiel eine Kombination von einem statistischen *Hidden Markov Model* (HMM)-Tagger, der die grundlegenden Wortarten wie Adjektiv und Substantiv bestimmt. Danach kann man ein *Context Free Grammar* (CFG) anwenden, um *Chunks* wie NP, VP, PP’s zu bestimmen. Statistische Methoden eignen sich zusammengefasst sehr gut dafür, ein finites Set an *Phasen* („states“ in der HMM-Terminologie) zu schaffen, der dann mit Regeln weiter bearbeitet werden kann. Schließlich kommt es bei der Verfügbarkeit von einem Ansatz aber auf die Verfügbarkeit von *Daten* an.

Die vorliegende Arbeit verwendet zwar beide Ansätze, aber der Schwerpunkt liegt eindeutig auf dem wissensbasierten, da man fürs Russische größtenteils keine statistischen Ansätze kennt, um Silben zu segmentieren. Das Silbenmodell 4 ist zur Hälfte eine Ausnahme.

2. Silbenmodelle

Die Gemeinsamkeiten der Silbenmodelle sollen hier bereits aufgezählt werden, um später nur die Unterschiede behandeln zu können. Sie beziehen sich vorrangig nur auf die Modelle 1-3:

1. Einzelne Konsonanten zwischen Vokalen, wie in до**р**о**г**а werden immer zur nächsten Silbe gerechnet: до.ро.га

2. /j/ zwischen Vokalen verhält sich wie ein Konsonant und wird in Fällen wie ма**й**ор immer zur nächsten Silbe gerechnet: ма.йор

3. Gleiches gilt auch für das grafisch implizite /j/ durch die jotierten Grapheme <я, ю, ё, е> ча**я** → ча.я [t͡ʃʲa.jɐ]

4. /j/ nach einem Vokal und vor einem Konsonanten gehört immer zur vorhergehenden Silbe: ча**й**ник → чай.ник

Wie man daran sieht, gibt es keine Unterschiede, wenn zwischen Vokalen nur ein Konsonant steht oder (formalisiert) eine VCVC-Struktur zu beobachten ist. Die eher universellen Modelle 4 & 5 stimmen in diesen Gemeinsamkeiten ebenfalls überein, allerdings implizit.

Die Theorien unterscheiden sich dann, wenn sich mehrere Konsonanten zwischen Vokalen befinden. Als motivierendes Beispiel kann man sich Segmentierungen in Tabelle 1 anschauen. Es sind 5 von 364 Wörtern aufgelistet, die jedes Modell anders segmentiert hat (Begriff für später: Set size (Ssi) von 5). Es lassen sich sicherlich bereits Muster finden, die die Modelle unterscheiden. Nach dem Durchlesen der folgenden Beschreibungen wird die Tabelle verständlicher.

Tabelle 1: Die 5 Silbenmodelle im Vergleich. Daten aus vorliegendem 1MK, segmentiert durch das Programm: (a) в*арварству, (b) арт*истку ,(c) лингв*иста, (d) д*окторской, (e) *орденские.

W	S1	S2	S3	S4	S5
(a)	[v'ar.və.rstvʊ]	[v'ar.vər.stvʊ]	[v'a.rvə.rstvʊ]	[v'a.rvər.stvʊ]	[v'ar.vərs.tvʊ]
(b)	[ɐ.rtʲ'is.tkʊ]	[ɐr.tʲ'i.stkʊ]	[ɐ.rtʲ'i.stkʊ]	[ɐ.rtʲ'ist.kʊ]	[ɐr.tʲ'is.tkʊ]
(c)	[lʲi.ngvʲ'is.tɐ]	[lʲin.gvʲ'i.stɐ]	[lʲi.ngvʲ'i.stɐ]	[lʲing.vʲ'i.stɐ]	[lʲin.gvʲ'is.tɐ]
(d)	[d'ɔk.tə.rskəj]	[d'ɔ.ktər.skəj]	[d'ɔ.ktə.rskəj]	[d'ɔk.tər.skəj]	[d'ɔ.ktərs.kəj]
(e)	['ɔr.dʲi.nskʲi.ji]	['ɔr.dʲin.skʲi.ji]	['ɔ.rdʲi.nskʲi.ji]	['ɔ.rdʲin.skʲi.ji]	['ɔr.dʲins.kʲi.ji]

2.1. Silbenmodell 1 (S1)

Dieser Ansatz wurde von Vinogradov u. a. (1953) und Ščerba (1983) vertreten. Außerdem ist S1 in Matusevič (1976) als Teil eines Lehrbuches verwendet worden. Das Silbenmodell hängt stark von der Wortbetonung ab:

1. Wenn man eine innere Silbenstruktur V'CCV hat, dann gehört der erste Konsonant des inneren Clusters zur betonten (vorherigen) Silbe wie in поч.та

2. Dies ist allerdings *nicht* der Fall, wenn der zweite Konsonant im Cluster nach dem betonten Vokal ein /r/, /l/ oder /j/ ist. In diesem Fall gehört der komplette Cluster zur nachfolgenden Silbe, so in му.дрый, ка.пля, ко.пья.

3. Im Falle von einem VCCV' wird der gesamte Cluster Onset der nächsten Silbe (mit dem betonten Vokal).

4. Bei mehr als 2 Konsonanten im Cluster wie z.B. VCCC+V gelten die Regeln 1 und 3, Regel 2 hat dann *keine* Gültigkeit.

Die angegebenen Regeln resultieren aus der Annahme Ščerbas, dass die muskuläre Anspannung (мускульная напряженность) bei der Artikulation der entscheidende Faktor bei der Silbentrennung ist. Silben sind nach ansteigen dieser Spannung strukturiert. Vokale bilden nach seiner Theorie die Höhepunkte dieser Anspannung, daher sind sie auch der Kern der Silbe. Die Anspannung bei Konsonanten hängt dagegen von der Position zum betonten Vokal ab. Ein Konsonant V'**C**CV hat eine stärkere Anspannung als der nachfolgende und gehört deswegen zur Coda der vorherigen Silbe: V'**C**.CV. Wie in den Regeln beschrieben dehnt sich dieses Modell auch auf größere Cluster aus.

2.2. S2

Dieses Silbenmodell geht auf Avanesov (1956) zurück, wurde aber auch in neuen Ansätzen von Kasatkin (2001) vertreten und bei Wade (1993) und Bolla u. a. (1968) in die Sprachlehre integriert. Es gibt im Prinzip nur drei Regeln und zwar unabhängig von der Größe des Clusters (sofern aber $\geq$ 2 Konsonanten):

1. Im Fall, dass in einem Konsonantencluster der erste Konsonant ein Sonor ist, dann gehört er immer zur vorhergehenden Silbe.

2. In Fällen, wo in einem Konsonantencluster alle Konsonanten Sonore sind, gehört der Cluster immer zur nachfolgenden Silbe wie во.лна.

3. In allen anderen Fällen gehört der Cluster komplett zur nächsten Silbe.

4. Eine Ausnahme: Sofern ein Cluster aus [rʃ]+Plosiv+Vokal besteht, dann setzt man die Grenze nach [ʃ] wie in [zʁ.dʲ'ɛrʃ.kʁ]

Dieser Ansatz berücksichtigt das Sonoritätsprinzip. Er unterteilt allerdings das gesamte Phonemsystem nur in drei Sonoritätsgruppen (hier nach der Stärke der Sonorität): (1) Vokale (2) Sonore (3) Plosive und Frikative. Dabei wird, wenn

man die oben aufgestellten Regeln in Betracht zieht, ein Anstieg der Sonorität im Onset bevorzugt, da man den Sonor zur Coda der vorherigen Silbe rechnet, wenn Regel 1 erfüllt ist. Ansonsten wird eine offene Silbe bevorzugt.

2.3. S3

Die S3 ist das wohl phonetisch fundierteste Modell. Argumentiert von Bondarko (1998) und mit einem Schwerpunkt in St. Petersburg, ist es dem Modell von Ščerba allerdings nicht sehr ähnlich. Die einzige wirkliche Regel ist:

1. Alle intervokalischen Konsonantencluster bilden mit dem jeweils nachfolgenden Vokal eine Silbe.

Kurz ausgedrückt: Nach S3 führen intervokalische Konsonantencluster immer zur Öffnung der vorherigen Silbe.

Die beiden ersten Modelle sind von Bondarko in Spektrogrammen gegenübergestellt worden und es wurde von ihr weder ein Unterschied gefunden zwischen S1 (z.B. пап.ка) und S2 (дым.ка) noch überhaupt einen phonetischer Hinweis auf eine geschlossene Silbe. In anderen Wörtern wie марку gab es phonetische Anzeichen, dass der Cluster Anzeichen einer Labialisierung (ма(рку)) des folgenden Vokals trug. Weder die Betontheit des ersten Vokals nach S1 noch ein Sonoritätsprinzip nach S2 hatten einen Einfluss auf diese Labialisierung.

2.4. S4

Als viertes Modell soll hier die Theorie von Pulgram (1970) angeführt werden. Es soll wiederholt werden, dass die oben erwähnten Gemeinsamkeiten nicht *explizit* bei diesem Modell zutreffen. Denn: Das Interessante an diesem Modell ist, dass es nicht sprachspezifisch ist, sondern eine gewisse Universalität für sich beansprucht. Außerdem ist der Ansatz deswegen so interessant, weil die qualitativen Methoden der Silbentrennung mit quantitativen ergänzt wurden (Kelih, 2012, S. 58). Zuerst die qualitativen Regeln in dieser Reihenfolge nach Pulgram (1970, S. 48–50):

1. Als Grundregel wird das Gesetz der offenen Silbe verstanden. Als erster Schritt werden also alle intervokalischen Konsonantencluster zu Onsets der nachfolgenden Silbe: бо.мба

2. Dann wird analysiert, ob der entstehende Onset phonotaktisch erlaubt ist, d.h. in einem Set aus allen konsonantischen Wortanfängen zu finden ist. In Falle von бо**мб**а trifft dies nicht zu. Deswegen wird vom Onset ein Konsonant weggenommen und der Coda hinzugefügt.

3. Die Regel 2 wird wiederholt bis der Onset phonotaktisch erlaubt ist.

4. Es kann dabei zu einem Konflikt kommen, nämlich, dass man durch die Verschiebung der Konsonanten ein phonotaktisch inakzeptables Coda bekommt. Dann wird eher ein inakzeptables Coda geduldet als ein inakzeptabler Onset.

