Bochumer Linguistische Arbeitsberichte 26

KidRef: Ein Kinderreferenzkorpus

Katrin Ortmann und Helena Wedig

Bochumer Linguistische Arbeitsberichte

Herausgeberin: Stefanie Dipper

Die online publizierte Reihe „Bochumer Linguistische Arbeitsberichte“ (BLA) gibt in unregelmäßigen Abständen Forschungsberichte, Abschluss- oder sonstige Arbeiten der Bochumer Linguistik heraus, die einfach und schnell der Öffentlichkeit zugänglich gemacht werden sollen. Sie können zu einem späteren Zeitpunkt an einem anderen Publikationsort erscheinen. Der thematische Schwerpunkt der Reihe liegt auf Arbeiten aus den Bereichen der Computerlinguistik, der allgemeinen und theoretischen Sprachwissenschaft und der Psycholinguistik.

The online publication series “Bochumer Linguistische Arbeitsberichte” (BLA) releases at irregular intervals research reports, theses, and various other academic works from the Bochum Linguistics Department, which are to be made easily and promptly available for the public. At a later stage, they can also be published by other publishing companies. The thematic focus of the series lies on works from the fields of computational linguistics, general and theoretical linguistics, and psycholinguistics.

Band 26 (June 2024)

Herausgeberin: Stefanie Dipper
Sprachwissenschaftliches Institut
Ruhr-Universität Bochum
Universitätsstr. 150
44801 Bochum

Erscheinungsjahr 2024
ISSN **2190-0949**

Katrin Ortmann und Helena Wedig

KidRef: Ein Kinderreferenzkorpus

2024

Bochumer Linguistische Arbeitsberichte

(BLA 26)

Hinweise zum vorliegenden Band

Die vorliegende Arbeit entstand im Rahmen eines studentischen Forschungsprojekts im Jahre 2018 und wird hier unverändert abgedruckt.

Die Korpusdaten zu KidRef stehen unter `https://gitlab.ruhr-uni-bochum.de/vamos-cl` zum freien Download zur Verfügung.

Stefanie Dipper (Hrg.), im Juni 2024

Forschungsprojekt-Dokumentation

zum

Kinderreferenzkorpus

von

Katrin Ortmann & Helena Wedig

26.04.2018

Abstract

Bislang existieren nur sehr wenige deutsche L1-Korpora mit Texten von jungen Schulkindern und häufig sind diese nicht frei verfügbar oder liegen in den verschiedensten Formaten und mit unterschiedlich detaillierten Annotationen vor, was die Erforschung des Schriftspracherwerbs erschwert. Ziel dieses Projekts war deshalb die Erstellung eines deutschen Kinderreferenzkorpus mit Texten von und für Grundschulkinder(n) aus drei großen deutschen L1-Korpora (Osnabrücker Bildergeschichtenkorpus, H1 Children's Writing Korpus, Litkey-Korpus) sowie zwei Internetressourcen (Grundschulwiki, Klexikon). Die fünf Subkorpora wurden semi-automatisch mit zahlreichen linguistischen Annotationen angereichert (Transkriptionen, orthographische und grammatische Zielhypothesen, POS-Tags, Dependenzrelationen, Satzgrenzen, direkte Rede, Phoneme, Grapheme, Silben, Morpheme, Rechtschreibfehler, Metadaten) und einheitlich im LearnerXML-Format gespeichert, das für diesen Zweck erweitert wurde. Die vorliegende Dokumentation gibt einen Überblick über die verschiedenen in diesem Projekt durchgeführten Verarbeitungsschritte und Ergebnisse. Sie enthält zudem eine Anleitung, wie weitere Daten zu dem Korpus hinzugefügt werden können.