In Bezug auf die slavischen Sprachen wurde dieser Algorithmus um quantitative Aspekte ergänzt und zwar zuerst von Lehfeldt (1971) fürs Ukrainische. Kempgen (2003) hat diesen Algorithmus auf das Russische übertragen. Seine Tabellen sind in S4 implementiert. Er hat die Tabellen anhand von Wörterbüchern erstellt und dabei die Frequenz von Clustern statistisch untersucht. Er stuft alle vorkommenden Cluster in „reguläre“ und „marginale“ (ebd., S. 199) ein. Die ersteren haben eine höhere Wahrscheinlichkeit zusammen vorzukommen als getrennt in der jeweiligen Position. Reguläre Cluster können im Onset stehen, marginale nicht. So werden auch Cluster so lange getrennt bis ihr Status regulär wird.

Die Theorie ist im Gegensatz zu S3 auf der phonologischen (genauer: phonotaktischen) Grundlage entwickelt worden. Interessant ist aber: Kempgens Ergebnisse für 12 ausgewählte Wörter ergaben 5 Übereinstimmungen für S1 und 7 für S2.

2.5. S5

Als letztes Modell soll das Sonority Sequencing Principle (SSP) implementiert werden. Es sagt aus, dass die Sonorität einer Silbe bis zum Vokal als dem Höhepunkt ansteigt. Dies äußert sich übersprachlich in der am häufigsten auftretenden Silbenstruktur CV. Beispielhafte Wörter im Russischen für die SSP sind: травма, жмот, дразнить.

An sich ist die SSP nichts neues. Es ist bekannt, dass die Phonotaktik der romanischen Sprachen sehr von SSP abhängt (Iacoponi und Savy, 2011). Bei germanischen und slavischen Sprachen ist dies nicht unbedingt der Fall. Obwohl die SSP für die diachrone Entwicklung des Russische sehr wichtig ist, gibt es trotzdem Silben, die eindeutig dagegen verstoßen,z.B. мгла, льстить, мстить. Nichtsdestotrotz haben Goldwater und Johnson (2003) herausgefunden, dass bei Anwendung der SSP für Deutsch und Englisch eine Übereinstimmung von 86 bzw. 87% mit sprachspezifischen Modellen zu beobachten ist. Die vorliegende Arbeit benutzt eine sehr verbreitete Hierarchie: stimmlose Plosive < stimmlose Frikative < stimmhafte Plosive < stimmhafte Frikative < Sonore < Approximanten < Vokale.

In der SSP gibt es aber auch einige Fälle, die nicht durch bloßen Sonoritätsabgleich gelöst werden können. Beispielsweise wenn Konsonanten einen Cluster bilden, die die gleiche Sonorität haben: кно**пк**а,те**мн**ый. In diesem Fall haben Goldwater und Johnson (ebd.) die SSP mit der Regel des Maximum Onsets kombiniert. Die erwähnten Fälle würden danach also gelöst werden, indem man den gesamten Cluster zur nächsten Silbe zählt. Ähnlich funktioniert auch die Implementierung in dieser Arbeit. Der Algorithmus weist zuerst allen Konsonanten

in einem Cluster Sonoritätswerte[2] zu und findet dann das Minimum und setzt vor dieses Minimum die Silbengrenze: о**стр**ый[str] → [519] → [5.19] → [ˈɔs.trɨj]; те**мн**ый [mn] → [77] → [.77] → [tʲɔ.mnɨj]; зе**мств**о [mstv] → [7516] → [75.16] → [zʲɛms.tvɐ].

2.6. Andere Modelle

Bestimmte andere Ansätze haben interessante Vorschläge bzw. Resultate erbracht. Als ein Beispiel kann man hier Coté und Kharlamov (2011) anbringen, die in ihrer Arbeit psycholinguistische Experimente zur Silbentrennung durchgeführt haben. Die Resultate sind auch deswegen spannend, weil sie den Modellen oben teilweise komplett widersprechen. In einer deutlichen Mehrheit der Fälle (92-98%) wurde der Cluster VCCV in VC.CV getrennt – das unabhängig von der Betonung und Zusammensetzung. Dies widerspricht komplett S3 und teilweise S2 und S1. VCVC sind dagegen in einer Mehrheit der Fälle (72-88%) in V.CV.C segmentiert worden; man bemerke aber den relativ niedrigen Zustimmungswert auch im Hinblick darauf, dass für S1-5 dies unter Punkt 1 auf Seite 7 eine unumstrittene Gemeinsamkeit ist.

Das sind die neuen Erkenntnisse aus Arbeit. Einige kritische Momente müssen aber hier angemerkt werden, die eine Verwendung als Silbenmodell ausschließen:

- Es wurden nur zweisilbige Wörter (=mit zwei Nucleii) verwendet.
- Der mittlere Cluster bestand entweder nur aus einem oder zwei Konsonanten.
- Unstimmigkeit innerhalb der Methoden: *Pause Insertion, 1st-syllable repetion, 2nd-syllable repetion, Slash insertion* (ebd., S. 281–282).
- Es wurden nur Pseudowörter segmentiert wie п*аксул, г*узун.
- Es gab keine verdoppelten Konsonanten.

Außerdem haben Kalnyn' und Maslennikova (1985) Daten vorgelegt. Sie ziehen aber einen deutlichen Unterschied zwischen Nordrussischen und Südrussischen Dialekten. Während die ersteren in allen Fällen eine VC.CV–Trennung bevorzugen würden, trennen die letzteren V.CCV. Da wir eine derartige Dialekt-Unterscheidung hier nicht machen, soll es nur bei dieser Erwähnung belassen werden.

Ein anderes Modell ist die Optimality Theory, die hier nicht dargelegt wird. Es ist eine große phonologische Theorie für sich, die z.z „in der Slawistik wenig rezipiert wird“ (Kelih, 2012, S. 7). Ein durchdachtes Makromodell existiert daher für das Russische nicht.[3]

[2] Ein ähnliches Modell auch http://sylli.sourceforge.net/sh.html

[3] Ein OT-Ansatz wurde von Knjazev (1999) entwickelt. Allerdings gibt es dort, wie (Kelih,

3. Material und Methoden

3.1. Der Korpus

3.1.1. Korpuswahl

Für das Programm an sich ist kein spezielles Material erforderlich. Aber: Da das Programm in dieser Untersuchung zuerst den Schritt zur phonetischen Transkription macht, ist zumindest ein Text erforderlich, bei dem die Betonungen gesetzt sind. Dies ist entscheidend für die richtige Transkription der Vokale. Konsonanten dagegen können auch ohne Betonung durch einfache Inferenz in Lautezeichen umgewandelt werden. Ein weiterer Grund für die Betonung ist die in Abschnitt 2.1 erwähnte Setzung der Silbengrenzen abhängig vom betonten Vokal.

Der Uppsala Corpus[4] ist sicherlich der ausgeglichenste frei zugängliche Korpus in der Russistik und bei Korpuslinguisten immer noch im Gebrauch. Aber aus einigen Gründen passt er hier nicht, (1) es sind keine Betonungen gesetzt, (2) die Metadaten sind nicht digital zugänglich, (3) der Text wurde nicht lemmatisiert, (4) keine Metadaten zu den Tokens, (5) eine Sammlung relativ alter Texte.

Die Erstellung eines eigenen Korpus kam dabei im begrenzten Rahmen der vorliegenden Arbeit nicht in Frage. Vor allem auch daher nicht, weil vom Russischen National-Korpus eine 1-Million Version[5] (im Folgenden: 1MK) zugänglich ist, in der Betonungen und Textarten schon bestimmt sind und als kleines, aber wichtiges Detail auch das Graphem <ë>.[6] Zum großen Russischen National-Korpus (240 Mio.) wurden bisher keine Silbenstatistiken durchgeführt, nur sind jeweils graphemische Uni- und Bigramme analysiert worden – ohne Bezug zu Silben.[7] Der Leser könnte sich nach dem Anschauen der im nächsten Unterabschnitt vorgestellten Makrodaten vielleicht schon die Frage beantworten, ob der 1MK repräsentativ ist. Obgleich hier tatsächlich eher eine negative Antwort kommen könnte, hat der Korpus ansonsten Parameter, die sich sehr gut für NLP-Aufgaben eignen: Denn die grammatische Homonymie ist gelöst, das Format ist das sehr zugängliche .xhtml (gültiges XML und HTML zugleich) und die modernen Texte können einen Klassifikator oder Parser besser auf die Herausforderungen moderner Sprache vorbereiten.

2012) richtig anmerkt, einige Unstimmigkeiten, z.B. zur Rangierung der Regeln sowie zur Prozedur an sich. Ein Teil, nämlich die SSP, die bei Knjazev Regel 1 ist, wurde unter S5 implementiert.

[4] http://www.moderna.uu.se/slaviska/ryska/corpus/abstract/

[5] http://www.ruscorpora.ru/corpora-usage.html

[6] Aus Lizenzgründen kann die Version weder auf github noch auf einem physischen Datenträger veröffentlicht werden. Um also die Experimente selbst zu machen, muss sich der Leser eine eigene Kopie verschaffen.

[7] http://dict.ruslang.ru/freq.php

3.1.2. Korpustatistiken

Der 1MK teilt sich in ein Standard- und Spoken-Subkorpus. Der erstere besteht zum größten Teil aus Zeitungsartikeln und einiger Belletristik (20-25%), während der Spoken-Subkorpus aus einigen Transkriptionen und mehrheitlich Texten für Reden besteht. Da die ruscorpora-Stiftung keine expliziten Statistiken zum 1MK veröffentlicht hat, soll hier ein kurzer Überblick darüber gegeben werden. Manche der Resultate sollen später eine größere Rolle spielen. Die wichtigsten Makroparameter aus der Tabelle 2 sind[8]:

Tokens Im Korpus sind tatsächlich etwas mehr als 1 Million Tokens, wobei es zwischen den Texten sehr große Unterschiede gibt wie man an der hohen Standardabweichung sowie am sehr großen Unterschied zwischen Mittelwert und Median beobachten kann.

Ca. 75% des Korpus werden durch Standard-Texte eingenommen. Die anderen 250,000 Tokens werden durch Spoken-Texte repräsentiert. Zwischen den Zahlen der beiden Gruppen sind sehr große Unterschiede. Nennenswert ist hier vor allem der Median und der Mittelwert. Es ist festzustellen, dass standard-Texte kleiner und ausgeglichener sind wie an den Quartilen zu sehen.

Textanzahl Insgesamt sind es 532 Dateien/Texte. Unterteilt in die beiden Unterkategorien *Standard* und *Spoken* sind es 467 und 65 respektive. Gleichzeitig ist ein Missverhältnis zu beobachten, denn 12% der Texte nehmen 25% der Tokensanzahl ein.

Jahrespanne Der früheste Text ist aus dem Jahr 1973. Man kann aber an dem relativ hohen Mittelwert und Median sehen, dass im Korpus eher aktuelle Texte zu finden sind. Bei der Betrachtung der Spoken-Parameter sieht man, dass nicht alle der 65 Texte eine Jahreszahl in den Metadaten haben. Das Drittel *mit* der Jahresangabe hat eine unebene Streuung wie man am Unterschied zwischen Mittelwert und Median feststellen kann. Die Streuung an sich ist ebenfalls größer als in Standard-Texten.