Inhaltsverzeichnis

1. Einleitung ... 4
2. Korpora ... 5
2.1 Osnabrücker Bildergeschichtenkorpus ... 5
2.2 H1 Children's Writing ... 7
2.3 Litkey-Korpus ... 10
2.4 Grundschulwiki ... 12
2.5 Klexikon ... 14
3. Zielformat ... 16
4. Annotationen ... 25
4.1 Transkription ... 26
4.1.1 Osnabrücker Bildergeschichtenkorpus ... 27
4.1.2 H1 Children's Writing ... 28
4.1.3 Litkey ... 30
4.1.4 Grundschulwiki ... 31
4.1.5 Klexikon ... 33
4.2 Orthographische Zielhypothese ... 35
4.2.1 Osnabrücker Bildergeschichtenkorpus ... 35
4.2.2 H1 Children's Writing ... 37
4.2.3 Litkey ... 39
4.2.4 Grundschulwiki ... 40
4.2.5 Klexikon ... 41
4.3 Grammatische Zielhypothese ... 41
4.3.1 Osnabrücker Bildergeschichtenkorpus ... 42
4.3.2 H1 Children's Writing ... 43
4.3.3 Litkey ... 45
4.3.4 Grundschulwiki ... 45
4.3.5 Klexikon ... 45
4.4 POS-Tags ... 45
4.4.1 Osnabrücker Bildergeschichtenkorpus ... 48
4.4.2 H1 Children's Writing ... 50
4.4.3 Litkey-Korpus ... 52
4.4.4 Grundschulwiki ... 54
4.4.5 Klexikon ... 57
4.5 Dependenz-Parses ... 60
4.5.1 Osnabrücker Bildergeschichtenkorpus ... 63
4.5.2 H1 Children's Writing ... 64

4.5.3 Litkey-Korpus ... 65
4.5.4 Grundschulwki ... 66
4.5.5 Klexikon ... 67
4.6 Satzgrenzen ... 68
4.6.1 Osnabrücker Bildergeschichtenkorpus ... 69
4.6.2 H1 Children's Writing ... 70
4.6.3 Litkey-Korpus ... 70
4.6.4 Grundschulwiki ... 71
4.6.5 Klexikon ... 72
4.7 Direkte Rede ... 73
4.7.1 Osnabrücker Bildergeschichtenkorpus ... 74
4.7.2 H1 Children's Writing ... 75
4.7.3 Litkey-Korpus ... 76
4.7.4 Grundschulwiki ... 78
4.7.5 Klexikon ... 78
4.8 Phoneme, Grapheme, Silben, Morpheme ... 79
4.8.1 Osnabrücker Bildergeschichtenkorpus ... 81
4.8.2 H1 Children's Writing Korpus ... 81
4.8.3 Litkey-Korpus ... 82
4.8.4 Grundschulwiki ... 82
4.8.5 Klexikon ... 82
4.9 Rechtschreibfehler ... 82
4.9.1 Osnabrücker Bildergeschichtenkorpus ... 84
4.9.2 H1 Children's Writing ... 84
4.9.3 Litkey-Korpus ... 85
4.9.4 Grundschulwiki ... 86
4.9.5 Klexikon ... 86
4.10 Metadaten ... 87
4.10.1 Osnabrücker Bildergeschichtenkorpus ... 87
4.10.2 H1 Children's Writing ... 88
4.10.3 Litkey-Korpus ... 90
4.10.4 Grundschulwiki ... 90
4.10.5 Klexikon ... 92
5. Anleitung zur Korpuserweiterung ... 94

1. Einleitung

Ziel dieses Projekts war die Erstellung eines deutschen Kinderreferenzkorpus, d. h. eines Korpus aus von Kindern (und für Kinder) im Grundschulalter geschriebenen Texten. Denn während es eine beständig wachsende Menge von L2-Korpora mit Texten aus dem Zweitspracherwerb von zumeist erwachsenen Sprachenlernern gibt[1], existieren bislang nur sehr wenige deutsche L1-Korpora mit Texten von jungen Schulkindern. Diese Korpora sind zudem häufig nicht frei verfügbar oder liegen in den unterschiedlichsten Formaten und mit unterschiedlich detaillierten Annotationen vor. Dadurch wird die Erforschung des Schriftspracherwerbs unnötig erschwert.

Um dieses Problem zu lösen, wurden im Rahmen dieses Projekts die drei größten deutschen L1-Korpora mit Texten von Kindern aus dem Grundschulalter sowie zwei Internetressourcen zu einem gemeinsamen Korpus zusammengefasst. In das Korpus aufgenommen wurden folgende Textsammlungen:

- Osnabrücker Bildergeschichtenkorpus
- H1 Children's Writing
- Litkey-Korpus
- Grundschulwiki
- Klexikon

Die unterschiedlichen Subkorpora enthalten verschiedene Textgenres (Geschichten, Bildbeschreibungen, Sachtexte) und bilden so ein breites Themenspektrum ab. Eines der Subkorpora besteht zudem aus Texten von Erwachsenen, die speziell für Kinder im Grundschulalter geschrieben sind und somit eine interessante Vergleichsmöglichkeit mit den von Kindern verfassten Texten bspw. hinsichtlich der Komplexität bieten.

In diesem Projekt wurden alle fünf Subkorpora mit verschiedenen Informationen wie Wortarten, Satzgrenzen, Dependenzen, Rechtschreibfehlern, usw. angereichert und die Annotationen in einem einheitlichen XML-Format gespeichert. So sollen die für die Forschung wertvollen Daten, die zuvor einzeln und sehr unterschiedlich aufbereitet waren, einheitlich verarbeitbar und als Referenzkorpus nutzbar gemacht werden.

Die vorliegende Dokumentation gibt einen Überblick über die verschiedenen in diesem Projekt durchgeführten Verarbeitungsschritte und Ergebnisse. In Abschnitt 2 werden zunächst die fünf einbezogenen Subkorpora im Detail vorgestellt, bevor Abschnitt 3 auf das einheitliche Zielformat eingeht. Abschnitt 4 beinhaltet ausführliche Erläuterungen zu allen in diesem Projekt erstellten

[1] Für einen Überblick über Lernerkorpora weltweit s. https://uclouvain.be/en/research-institutes/ilc/cecl/learner-corpora-around-the-world.html

Annotationen. In Abschnitt 5 folgt abschließend eine Anleitung, wie weitere Daten zu dem Korpus hinzugefügt werden können.

2. Korpora

Das Korpus enthält fünf Subkorpora: drei Korpora mit Geschichten oder Bildbeschreibungen sowie zwei lexikonartige Korpora. Vier der fünf Subkorpora enthalten von Kindern geschriebene Texte, das fünfte Subkorpus besteht aus Texten, die von Erwachsenen für Kinder geschrieben wurden.

In den Kapiteln 2.1 bis 2.5 werden die Subkorpora nacheinander vorgestellt. Die Kapitel thematisieren u. a. Herkunft und Inhalt der Subkorpora, Textgenre und Umfang sowie Erstellungszeitraum, Format und Verfügbarkeit der Daten. Außerdem wird auf die relevante Literatur zu den Korpora verwiesen.

2.1 Osnabrücker Bildergeschichtenkorpus

Das Osnabrücker Bildergeschichtenkorpus wurde von Tobias Thelen (2000) zusammengestellt und enthält frei geschriebene Texte zu Bildergeschichten von Grundschülern aus verschiedenen Regionen Deutschlands. Das Korpus umfasst 714 Texte mit 52.013 Token (2.452 Types ohne Groß- und Kleinschreibung). Die Texte liegen im XML-Format vor und sind mit orthographischen und grammatischen Zielhypothesen sowie Überschriften, Zeilenenden, direkter Rede und Satzanfängen annotiert. Darüber hinaus enthält das Korpus Meta-Informationen zu den 705 Kindern und 36 Klassen. Das Korpus ist im Internet für wissenschaftliche Zwecke frei verfügbar. Die folgende Tabelle gibt einen strukturierten Überblick über alle Informationen.