Erwähnenswert ist, dass den 1 Mio. Tokens ein Set von 124,000 Types entspricht. Der Leser sollte für die folgenden Kapitel die Erkenntnis mitnehmen, dass der 1MK in zwei Teile zerfällt, die sich quantitativ sehr unterscheiden. Der standard-Teil hat ergiebigere Metadaten , mehr Tokens, ausgeglichenere Texte und eine moderne Sprache.

[8] Dies ist nur ein kleiner Ausschnitt der möglichen Statistiken und Darstellungen. Die Metadaten geben noch reicheren Aufschluss über die Texte. Histogramme mit den Tokensverteilungen sowie weitere Genretrennungen sind wiederum eine Arbeit für sich und sollen hier nicht weiter verfolgt werden.

Tabelle 2: Übersicht über die Korpusparameter *Jahr (J)* und *Tokens (Tok)*. Eingebunden sind der ganze Korpus (1MK) und die Subkorpora standard (ST) und spoken (SK).

	Par	$\sum$	N	$\overline{x}$	Δ	Min	Q_1	Q_2	Q_3	Max
1MK	Tok	1,003,612	532	1886	2747	30	342	832	2079	21,825
	J	-	485	2001	3.7	1973	2002	2003	2003	2005
SK	Tok	251,230	65	3865	3269	30	897	2911	6052	12,885
	J	-	20	1995	10	1978	1993	2002	2003	2003
ST	Tok	752,382	467	1611	2552	72	330	736	1652	21,825
	J	-	465	2001	3	1973	2002	2003	2003	2005

Wichtig ist aber zu sagen, dass sich alle Werte nur auf vorhandene Grapheme beziehen, für den phonetischen Teil wird in Abschnitt 3.5 eine Filterung durchgeführt, die aber auf den vorgestellten Parametern basiert.

3.2. Der phonetische Algorithmus

3.2.1. Schwierigkeiten

Bevor das Programm die Silbentrennung durchführt, transkribiert es zuerst die graphischen Wörter in ihre phonetische Form. Dies ist für die Modelle notwendig, da sie alle auf phonetischen bzw. phonologischen Eigenschaften aufbauen. Zum einen muss man in die phonetische Transkription gehen, weil man das implizite /j/ aus den jotierten Vokalen extrahieren muss. Auch muss man die Palatalisierung aus den Vokalen in ein phonetisches Zeichen verwandeln, weil z.B. der Algorithmus von S4 zwischen palatalisierten und velarisierten Konsonanten unterscheidet. Zum anderen beeinflusst die regressive Assimilation und die Auslautentstimmlichung das Aussehen von Konsonantenclustern.

Russisch bietet dabei eine gute Grundlage für einen Transfer des Wortes auf die phonetische Ebene, denn für Vokale lässt sich durch Referenz auf die betonte Stelle und/oder den vorhergehenden Konsonanten die richtige Form finden. Konsonanten können demgegenüber durch Referenz auf den folgenden (graphischen) Vokal oder Konsonanten transkribiert werden.

Dies trifft auf eine Mehrheit der Fälle zu. Im Russischen kann man aber nicht immer die phonetische Form aus der graphischen ableiten. Wörter wie ар**т**ерия, мя**г**кий, **ч**то, коне**ч**но haben Elemente, die nicht vorraussagbar sind. Solche Abweichungen sind im Russischen in orthoepischen Wörterbüchern fixiert wie bei Ivanova (2004), Ageenko und Zarva (1993), Rezničenko (2005) und Krukover (2008). Für die vorliegende Arbeit sind folgende Fälle wichtig:

- Graphem <e>, das nicht immer die Palatalisierung anzeigt: антенна, дельта, гротеск ...

- Verdoppelte Grapheme wie <нн,лл> wie in бриллианты, ванная, террор, гетто, лобби
- Phonetische Vereinfachungen wie сердце [sʲɛrt͡sə], солнце [sʼɔnt͡sə],
- Assimilationen wie мягкий [mʲʼaxkʲij], лёгкий [lʲʼɔxkʲij], конечно [kɐnʲʼɛʃnə]
- Palatalisationsassimilierung wie вести [sʲtʲ], истинный [sʲtʲ], утонченный [nʲt͡ʃʲ]

3.2.2. Lösungen

Man muss sich größtenteils keine neuen Algorithmen ausdenken, da Itapov und Stepanova (1997) bereits ein Programm zur automatischen Transkription geschrieben haben. Es ist nicht verfügbar und wäre auch mit der vorliegenden Programmiersprache nicht vereinbar, aber der Algorithmus ist klar beschrieben und basiert auf formalen Elementen und durch Experimente gewonnene Daten. Das Endstadium des Algorithmus wurde von der problematischen kyrillisch-phonetischen Transkription auf echte Sampa (und Pysampa) übertragen.

Die Formalisierung des Algorithmus ist relativ strikt; ein besserer Ansatz, der in der vorliegenden Arbeit teilweise verwendet wird, ist die Benutzung von orthoepischen Wörterbüchern. Sie können Probleme der Graphem-Phon-Übertragung generell gut lösen, denn die systematische Art erlaubt es, Wörter in die richtige Form zu setzen. Dies stellt allerdings die Bedingung, dass der Text lemmatisiert ist wie im Falle von 1MK. Denn Wörterbücher benutzen das Lemma als Index wie in Abbildung 3 zu sehen ist. Man kann also den Lokativ антенне nicht im Wörterbuch finden, sondern nur das Lemma антенна und muss die Operation daher formalisieren.[9] Die verwendeten Wörterbücher von Rezničenko und Krukover sind nicht die neuesten oder die größten[10], aber als einzige in digitaler Form verfügbar gewesen und damit für die vorliegenden NLP-Aufgaben geeignet.[11]

Der Algorithmus von Itapov gibt keinen Hinweise auf den Umgang mit den velarisierten Konsonanten vor <e>, daher wird ein Wörterbuch verwendet, das aus den digitalen Fassungen von Rezničenko (2005) und Krukover (2008)[12] zusammengesetzt wurde und die Änderungen für die Programmiersprache formalisiert wurden (Abbildung 3 rechts). Alle unbekannten Wörter werden der Graphie nach transkribiert.

[9] Es ist natürlich möglich Wörterbuch mit allen deklinierten Formen zu schaffen. Es ist allerdings nicht sehr elegant, da man mit relativ vielen (>100,000) Wörtern arbeiten muss.

[10] Kalenčuk u. a. (2012) scheint vom Umfang und Jahr her ein ideales Buch zu sein, ist aber nicht digital verfügbar und besitzt nach Erfahrungsberichten starke editorische Mängel (kein Anfangsbuchstabe й, gekürzt beim и und э)

[11] Trotzdem muss angemerkt werden, dass es für die russische Phonologie aktuell keine *machine-readable-dictionaries* (mrd's) gibt; alle derzeit vorhandenen Wörterbücher orientieren sich an Menschen, was NLP in russischer Phonologie nur durch Improvisation ermöglicht wie in dieser Arbeit.

[12] http://dazor.narod.ru/russkie/slovari/krukover/krukover-all.htm

антаблеме́нт [*м'е́*] ! *неправ.* антамблеме́нт
антве́рпенцы, -ев, *ед.* -нец [*в'е́*] и [*вэ́*], [*рп'*] ! *произн.* [*р'п'*] *устар.* // *для выраж. знач. жен. употр. соч.* жительница Антве́рпена *и др.*
анте́нна [*тэ́*]; анте́[*нн*]а *и* анте́[*н*]а, *см. прим. 2, 3*
а̀нтиглобали́ст ! *неправ. произн. ф. им. ед.* а̀нтиглобали́с *без* [*т*]
□ *Употр. по отнош. к лицам муж. и жен. пола*

анестезиолог,ане-»анэ — сте-»стэ
анестезия,ане-»анэ — сте-»стэ
антенна,нте-»нтэ
антисептика,исе-»исэ
антитеза,ите-»итэ
антитезис,ите-»итэ
антитетический,ите-»итэ

Abbildung 3: Links: Beispiel eines Eintrages aus Rezničenko (2005, S. 33). Rechts: Einige Formalisierungen für die Anwendung des Wörterbuches.

Bei der Palatalisationsassimilierung wird nach dem eingangs erwähnten Algorithmus vorgegangen, der dies nur zulässt bei /sʲtʲ/, /zʲdʲ/, /nʲt͡ʃʲ/, /nʲtʲ/, /nʲsʲ/, /nʲzʲ/.

Doppelte Konsonanten bedeuten, dass sie lang ausgesprochen. Damit bilden sie für Silbentrennung zwei getrennte Konsonanten und werden dementsprechend getrennt oder nicht: Nach Vinogradov u. a. (1953) Анна → Ан-на, S5 → А-нна. Um nach Itapov und co. zu entscheiden, ob die verdoppelten Grapheme auf phonetischer Ebene zwei Konsonanten bilden, gibt es einen Entscheidungsbaum, der Kriterien wie Betonung, Wortende oder -anfang oder, ob nachfolgendes Phonem ein Konsonant oder Vokal ist, einbezieht. Die eigene Implementierung wurde im Pseudocode-Beispiel in Algorithmus 1 dargestellt. Darin entscheidet zuerst die Position; sofern der Cluster in der Mitte des Wortes ist, gelten Regeln je nach Gruppe. Beispiele für Anwendung von ihm: балл → 1, русский → 1, суббота → 1, сумма → 2, аккорд → 1, рассада → 2.

Algorithm 1 Doppelter Konsonant (DK) – Pseudocode-Lösung

```
if DK am Wortanfang then
    Doppelter Konsonant
else if DK am Wortende then
    Ein Konsonant
else if DK in Wortmitte then
    if Anderer Konsonant nach DK (C1C1C2) then
        Ein Konsonant
    else if DK besteht aus <б,в,ф,р,л,г> then
        Ein Konsonant
    else if DK besteht aus <ж,т,д,з,с> then
        Zwei Konsonanten
    else if Vokal vor DK betont and DK besteht aus <к,н,м,п> then
        Zwei Konsonanten
    else
        Ein Konsonant
    end if
end if
```

3.3. Theoretische Grundlage zur Silbentrennung

Wie Jurafsky und Martin (2008, S. 406) beschreiben, gibt es nicht nur mehrere Möglichkeiten wie man Silben segmentiert (unser Abschnitt 2 auf Seite 7), sondern auch mehrere Arten, wie man dies automatisch tun könnte:

> One reason syllabification is a difficult computational task is that there is no completely agreed-upon definition of syllable boundaries. Different on-line syllabified dictionaries (such as the CMU and the CELEX lexicons) sometimes choose different syllabifications. [...]
>
> Like much work in speech and language processing, syllabifiers can be based on hand-written rules, or on machine learning from hand-labeled training sets.