Korpus	
Erstellt von	Tobias Thelen
Erhebungszeitraum	1998/1999; einmalige Erhebung
Textsorte	Freie Verschriftungen zu 3 versch. Bildergeschichten; 9 Erlebnisberichte
Format	XML-Format
Anzahl Texte	714 Texte (1 Text pro Kind, 9 Kinder mit 2 Texten)
Anzahl Token/Types	Orthographische Zielhypothese: • Token: 48.553 • Types: 2.713 (2.527 ohne Groß-/Kleinschreibung) Orthographische Zielhypothese ohne Satzzeichen (POS-Tag enthält kein "$"): • Token: 44.357 • Types: 2.700 (2.514 ohne Groß-/Kleinschreibung) Grammatische Zielhypothese: • Token: 52.013 • Types: 2.637 (2.452 ohne Groß-/Kleinschreibung)

	Grammatische Zielhypothese ohne Satzzeichen (POS-Tag enthält kein "$"): • Token: 44.357 • Types: 2.628 (2.443 ohne Groß-/Kleinschreibung) (alle Werte ohne \h)

Verfasser der Texte	
Anzahl Kinder	705 Kinder aus 36 Klassen
Klassenstufe(n)	1 x Klasse 1, 35 x Klasse 2
Region(en)	Baden-Württemberg, Bayern, Berlin, Hamburg, Hessen, Niedersachsen, Nordrhein-Westfalen, Saarland

Zusätzliche Annotationen und Informationen	
Annotationen	• Orthographische Zielhypothese • Grammatische Zielhypothese • Überschriften • Satzanfänge • Zeilenenden • Direkte Rede
Metadaten	Über die Kinder: • Geschlecht • Herkunft • Muttersprache • Förderung Über die Klassen: • Datum der Verschriftung • Ort u. PLZ • Klassenstufe • Hilfestellung(en) • Unterrichtsform • Orthographie-Lehrkonzept • Sprachbuch • Erfahrung mit freiem Schreiben • Vermittlung orthographischer Regeln • Verfügbarkeit eines Computers

Quellen	
Publikationen	• Thelen, T. (2000). *Osnabrücker Bildergeschichtenkorpus. Version 1.0.0.* • Thelen, T. (2010). *Automatische Analyse orthographischer Leistungen von Schreibanfängern* (Dissertation).
Verfügbarkeit	online unter https://repositorium.uni-osnabrueck.de/handle/urn:nbn:de:gbv:700-201006096307

Ausgangsformat:

Das Korpus besteht aus 36 XML-Dateien, die jeweils alle Texte einer Schulklasse enthalten. Das Wurzel-Element ist <klasse>, seine Tochterelemente sind <info> sowie n-mal <person>. Ein <person>-Element kann mehrere <text>-Kinder enthalten, wenn ein Kind mehr als einen Text geschrieben hat. Dies ist allerdings nur insgesamt neun Mal der Fall. Der übliche Aufbau eines <person>-Elements mit einer <text>-Tochter kann der folgenden Abbildung entnommen werden:

```
<person id="s01"  geschlecht="m">
  <text bezug="bildergeschichte01">
    <titel><sanf>Ein</sanf> armer Hund</titel>
    <body>
      <sanf>Ein</sanf> alter <f ortho="Mann">man</f>
      <f ortho="guckt">ckugt</f> aus dem Haus
      <punkt vorhanden="nein"/>
      er sieht <f typ="g" ortho="einen">ein</f> &ze;
      Hund
      <punkt vorhanden="nein"/>
      er <f ortho="geht">gehd</f> zur <f typ="gks">tür</f>
      <punkt vorhanden="nein"/>
      der Hund <f ortho="rennt">rend</f> <f ortho="weg">veg</f>
      <punkt vorhanden="nein"/>
      der <f ortho="Mann">man</f> &ze; <f ortho="folgt">volgt</f>
      den <f ortho="Fußspuren">vusspuren</f>
      <punkt vorhanden="nein"/>
      <f ortho="dann">dan</f> kommt er zu einer
      <f ortho="Hundehütte">Hundehüte</f>
      <punkt vorhanden="nein"/> &ze;
      da ist der Hund <punkt/>
    </body>
  </text>
</person>
```

Abbildung 1: Ausschnitt aus einer XML-Datei aus dem Osnabrücker Bildergeschichtenkorpus

2.2 H1 Children's Writing

Das H1 Children's Writing Korpus wurde von Kay Berkling (2016) erstellt und enthält Geschichten und Bildbeschreibungen von Grundschülern der Klassen zwei und drei aus Baden-Württemberg. Das Korpus umfasst in seiner originalen Form 996 Texte. Dazu zählen allerdings auch Bildbenennungs-aufgaben, die nur einzelne Wörter und keine Texte enthalten und deshalb vor der Verwendung des Korpus für dieses Projekt aussortiert wurden. Übrig bleiben 824 Texte, die als csv-Dateien vorliegen und 62.914 Token (4.611 Types ohne Groß- und Kleinschreibung) umfassen. Das Korpus enthält die Schreibung der Kinder sowie orthographische und grammatische Zielhypothesen. Darüber hinaus sind Meta-Informationen zu 85 der 88 Kinder vorhanden. Das Korpus ist via LDC (Linguistic Data Consortium) verfügbar. Die folgende Tabelle gibt einen strukturierten Überblick über alle Informa-tionen.

Korpus	
Erstellt von	Kay Berkling
Erhebungszeitraum	2. Halbjahr des Schuljahres 2014/2015 über einen Zeitraum von 12 Wochen

Textsorte	Bildbeschreibung oder Geschichte zu Bildern; die Bilder provozieren viele Doppelkonsonanten- und <ie>-Schreibungen
Format	csv-Format
Anzahl Texte	824 Texte (unter Ausschluss von Bildbenennungsaufgaben)
Anzahl Token/Types	Orthographische Zielhypothese: • Token: 61.447 • Types: 5.083 (4.667 ohne Groß-/Kleinschreibung) Orthographische Zielhypothese ohne Satzzeichen (POS-Tag enthält kein "$"): • Token: 55.402 • Types: 5.077 (4.661 ohne Groß-/Kleinschreibung) Grammatische Zielhypothese: • Token: 62.914 • Types: 5.023 (4.611 ohne Groß-/Kleinschreibung) Grammatische Zielhypothese ohne Satzzeichen (POS-Tag enthält kein "$"): • Token: 55.500 • Types: 5.018 (4.606 ohne Groß-/Kleinschreibung)

Verfasser der Texte	
Anzahl Kinder	88 Kinder von einer Schule
Klassenstufe(n)	Je dreimal Klasse 2 und 3
Alter der Kinder	8 bis 11 Jahre
Region(en)	Baden-Württemberg
Besonderheiten	57 mehrsprachige, 28 einsprachige Kinder

Zusätzliche Annotationen und Informationen	
Annotationen	• Orthographische Zielhypothese • Grammatische Zielhypothese • Eigennamen • Fremdwörter
Metadaten	Verfügbar für 85 Kinder: • Zeitpunkt • Alter • Geschlecht • Klasse u. Klassenstufe • Zuhause gesprochene Sprachen • Schulmaterial für den Deutschunterricht

Quellen	
Publikationen	Berkling, K. (2016) Corpus for Children's Writing with Enhanced Output for Specific Spelling Purposes (2nd and 3rd Grade). In: *Proceedings of LREC*, 3002-3006.
Verfügbarkeit	Via LDC unter https://catalog.ldc.upenn.edu/LDC2016T01

Ausgangsformat:

Das Korpus besteht aus je zwei csv-Dateien pro Text: Eine enthält die Transkription des Textes, d. h. die originale Schreibung, die zweite Datei enthält die Zielhypothese. Korrekturen können sowohl in der Transkription als auch in der Zielhypothese durch die in der folgenden Abbildung aufgeführten Sonderzeichen markiert sein.

Letter- and Word-Level Annotations:	
*	unreadable letter
a_b	a and b should have been written separately
a§b	a and b should have been joined
a=b	missing hyphen
a∼b	wrongly placed hyphen
a−−b	denotes split of word at end of line (not hyphen)
a{n}	n repetitions of word a
a{F}	Foreign word defined by non-German graphemes foreign grapheme-phoneme correspondence
a{G}	grammatical errors not to be analyzed for spelling
a{N}	Names, not analysed with the spell tagger
Sentence Level Annotations	
[§ fW]	an unknown deletion
[§ b]	a known deletion b
[a §]	an insertion a
[a b]	substitution of a for b a is corrected on target side Achieved: [seinne ihre] Target: [seine ihre]
[a b_c]	best guess of word boundary
[a_b c]	kanicht = ka[n nn_n]icht
[a *]	some combinations of letters make up word a the real word can not be identified.
a can include conventions from word-level annotations For example: [rtchen**gdsdfg *] [rtchen**gdsdfg *] or [a{G} b] Numbers (1,2,..): kept as numbers. Words with exaggerated spelling: [Leeeeooooooon Leon].	