Zumindest die erste Schwierigkeit umgeht diese Arbeit, indem es 5 Modelle implementiert und dem Forscher die Wahl überlässt. Als Beispiel für Silbentrennung nennen sie die bereits erwähnten Goldwater und Johnson (2003), die mit den Prinzipien Maximum Onset und SSP (Sonority Sequencing Principle) Regeln konstruieren, die über einzelne Sprachen hinweg verwendbar sind und mit diesem Ansatz eine sehr hohe Übereinstimmung mit sprachspezifischen Modellen erreichten. Für Sprachen wie Italienisch, die dem SSP folgen, kann man in den neueren Ansätzen eine Genauigkeit von 100% erreichen wie bei Iacoponi und Savy (2011).

Jurafsky und Martin (2008, S. 407) stellen des Weiteren heraus, dass es drei Schritte zur Bestimmung einer Silbe gibt: „Rule 1 forms nuclei at each syllabic segment, Rule 2a attaches onset consonants to the nucleus, and Rule 2b attaches coda consonants.“ Dies ist auch die Reihenfolge, die S1-5 machen. Es wird zuerst eine Silbe mit einem Nucleus gebildet, dann wird der gesamte Cluster provisorisch zu ihr hinzugefügt. Am Ende werden die Silben ausbalanciert, nämlich so viel vom Onset abgegeben bis er (und die Coda der vorherigen Silbe) der jeweiligen Regel entspricht. Dies macht auch rein logisch Sinn: Wie Bondarko (1998, S. 212), ohne allerdings das Modell zu nennen, zeigt, hat das Russische 54% offene CV Silben. Natürlich ist es nur ein Modell und die Zahlen können sich von Korpus zu Korpus ändern, doch die Tendenz zur einfachen und offenen Silbe spiegelt sich nicht nur durch die Intuition, sondern auch in diesen Zahlen wider.

Machine learning kommt für einen Silbenparser nicht in Frage, da es keinen großen (notwendig zumindest 1-2 Mio.), manuell getrennten Silbenkorpus fürs Russische gibt. Außerdem sind S1-3 eine Sammlung von manuellen Regeln und nicht statistischen. Als einzige (quasi) statistische Implementierung kann man nur die S4 betrachten, wenngleich es ein sehr simpler Mechanismus ist. Quasi und simpel deswegen, weil Kempgen keinen in Silben getrennten Text verwendet hat, sondern nur den konsonantischen Wortbeginn und das Wortende von Lemmata aus einem Wörterbuch.

3.4. Programm-Mechanik

Der russische Text wird zuerst in pseudo-russische Wörter (лёхкий, серце) übersetzt, da auch orthoepische Wörterbücher, auf denen das Programm teilweise basiert, so vorgehen. Danach wird das Wort in eine *modifizierte* SAMPA-Kodierung (im Folgenden: Pysampa) übersetzt. Die Betonung auf Pysampa soll anzeigen, dass SAMPA, obwohl immer noch maschinenfreundlicher als die IPA, für NLP nicht perfekt ist. Intern, um bestimmte Stärken von der zu Grunde liegenden Code-Sprache zu nutzen, werden alle SAMPA-Zeichen, die aus zwei Graphemen bestehen (ч → SAMPA: tS' → Pysampa: 4 oder ц → SAMPA: ts → Pysampa: 3) zu unikalen Zeichen. Ansonsten wird bei der Iterierung der Eindruck erweckt, dass Affrikate oder palatalisierte Konsonanten aus zwei unabhängigen Zeichen bestehen. Durch Zusammenfassen zu einem Zeichen entgeht man diesem Problem. Alle Zeichen haben überdies die grundlegende Eigenschaft im ASCII-Zeichensatz zu sein und damit Probleme bei Konversionen zu umgehen.[13] Die Umwandlung für Grapheme ist im Anhang C beschrieben.

Das Programm wurde stark für große Korpora optimiert, hauptsächlich durch einen *dynamic programming* Algorithmus auf Grundlage von Sets. Diese Optimierung lässt sich in Zahlen festhalten. Das Verarbeiten des 1MK dauert 480 Sekunden auf einem älteren Computer (Dual Core mit 2.4 GHz), der Testtext „Интересно придумала“ mit 179 Tokens braucht 1.4 Sekunden, pro Token wird der Unterschied klarer: 1MK–480 μs (Mikrosekunden), der Testtext–7.8 ms (Millisekunden): 1MK ist mehr als 16x schneller!

Mit den angesprochenen Verbesserungen lässt sich das Parsen des Korpus also um ein Vielfaches beschleunigen. Dies kann man in direkte Beziehung zu den Statistiken des Korpus in Verbindung bringen: Da die globale Set-Größe auch nur einen Bruchteil der Tokens-Größe (124,000 vs. 1 Mio.) beträgt, muss man nicht jedes Wort direkt den Operationen des Programms unterziehen, sondern kann auf das Ergebnis von früheren Vorkommen zurückgreifen. Nehmen wir ein Wort дорогой, das schon bereits einmal transkribiert und in Silben zerlegt wurde. Wenn es ein zweites Mal auftritt, ist es billiger (aus der CPU-Perspektive) das Wort aus einem „Wörterbuch“ aller bisher berechneten Ergebnisse zu holen als das Wort neu zu berechnen. Das ist eine sehr einfache Implementierung von *dynamic programming*.

3.5. Die praktische Implementierung

Die praktische Implementierung geht nach einem Mehrebendesign wie der Parser fürs Italienische von Iacoponi und Savy (2011) vor. Es gibt am Anfang eine Ebene, die den Text tokenisiert, d.h. in Wörter trennt. Innerhalb des vorliegenden

[13] Benutzung von ž oder č sind dadurch auch ausgeschlossen, da sie sich erst im erweiterten Zeichensatz cp1251 und UTF-8 befinden.

Korpus wurde dies bereits gemacht, normalerweise läuft dies auf Grundlage von Leerzeichen.

Danach wird jedes Wort auf einen Eintrag in einem zu Grunde liegenden Wörterbuch geprüft, ob phonetische Operationen nötig sind wie in Abschnitt 3.2 auf Seite 14 aufgezählt. Die phonetischen Änderungen führen zu einer pseudo-phonetischen Form.

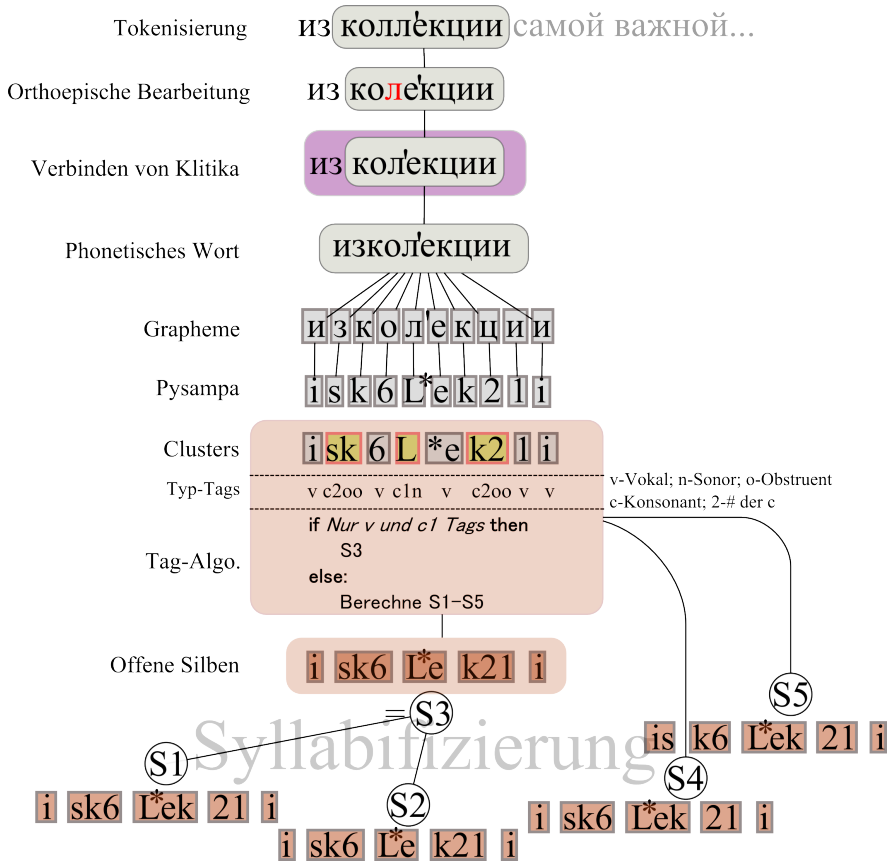

Abbildung 4: Bildliche Darstellung des Silbentrennungsprogramms.

Als nächstes werden die Klitika mit dem Hauptwort verbunden, sodass ein phonetisches Wort entsteht – dies ebenfalls zunächst auf der graphemischen Ebene. Diese Operation wird auf den ganzen Text angewendet. Auf Grund der Erfahrung aus dem 1MK werden auf dieser Ebene nach dem Verbinden der Klitika alle Verbindungen aus der Analyse getilgt, die nicht-kyrillische Zeichen enthalten, keine Betonung haben (aber mehrere Vokale) oder Abkürzungen sind. Im 1MK wurden auf diese Art 30,000 Wörter der ersten Art eliminiert, 60.000 der zweiten Art und 2400 Wörter der dritten Kategorie plus einer unbestimmten Zahl von Klitika, die diese Wörter umgeben haben. Nach diesem Filter blieben nur noch 784,000 phonetische Wörter (Tokens) für die Analyse, die sich aus 124,000 Types zusammensetzen bzw. 117,000 für den später gewählten Subkorpus. Danach werden die einzelnen Grapheme in Pysampa transkribiert.

Der nächste Schritt ist es, die Allophone in konsonantische und vokalische Intervalle zu teilen. An dieser Stelle passiert auch ein Tagging der Cluster, um

zu bestimmen, ob ein Wort vielleicht nur aus einfachen (V)CV-Verbindungen (c1(x) und v Tags) besteht und daher das S3 ausreicht. Dies spart bei ca. 42% der Wörtern unnötige CPU-Zyklen (=Diese Wörter haben eine *Set size* von 1). Im Falle, dass die Tags Hinweise auf Cluster geben (z.B. c2oo oder c2on usw.), werden die Silbenmodelle ausgerechnet. S1 und S2 werden auf Grundlage von S3 bzw. der Ebene der offenen Silbe bestimmt.

S4 und S5 wurden zuerst auch von dort aus berechnet. Nach einigen Tests aber erwies sich eine Abzweigung von der Clusterebene direkt zu den Modellen effektiver, da, wie man später sehen wird, bei diesen Modellen weniger offene Silben herauskommen.

4. Statistische Auswertung des Korpus

Wie die Analyse des Korpus in Abschnitt 3.1.2 auf Seite 13 gezeigt hat, gibt es zwei große Teile. Da das Ziel dieser Arbeit ein Vergleich der Silbenmodelle ist, beschränkt sich die Analyse auf den standard-Korpus. Die zusätzliche Variable des Subkorpus ist eine etwas schwierige Frage, da der eine Subkorpus 3x größer ist. Wie Baayen (2001) gezeigt hat, hängt die Wörtervielfalt von der Größe des Textes oder Korpus ab. Da Silben Wörter bilden, würde für Silben das Gleiche gelten. Für einen Vergleich müsste man nach Monte Carlo-Methoden aus den beiden Subkorpora ein zufälliges, gleich großes Sample ziehen – damit würde aber sich der Schwerpunkt dieser Arbeit auf den Vergleich der Subkorpora verschieben.

4.1. Globale Statistiken

Im gewählten standard-Subkorpus bekommt man nach dem Parsen von ca. 600,000 Wörtern 1,688 Mio. Silben, was 2.8 Silben pro Wort ausmacht. Da die Silbenanzahl bei allen Modellen gleich ist[14] (vereinfacht: Silbenanzahl entspricht immer der Anzahl der Vokale), erlaubt dies die Verteilungen der Silbenanzahl anzuschauen ohne die Modelle einzubeziehen. Die Statistik hilft dabei, zu entscheiden, auf welcher Ebene die folgenden Statistiken ausgeführt werden sollen: Types oder Tokens.

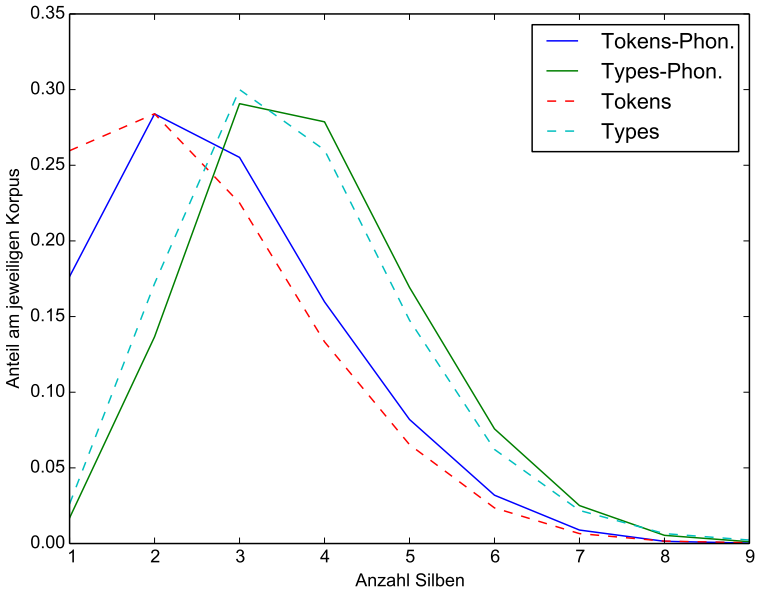

Abbildung 5: Silbenverteilungen auf den Ebenen der Types und Tokens (gestrichelt), dazu nach dem Verbinden der Klitika (durchgehend, mit „Phon.“-endend).

[14] Bei S4 und S5 wurde 1 einzige (!) Silbe weniger gefunden, was wohl ein Kodierungsfehler im Text ist.

Man kann in Abbildung 5 auf der vorherigen Seite sehen, dass die Tokens-Verteilung viel stärker nach links verzerrt ist, während die Types-Verteilung eher symmetrisch aussieht. Dies lässt sich mit dem Bezug zur Wortstatistiken erklären, denn einfache (1 oder 2-silbige) Verbindungen wie я, ты, и machen auch den größten Teil des Russischen National-Korpus bei den Häufigkeiten aus, sowie tendenziell auch jeder anderen Sprache. Was in der jeweiligen Ebene interessanter ist, ist der Effekt von verbundenen Klitika. Bei Tokens kann man eindeutig sehen, dass viele einsilbige Wörter (sprich: viele Klitika) innerhalb von größeren Wörtern aufgegangen sind und damit die Unterscheidung in längeren Wörtern verursachen. Bei Types ist die Unterscheidung ebenfalls mit dem Wirken der Klitika zu erklären. Auch wenn natürlich hochfrequente Wörter wie Klitika keine Rolle auf Typesebene spielen, so haben sie doch durch Verbinden auf der Tokensebene zu einem phonetischen Wort für mehr Vielfalt (sprich: mehr Types) *und* längere Wörter gesorgt.[15]

Nun muss aber eine Entscheidung im Sinne der Analyse getroffen werden: *Eine* Entscheidung wurde schon beim Design des Programms getroffen, nämlich Silben innerhalb von phonetischen Wörtern zu untersuchen. Die obige Hinzunahme von Korpora ohne Verbindung mit Klitika sollte nur den Effekt der Maßnahme verdeutlichen. Bei der wichtigen Entscheidung zwischen Tokens (Tokens-Phon. in Abbildung 5 auf der vorherigen Seite) und Types (Types-Phon.) wählt man wegen der besseren Verteilung und Abwesenheit von Frequenzeffekten von Klitika die Types-Ebene.

4.2. Vergleich der Silbenmodelle

Um eine Vergleichsform zu finden, lässt sich am besten zuerst zusammenfassen, was das Programm für Daten über einen Text ausgibt.

1. Silbenstatistiken und ihre Frequenz wie in Tabelle 3 auf der nächsten Seite links.

2. Wörter und ihre Segmentierung zwischen den Modellen + Anzahl der unterschiedlichen Segmentierungen wie in Tabelle 3 auf der nächsten Seite rechts. Ssi ist dabei eine Abkürzung für Set size – wie groß also der Set (einzigartige Segmentierungen) für ein Wort ist.

Die Wahl des zweiten Weges würde bedeuten, dass der Schwerpunkt sich auf die Eigenschaften der Wörter verlagern würde wie die Anzahl ihrer Laute, Frequenz, Google Ngram-Profil usw. Schaut man sich die Häufigkeiten der einzelnen Ssi's an, sieht man aber, dass von den 117,000 Types fast 42% keinen Unterschied

[15] Vereinfacht: In einer Liste [a,b,a,b,c,b,d,d,c] mit b als Klitika haben wir 9 Tokens und 4 Types (Durch.-Länge: 1). Durch Verbinden von b mit jeweils folgendem Item bekommen wir 6 Types (Durch.-Länge: 1.5). Dies ist der Unterschied zwischen Types und Types-Phon.

zwischen den Modellen ergeben haben (=Ssi von 1; Ssi von 5 bedeutet, dass es 5 verschiede Trennungen gab). Eine gepaarte Pearson-Korrelation ergibt daher für alle Modelle einen hohen Wert zwischen 0.98-0.99, nach Spearman immerhin nur 0.93-0.98. Dieses Zwischenergebnis zeigt, dass die Modelle allesamt ähnliche Silbenfrequenzen ergeben.

Tabelle 3: Links: Ein kleiner Teil (5 von 16,878) an Silben mit Frequenzen und ob sie ausgewählt wurden (An(alyse)?). Rechts: Häufigkeiten der Set size (Ssi) und ihr Anteil am Types-Subkorpus sowie Beispiele. „*“–Betonung; links und rechts ohne Beziehung.

σ	s1	s2	s3	s4	s5	An?
*ent	2	2	2	2	2	×
i	5	93	117	104	27	✓
2*e	70	154	215	118	169	✓
us	28	0	0	0	21	×
al	8	8	2	7	8	✓

Ssi	N_{Ssi}	Anteil	Beispiel
1	49,676	42.3%	декабр*я
2	50,177	42.7%	б*ублики
3	14,002	11.9%	моск*овской
4	3079	2.6%	напросп*екте
5	328	0.2%	инсп*ектора

Dieses Ergebnis ist aber vom statistischen Standpunkt nicht zufriedenstellend, schließlich wollen wir wissen, *wie* sich die Modelle unterscheiden. Dafür reicht eine solche Korrelation nicht aus und vor allem die Daten benötigen eine weitere logische Spezifizierung. Zum einen wählen wir im Folgenden für den Vergleich nur die Wörter, die auch Unterschiede gezeigt haben (Ssi>1) und wir setzen eine weitere Filterung ein, um Ausreißer zu filtern. Dies ist dringend notwendig, da nach der Einbeziehung des Kriteriums Ssi>1 noch immer die Daten sehr ungleichmäßig sind. Der Median der Summe aller Modelle ist 52, der Mittelwert 465. Die untere Quartile beträgt 16, verteilt auf 5 Modelle ergibt das ≤ 3.2 Silben an durchschnittlichen Beobachtungen für die unteren 25% der Silben. Auch dieses Zwischenergebnis erbringt die Erkenntnis, dass auf Silbenebene ähnliche Gesetze gelten wie auf Wortebene. Eine geringe Zahl von Silben nimmt den größten Teil der Vorkommen ein. In Zahlen ausgedrückt: 50% der Vorkommen werden von nur 42 Silben eingenommen. Extremer würde es sicher sein, falls wir noch die Frequenzeffekte einbezogen hätten (Tokens-Ebene).

Das Problem ist in der Textanalyse nicht neu und wurde schon im obigen Unterkapitel angesprochen. Als erster Schritt wurden zuerst alle Zeilen aussortiert, die auch nur einen einzigen Wert 0 hatten. Die Lösung für die weitere Analyse ist, um Lebart u. a. (1998) zu folgen, zunächst einen *threshold* (eine Untergrenze zur Berücksichtigung von Beobachtungen) anzusetzen. Dazu gibt es keinen bestimmten Wert; wichtig ist, eine aussagekräftige Datengrundlage für weitere Analysen zu erhalten. Wir wählen zunächst einen Wert von 25, pro Modell also durchschnittlich 5 Beobachtungen, was eine Voraussetzung für den χ^2-Test ist. Bezogen auf Tabelle 3 heißt dies, dass man alle Zeilen aussortiert, die in der Summe weniger als 25 erbringen. Die Abbildung 6 auf der nächsten Seite zeigt die bisher erfolgte Filterung

in chronologischer Abfolge von der Tokensebene zu Types und dann zu Silben bis zum uns interessierenden Teil der Silben. Die besprochene Filterung *innerhalb* des Silbensets kann man auch anhand von Beispielen in Tabelle 3 auf der vorherigen Seite links (Spalte „An?“) nachverfolgen, wobei Häckchen die Zugehörigkeit zur schließlichen Auswahl anzeigt.

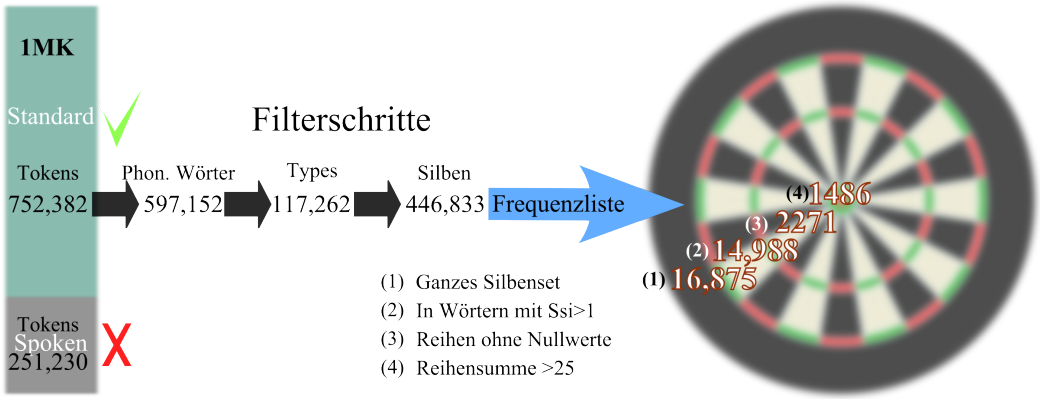

Abbildung 6: Die unterschiedlichen Schritte der Filterung; jeder Schritt nach rechts übernimmt die Bedingung ($\sigma \ni \ldots$) des linken Schritts. Silben in der Mitte der Frequenzliste haben die größte Aussagekraft.

4.3. Correspondence Analysis

Ein χ^2-Test ist – dem großen Sample wegen – wenig überraschend hochsigfikant (p>0.01). Die verbleibenden 1486 Silben von ehemals über 16,000 dienten folglich für eine Correspondence Analysis, die von Baayen (2008) und Lebart u. a. (1998) angewandt wurde für *count data*; eine Schwestertechnik für Messdaten ist die principle components analysis (PCA). PCA und CA behandeln (vereinfacht gesagt) die Daten wie einen multi-dimensionalen Raum, in dem durch Einsetzen von Ebenen versucht wird, möglichst viele Datenpunkte zu treffen. Je nachdem wie erfolgreich eine Ebene ist, vereint sie mehr oder weniger Erklärungspotenzial, das in *eigenvalues* gemessen wird bzw. im Plot als Anteil an der gesamten Standardabweichung. Nach Baayen (2008, S. 141) hat die CA aber einen weiteren praktischen Vorzug: „The attractiveness of correspondence analysis resides in the possibilities it offers for visualization“.

Alle 1486 Silben in ihrer graphischen Form einzubeziehen, würde die gleiche Anzahl an Kategorien bedeuten (der Silbenpool ist ja ein Set). Deswegen soll abstrahiert werden: Es sollen die in den vorherigen Abschnitten benutzten Begriffe wie Onset, Coda benutzt werden. Zum Beispiel soll eine Silbe „sP*ek“ (aus напросп*екте) charakterisiert werden als mit Obstruent beginnend (Plot 1) und endend (P2), 1 Konsonant in Coda (P3), 2 Konsonanten im Onset (P4).

Zunächst wurde mittels der umgewandelten Frequenzen in Ränge eine CA-Faktor-Matrix erstellt mit 4 Faktoren, wobei die ersten beiden (X1 und X2) am

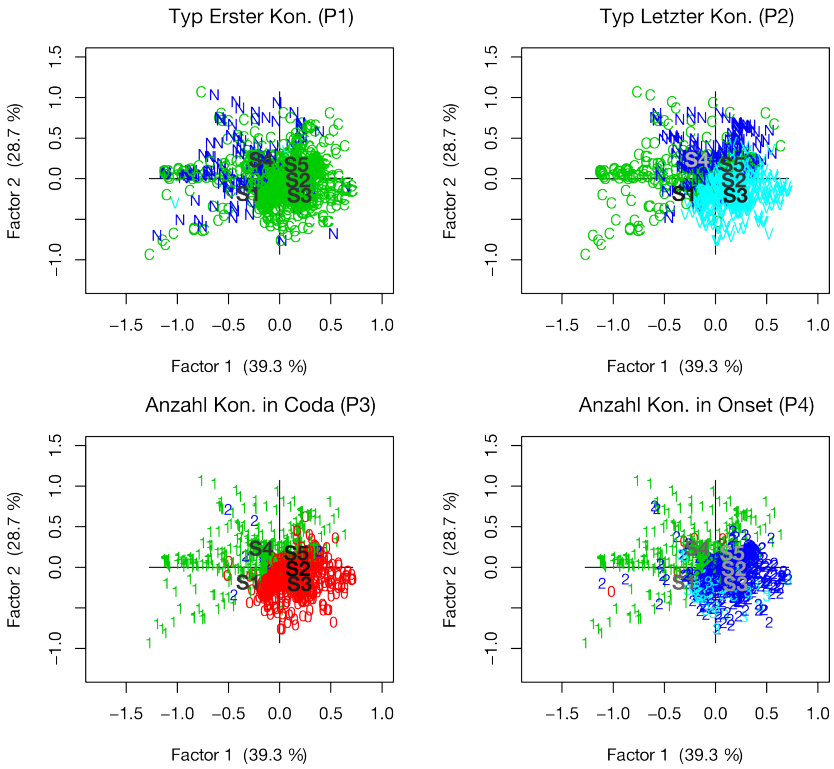

Abbildung 7: CA der Silbenmodelle. Statt graphischer Form der Silben sind ihre Eigenschaften gezeigt. Kon.–Konsonant. Farben von S1-5 wurden jeweils wegen der Sichtbarkeit angepasst.

meisten die Varianz zusammenfassen und daher im folgenden für die Analyse verwendet werden. Danach wurden die vorgestellten vier Silbenkategorien nacheinander in Abbildung 7 eingebracht worden. Der erste Plot (P1) scheint auch gleich die Clusteeffekte teilweise aufzulösen. Die diagonale Grenze, die durch fast alle Plots geht, erklärt sich aber vor allem durch den zweiten Plot. Er zeigt, dass das Bondarko-Modell tatsächlich ein Extrem markiert und zwar durch die Eigenschaft, dass der letzte Laut in der Silbe ein Vokal ist. Ebenfalls befindet sich S2 innerhalb der vokalischen Silbenendung. S5 und S1 befinden sich am Übergang zur Gruppe mit konsonantischer Silbenendung. Andererseits befindet sich S4 abseits der anderen Modelle in einem deutlichen Cluster konsonantischer Endungen – hier wiederum zwischen Sonoren und Obstruenten. P3 und P4 imitieren die Aussage von P1 und P2 mit anderen Eigenschaften: Die Gruppen trennen sich durch die Eigenschaft, wie viel die Modelle im Coda oder Onset erlauben. S3, S2 erlauben eindeutig mehr Konsonanten im Onset, was zu Lasten der Codas geht, bei S4 ist andersherum. S5 und S1 liegen auf der Grenze.

Bei der Interpretation sagt P2 (Silbenauslaut) am meisten aus; er erklärt die Tendenz von S1 nach links, von S5 nach oben und S4 zwischen S1 und S5. Im Bereich von S5 und S2 liegen deutlich mehr Silben mit Sonoren am Ende. Wir wissen aus der Theorie: Für S2 ist die Anwesenheit eines Sonors im Cluster entscheidend, um ihn zur Coda zu zählen. S5 zählt Sonore ebenfalls zur Coda, sofern nach ihm ein Obstruent kommt. S1 kann auch Obstruenten in die Coda

Tabelle 4: Spearman-Korrelationsmatrix der CA-Faktoren mit Silbeneigenschaften. Namen oben sind abgekürzt.

	X1	X2	LL	EL	KC	KO
X1	1.00					
X2	-0.12	1.00				
Let. Laut	0.01	-0.69	1.00			
Ers. Laut	0.18	-0.30	0.23	1.00		
Anzahl Kon. in Coda	0.07	-0.72	0.96	0.26	1.00	
Anzahl Kon. in Onset	0.02	0.60	-0.53	-0.26	-0.55	1.00

nehmen, sofern die Betonung auf der Silbe ist. Dieser Fall spiegelt sich durch die Verortung innerhalb der Obstruenten-Coda wider. S4 hat dagegen ein Coda-Profil, dass sich zwischen S5 und S1 orientiert.

Die Korrelationen in Tabelle 4 mit den einzelnen Faktoren sind für den ersten Faktor (X1) unbedeutend; nicht nur der visuelle Eindruck bestätigt, dass der 2. Faktor entscheidend ist, sondern auch die hohen Korrelationen mit den jeweiligen Silbeneigenschaften. Dass so gut wie alle Korrelationen (≥ 0.1) signifikant sind ($p>0.05$), soll angesichts des großen Samples von 1486 Silben nicht überraschen. X1 wird wohl größtenteils entweder eine andere latente Eigenschaft widerspiegeln oder einfach eine zusammengefasste Variation sein. An der negativen Korrelation zwischen Anzahl der Konsonanten in Coda und im Onset sieht man, dass eine *inverse* Beziehung existiert; die beiden Silbeneigenschaften haben des Weiteren eine hohe Korrelation mit X2 – dem Faktor, der für die Clustering-Effekte auf der vertikalen Achse verantwortlich ist. Ergänzend dazu ist vor allem die Eigenschaft des letzten Lautes in der Silbe mit der Anzahl der Größe des Onsets und Coda stark korreliert (0.6, -0.72), sodass die Variation, die X2 ausdrückt, vor allem durch die Silbeneigenschaften entsteht und die Silbenmodelle unterscheidet.

Zusammengefasst bestätigt sich der Eindruck, dass die Coda den Unterschied in den Silbenmodellen ausmacht. Dies drückt aber eine größere, *latente* Eigenschaft aus, nämlich die Offenheit bzw. Geschlossenheit. S4 bevorzugt eindeutig eine geschlossene Silbe, während S1 und S5 eine gemischte Tendenz aufweisen allerdings mit anderen Vorzeichen (Obstruent vs. Sonor). S3 – das zeigt sich nicht nur in der Theorie, sondern auch in Statistiken – präferiert eine offene Silbe.

Abschließende Worte und Ausblick

Im Rahmen dieser Arbeit wurde mehrere Dinge praktisch angegangen und gelöst. Es wurden einige grundlegende Begriffe (Silbe, NLP) zum Rahmen der Arbeit definiert.

Danach wurden konkrete Silbentheorien und die Hauptgemeinsamkeiten und -unterschiede dargestellt. Dabei wurden russische und westliche Ansätze zunächst theoretisch gegenübergestellt.

Die Korpusbeschreibung stellte die Weichen für eine weitere Eingrenzung der Daten. Anschließend wurde die technische Umsetzung vorgestellt. Dabei wurden Schwierigkeiten festgestellt, die nicht optimal in dieser Arbeit gelöst werden konnten (Transkription); allerdings konnten für die meisten Probleme formale und systematische Ansätze aus der Russistik angewendet werden. Es wurde unter Einbeziehung der aktuellen Theorien eine grundsätzliche Entscheidung zu Gunsten von manuellen Regeln gefällt, die auch den Charakter der Daten widerspiegeln.

Die wirkende Mechanik konnte zunächst auf einem technischen und abstrakten Niveau (Kap. „Programm-Mechanik“) vorgestellt werden und dann auf einem praktischen („Die praktische Implementierung“). Es zeigte sich, dass ein Programm durch ein modulares Design am besten für Texte geeignet ist und bei Korpora Geschwindigkeit und Effizienz am besten sicherstellt.

Der statistische Abschnitt rekapitulierte einige Korpusparameter und brachte dies mit Schwierigkeiten der Analyse in Verbindung. Ein Ergebnis war, dass im Russischen unter Hinzunahme von Frequenzeffekten kaum Unterschiede zwischen den Silbenmodellen existieren. An den Zahlen zeigte sich außerdem das, was man von der Silbe als Brücke zwischen Wort und Phonem erwarten würde: 117,000 Types lösten sich in 16,000 Silben auf, die sich wiederum aus 44 Phonemen bzw. Allophonen zusammensetzten.

Andererseits zeigten sich ähnliche Frequenzeffekte wie auf Wortebene, dass nämlich eine geringe Anzahl von Types einen großen Teil an Beobachtungen einnahm. Daher wurden im Weiteren die Daten nach bestimmten Kriterien gefiltert bis ein aussagekräftiges Sample erreicht war. An diesem konnten Unterschiede durch Anwendung der CA und Einbeziehung von ouverten und latenten Silbeneigenschaften aufgezeigt werden, die eine gewisse Clusterung nach Eigenschaften verursachten. Diese Clusterung konnte man gut mit den theoretischen Eigenschaften erklären.

Die Ergebnisse sind interessant, reizen aber das Potential der Daten und des Programms nicht komplett aus: Die Wortebene wurde nur für die Filterung von Ssi's verwendet sowie zum Gewinnen der Silbenfrequenzen, man kann aber von Wörtern ausgehend schauen, inwiefern ihre Eigenschaften mit Ssi oder ihre Trennung mit ihrer Herkunft (Wörter aus romanischen Sprachen nach SSP?) korrelieren. In Bezug auf Silben könnte man die Korpusgrundlage vergleichen z.B.

mit Spoken-Teil oder Lehrbücher, wo Betonungen gesetzt sind, mit den Subkorpora. Dafür müsste man allerdings ein Modell wählen, denn sonst hätte man neben Silben und Korporatypen als Variablen auch Silbenmodelle. Die vorliegende Arbeit entschied sich durch Filterung dafür, den Korpus als Kontrollvariable zu nehmen. Die weitere Perspektive könnte der Versuch eines Klassifikators von Texten sein, der auf Grundlage von Silben das Genre eines Textes bestimmt.

Es wurde viel Zeit investiert in eine einfache, aber auch flexible Struktur, die als Basis für weitere Ergänzungen dienen kann und soll. Die weitere Entwicklung soll ermöglichen, dass man einfache psycholinguistische Daten erheben kann und diese mit der Datenbank abgleichen kann, als Fragestellung würde sich anbieten: Welchem Modell ist die Segmentierung einer Person ähnlich? Andererseits müsste man in Zukunft einige andere Richtungen bearbeiten: Automatisches Betonungssetzen, Zusammensetzen von Phonsegmentierung in Praat-Textgrids zu Silben, Erweiterung der phonetischen Wörterbuchbasis, um eine Transkription mehr von Daten und weniger von formalen Regeln abhängig zu machen. Außerdem könnte man eine graphische Oberfläche für das Programm schaffen.

Literatur

Ageenko, Florencija und Majja Vladimirovna Zarva (1993). *Slovar Udarenij russkogo jazyka : okolo 76000 slovarnych edinic*. Moskva: "Russkij Jazyk" (siehe S. 14).

Allen, James F. (1995). *Natural language understanding*. Redwood City, Calif: Benjamin/Cummings (siehe S. 6).

Avanesov, Ruben (1956). *Fonetika sovremennogo russkogo literaturnogo jazyka*. Moskva: Izdat. Moskov. Univ. (siehe S. 8).

Baayen, R. H. (2001). *Word Frequency Distributions*. Dordrecht: Kluwer (siehe S. 21).

– (2008). *Analyzing linguistic data : a practical introduction to statistics using R*. Cambridge: Cambridge Univ. Press (siehe S. 24).

Bird, Steven u. a. (2009). *Natural Language Processing with Python*. O'Reilly (siehe S. 6).

Bolla, K. u. a. (1968). *Kurs sovremennogo russkogo jazyka*. Budapest: Tankönyvkiadó (siehe S. 8).

Bondarko (1998). *Fonetika sovremennogo russkogo jazyka*. St. Petersburg: Izd. S.-Peterburgskogo Univ. (siehe S. 9, 17).

Cholin, Joana (2011). »Do Syllables Exist? Psycholinguistic Evidence for the Retrieval of Syllabic Units in Speech Production.« In: *Handbook of the Syllable*. Hrsg. von Charles Cairns. Leiden, S. 225–254 (siehe S. 5).

Coté, Marie-Helene und Viktor Kharlamov (2011). »The Impact of Experimental Tasks on Syllabification Judgements. A Case Study of Russian«. In: *Handbook of the Syllable*. Hrsg. von Charles Cairns. Leiden, S. 273–294 (siehe S. 11).

Duanmu, San (2009). *Syllable structure : the limits of variation*. Oxford: Oxford Univ. Press (siehe S. 4).

Goldsmith, John (2011). »The syllable«. In: *The handbook of phonological theory*. Hrsg. von John Goldsmith u. a. Malden, MA [u.a.]: Wiley-Blackwell (siehe S. 2).

Goldwater, S. und M. Johnson (2003). »Learning OT constraint rankings using a maximum entropy model.« In: *Stockholm Workshop on Variation within Optimality Theory,* Stockholm University Press. (siehe S. 10, 17).

Grishman, Ralph (1989). *Computational linguistics : an introduction*. Cambridge [u.a.]: Cambridge Univ. Press (siehe S. 6).

Iacoponi, Luca und Renata Savy (2011). »Sylli: Automatic Phonological Syllabification for Italian.« In: *INTERSPEECH*. ISCA, S. 641–644. URL: `http://dblp.uni-trier.de/db/conf/interspeech/interspeech2011.html#IacoponiS11` (siehe S. 4, 10, 17–18).

Itapov, J. V und S.B. Stepanova (1997). »Podrobnoe opisanie avtomatičeskogo transkriptora orfografičeskogo teksta«. In: *Eksperimentalno-fonetičeskie analiz*

reči: problemy i metody. Hrsg. von L. V. Bondarko. Bd. 3. St. Peterburg: Izdatelstvo St. Peterburgskogo universiteta (siehe S. 15).

Ivanova, Tat'jana F. (2004). *Novyj orfoepičeskij slovar' russkogo jazyka : proiznošenie, udarenie ; grammatičeskie formy ; okolo 40000 slov*. Moskva: Russkij Jazyk Media (siehe S. 14).

Jurafsky, Dan und James H Martin (2008). *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. 2. Aufl. Upper Saddle River, NJ: Pearson Prentice Hall (siehe S. 17).

Kalenčuk, M. L. u. a. (2012). *Bol'šoj orfoepičeskij slovar' russkogo jazyka*. Moskva: AST-Press Kn. (siehe S. 15).

Kalnyn', Ljudmila E. und Ljudmila I. Maslennikova (1985). *Opyt izučenija slova v slavjanskich dialektach*. Moskva: Nauka (siehe S. 11).

Kasatkin, Leonid (2001). *Russkij jazyk*. Moskva: Academia (siehe S. 8).

Kavitskaya, Darya und Maria Babyonyshev (2011). »The Role of Syllable Structure: The case of Russian-Speaking Children with SLI«. In: *Handbook of the Syllable*. Hrsg. von Charles Cairns. Leiden, S. 353–371 (siehe S. 5).

Kelih, Emmerich (2012). *Die Silbe in slawischen Sprachen : von der Optimalitätstheorie zu einer funktionalen Interpretation*. München: Sagner (siehe S. 9, 11).

Kempgen, Sebastian (2003). »Phonologische Silbentrennung im Russischen«. In: *Rusistika - Slavistika-Lingvistika. Festschrift für Wemer Lehfeld zum 60. Geburtstag*. Hrsg. von Sebastian et al. Kempgen. München: Sagner, S. 195–211 (siehe S. 10).

Knjazev, Sergej (1999). »O kriterijah slogodelenija v sovremennom russkom jazyke: Teorija volny sonornosti i teorija optimalnosti«. In: *Voprosy Jazykoznija* 1, S. 84–102 (siehe S. 11).

Krukover, V. (2008). *Orfoepičeskij slovar' russkogo jazyka*. Viktorija plus (siehe S. 14–15).

Lebart, Ludovic u. a. (1998). *Exploring textual data*. Dordrecht: Kluwer (siehe S. 23–24).

Lehfeldt, Werner (1971). »Ein Algorithmus zur automatischen Silbentrennung«. In: *Phonetica* 24, S. 212–237 (siehe S. 10).

Manning, Christopher D. und Hinrich Schütze (1999). *Foundations of statistical natural language processing*. Cambridge, Mass.: MIT Press (siehe S. 6).

Matusevič, Margarita I. (1976). *Sovremennyj russkij jazyk : fonetika ; Učeb. posobie dlja studentov ped. in-tov po spec. "Russkij jazyk i literatura"*. Moskva: Prosveščenie (siehe S. 7).

McKinney, Wes (2012). *Python for Data Analysis*. O'Reilly (siehe S. 2, 32).

Ooi, Vincent B. Y. (1998). *Computer corpus lexicography*. Edinburgh: Edinburgh Univ. Press (siehe S. 6).

Pulgram, Ernst (1970). *Syllable, word, nexus, cursus*. The Hague: Mouton (siehe S. 9).

Rezničenko, Irina L. (2005). *Orfoepičeskij slovar' russkogo jazyka : proiznošenie, udarenie ; okolo 25000 slov*. Moskva: AST (siehe S. 14–16).

Vinogradov, Viktor u. a. (1953). *Grammatika russkogo jazyka*. Moskva: Academy of Science of the USSR (siehe S. 7, 16).

Wade, Terence L. B. (1993). *A comprehensive Russian grammar*. Oxford: Blackwell (siehe S. 8).

Ščerba, Lev V. (1983). *Teorija russkogo pisma*. Leningrad: Nauka (siehe S. 7).

A. Technische Spezifikationen

Die Statistiken wurden mit Hilfe der Code-Sprachen R und Python sowie indirekt mit C gemacht. Bei R[16] wird die aktuelle Version 3.0.1 verwendet, Python[17] und die Statistikbibliothek *pandas* (McKinney, 2012) werden in der Standardimplementierung in C (CPython, nicht zu verwechseln mit Cython) verwendet. Die benutzte Version ist 2.7.4, das getestete Betriebssystem ist Windows XP, das Programm wurde aber ebenfalls auf Windows 8 und 7 erfolgreich getestet. Unter Mac oder Unix OS steht eine lauffähige Version (noch) aus, die unkompilierten Skripte mit der Endung `.py` sollten auf jeden System laufen, wenn man Python 2.7 installiert. Die kompilierte Fassung auf dem USB-Stick ist auf Windows zugeschnitten. Vorrangig ist es als Skript für den Python Interpreter gedacht; auf dem Stick befindet sich eine allein lauffähige *cmd*[18]-Anwendung. Das neueste Skript wird auf der github-Plattform zu finden sein, wo die Entwicklung nach der vorliegenden Arbeit weiterverlaufen wird.

Für Python wird neben pandas (10.1) auch lxml[19] benötigt.

Zur Command Line Anwendung: Mit dem Programm werden 5 Texte geliefert. 3 belletristische Texte und zwei zufällige Texte aus dem 1MK. Ihre Metadaten kann man lesen, wenn man es in einem Editor oder besser in einem beliebigen Browser (Firefox, IE usw.) öffnet.

Zum Starten muss man auf `syl_parse.exe` klicken. Dann parst das Programm automatisch den Ordner „test_data/mk1“ mit seinem gesamten Inhalt. Dies ist nützlich, wenn man sich von der ruscorpora-Stiftung eine eigene Kopie des 1MK besorgt. Die gesamten Files legt man in den Ordner und lässt sie Parsen. Silbenstatistiken zum 1MK sind übrigens auch so mitgeliefert im Ordner „syl_data“, nur nicht die Ausgangstexte oder die Wörter.

Wenn man allerdings etwas anderes als den 1MK parsen will, so sollte man im Auge behalten, dass das Programm zuallererst auf das Format des Korpus abgestimmt wurde. Alle anderen Texte außer den Testtexten können Probleme verursachen (Betonungszeichen müssen immer mit „'“ markiert sein, Satzzeichen usw.). Will man Testtexte außer dem 1MK parsen, so muss man in die Command Line von Windows gehen und dann z.B. folgenden eintippen (Zeilen mit „#-Kommentar):

```
#wechselt zum Ordner mit der .exe
E:
cd E:/comp_syl_parser
syl_parser test_data/lit_texts txt
```

[16] http://www.r-project.org/
[17] http://www.python.org/
[18] Windows Command Line
[19] http://lxml.de/

```
#Programm startet und gibt die Etappen aus: read done, map done usw.
```

Alternativ dazu kann man auch die Datei `syl_parser.bat` anklicken. Sie startet automatisch das Parsen der Belletristik-Texte.

Die geparsten Files sowie die Silbenstatistiken finden sich dann im Ordner „outp_data“.

B. USB-Inhalt

Es wurde sich für ein USB-Stick als Datenträger entschieden, weil das Programm auch einen Speicherplatz braucht, um die Parser-Ergebnisse zu schreiben. CD's oder DVD's sind aber nur durch einen Brenner beschreibbar und nicht durch ein kleines Programm. Sehr wichtig ist folgendes: Alle Dateien sind in cp1251 Enkodierung und nicht UTF-8, um also Daten mit Nicht-ASCII Zeichen in Excel oder sonst wo zu bearbeiten, sollte man als Decoder cp1251 wählen, sonst würde man nur seltsame Zeichen lesen.

Es sind 4 Ordner vorhanden:

comp_syl_parser Hier befindet sich eine kompilierte .exe Anwendung für den Parser. Um diese zu benutzen braucht man nichts zu installieren, sondern es reicht nur auf `syl_parser.exe` zu klicken. Dann wird automatisch ein Teil des 1MK geparst. Für die anderen Testtexte sollte man wie zuletzt im Anhang A beschrieben vorgehen, d.h. zum Beispiel die `syl_parser.bat` Datei anklicken. Alles andere in diesem Ordner ist irrelevant. Es sind einfach Bibliotheken, die für das Programm notwendig sind.

uncomp_syl_parser Hier ist eine unkompilierte Fassung sämtlicher Funktionen zu finden. Man kann alle Files mit der Endung .py in jedem beliebigen Editor öffnen und lesen. Das File `syl_parser.py` ist das, was man als Python-Anwendung startet. Man muss dafür aber die Dependenzen pandas und lxml installieren. Alle anderen Files übernehmen verschiedene Funktionen wie orthoepisches Anpassen (`orfo_prep.py`), Verbinden der Klitika (`parse_txt_prep.py`) und Transkription (`new_syl_class.py`).

- Der Ordner **orfo_data** beinhaltet das orthoepische Wörterbuch mit ca. 900 Wörtern und die Tabellen von Kempgen im .csv Format. Alle Dateien werden von dem Programm benötigt, daher sollte hier nichts entfernt werden oder modifiziert werden. Öffnen und lesen geht aber ohne Probleme, solange man die Kodierung nicht verändert.
- Der Ordner **test_data** beinhaltet alle Files zum Parsen, so den 1MK im Ordner mk1 und die Belletristik Texte in lit_texts. mk1-Ordner enthält zwei zufällig ausgewählte Files. lit_texts-Ordner enthält eine Geschichte von M. Zošenko „Interesno pridumala“ sowie den Text „Tolstyj i Tonkij“ von A. Čechov und „Bargamot und Garyska“ von L. Andreev. Die letzten zwei Texte wurden aus Wikipedia heruntergeladen und für die Verarbeitung durch das Programm modifiziert.[20] Der erste Text wurde für ein anderes Projekt bereits in einigen Worten modifiziert, es wurden aber Betonungen ergänzt.

[20] http://ru.wikisource.org/wiki/Категория:Тексты_с_ударениями

- In den Ordner **outp_data** werden die Daten geschrieben, die man geparst hat: Ein File mit Wörtern und den 5 Segmentierungen und dazu Silbenstatistiken basierend darauf.

syl_data Hier sind einige Silbenstatistiken aus dem 1MK (`syl_share_spokstan.csv`) zu finden, getrennt nach Spoken- und Standard-Subkorpora, allerdings wurden sie vor einer kleinen Korrektur des Programms gemacht, sodass sie 23 Silben weniger haben als eigentlich da ist. Dazu gibt es Statistiken zum ganzen Korpus (`syl_compcorp.csv`), die auch aktuell sind, allerdings wurde dazu keine Unterscheidung zwischen Spoken und Standard gemacht. Dazu wird die CA (`factor_postca.csv`) mit den Faktoren und den Eigenschaften sowie den Ausgangswerten (`syl_preca.csv`) geliefert. Wörterdaten kann man durch die Lizenz nicht zugänglich machen. Sofern man aber den Korpus bekommen hat, kann man ihn durch Ablegen der .xhtml-Files in den Ordner „/test_data/mk1“ parsen lassen. Dazu startet man einfach die `syl_parser.exe`. Dann sollte der Output geliefert werden.

thesis Hier ist die Arbeit im .pdf Format zu finden und die Abbildungen, sofern verfügbar im .svg und .png/.pdf.

C. Konvertierung der Grapheme

Die Konvertierung von der Graphemebene auf die Allophonebene hin zu einer computerfreundlichen Notierung unterscheidet sich von SAMPA. Die Tabelle 5 zeigt die Abweichungen für alle Vokale sowie für abweichende Konsonanten. Die Logik bei Konsonanten ist Zeichen, die aus zwei Elementen bestehen, durch ein zu ersetzen. Bei paarig velarisierten und palatalisierten Konsonanten wird ein kleiner Buchstabe für velarisierte Zeichen ([sˠ] → s) verwendet und ein großer für palatalisierte ([sʲ] → S). Das gilt für alle Konsonanten *außer* den unpaarig velarisierten/palatlasierten Konsonanten. Dieses Vorgehen tritt nämlich in Konflikt mit der SAMPA-Notierung für <ж, ш>, deshalb wird für diese Zeichen eine Nummer gewählt. Für Affrikate nutzt die SAMPA zwei Zeichen; um Morphemgrenzen anzuzeigen, wird ein „-“-Zeichen verwendet. Diese Praxis ist für Textprocessing nicht gut, da man bei der Iteration Probleme bekommt. Daher werden Affrikate ebenfalls zu einem Zeichen und das Zeichen für die Morphemgrenze kann dann entfallen. Alles in Allem kann man diese Tabelle dazu verwenden, alle Ergebnisse in SAMPA zu konvertieren z.B. durch regular expressions. Aus Zeitgründen wurde das zunächst nicht ins Programm implementiert.

Tabelle 5: Graphem-Sampa Konvertierung für ausgewählte Zeichen

Graphem	IPA	SAMPA	Pysampa
а & я	ə	@	
	ɐ	6	
	’a	a	
	i	i	
е	ə	@	
	i	I	
	ɛ	E	e
ё	ɔ	O	o
о	ə	@	
	ɐ	6	
	’ɔ	O	o
у & ю	ʊ	U	u
ы	ɨ	1	y
э	ɛ	3	e
	ə	@	
ш	ʃ	S	7
ж	ʒ	Z	5
ц	t͡s	ts	2
ч	t͡ʃʲ	tS’	4
щ	ʃʲː	S’:	3

Eidesstattliche Versicherung

Ich versichere an Eides Statt durch meine eigene Unterschrift, dass ich die vorliegende Arbeit selbständig und ohne fremde Hilfe angefertigt und alle Text-Stellen, die wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen sind, als solche kenntlich gemacht und mich auch keiner anderen als der angegebenen Literatur, insbesondere keiner im Quellenverzeichnis nicht benannten Internet-Quellen, bedient habe. Diese Versicherung bezieht sich auch auf die in der Arbeit gelieferten Zeichnungen, Skizzen, bildlichen Darstellungen und desgleichen.

Ich versichere, diese Arbeit nicht bereits in einem anderen Prüfungsverfahren eingereicht zu haben und bestätige, dass die eingereichte schriftliche Fassung derjenigen auf dem Speichermedium entspricht.