Abbildung 2: Annotationen im H1 Children's Writing Korpus (Berkling, 2016, S.4)

Das folgende Beispiel illustriert das Ausgangsformat:

TRANSKRIPTION (DATEI 1):	ZIELHYPOTHESE (DATEI 2):
Theo kletert auf den Klodeckel . Auf einmal hat er denn Rasirschaum und denn Rasirer in der hand . Er Rasirt sich aus_fersen eine Glatze . Alz Mama in das Badezimmer [§ kommt] muss sie lachen . Sie sagt hir ist ein Fisör ein§gekert .	Theo klettert auf den Klodeckel . Auf einmal hat er den Rasierschaum und den Rasierer in der Hand . Er Rasiert sich aus_versehen eine Glatze . Als Mama in das Badezimmer [§ kommt] muss sie lachen . Sie sagt hier ist ein Frisör ein§gekehrt .

Zwei Dateien aus dem Korpus waren falsch benannt. Die Dateinamen wurden unter Zuhilfenahme der Original-Scans korrigiert: H1.KA.G2.16 => H1.KA.G2.14 und H1.KC.G2.21 => H1.KC.G2.22.

2.3 Litkey-Korpus

Das Litkey-Korpus wurde von Hendrike Frieg (2014) erhoben, im Litkey-Projekt[2] aufbereitet und manuell annotiert. Es enthält frei geschriebene Texte zu acht Bildergeschichten von Grundschülern der Klassen zwei bis vier aus Nordrhein-Westfalen. Pro Kind existieren dabei bis zu zehn Texte aus einem Zeitfenster von zwei Jahren. Das Korpus, wie es hier verwendet wurde, entspricht dem Stand vom 12.02.2018 und umfasst 1.862 Texte mit 204.175 Token (5.615 Types ohne Groß- und Kleinschreibung). Das Korpus enthält die Schreibung der Kinder sowie eine orthographische Zielhypothese nach Laarmann-Quante et al. (2017). Darüber hinaus sind zahlreiche Meta-Informationen zu den Kindern vorhanden. Das Korpus wird in Zukunft auf der Webseite des Litkey-Projekts verfügbar sein. Die folgende Tabelle gibt einen strukturierten Überblick über alle Informationen.

Korpus	
Stand	12.02.2018
Erstellt von	Hendrike Frieg, Laarmann-Quante et al.
Erhebungszeitraum	Februar 2010 bis Juli 2012
Textsorte	Geschichten zu 8 verschiedenen Bildergeschichten aus Schroff (2000)
Format	csv-Format
Anzahl Texte	1.862 Texte
Anzahl Token/Types	Orthographische Zielhypothese: • Token: 204.175 • Types: 6.196 (5.615 ohne Groß-/Kleinschreibung)

[2] https://www.linguistics.ruhr-uni-bochum.de/litkey/Scientific/Corpusanalysis/index.html

	Orthographische Zielhypothese ohne Satzzeichen (POS-Tag enthält kein "$"): • Token: 182.714 • Types: 6.159 (5.578 ohne Groß-/Kleinschreibung) (alle Werte ohne \h)

Verfasser der Texte	
Anzahl Kinder	251 Kinder aus 15 Schulklassen
Klassenstufe(n)	2 bis 4
Alter der Kinder	7 bis 13 Jahre
Region(en)	Nordrhein-Westfalen

Zusätzliche Annotationen und Informationen	
Annotationen	• Orthographische Zielhypothese • Zeilenenden • Überschriften
Metadaten	Unter anderem: • Testzeitpunkt • Alter • Geschlecht • Klasse u. Klassenstufe • Schule • Zuhause gesprochene Sprachen • Förderung

Quellen	
Publikationen	• Frieg, H. (2014). *Sprachförderung im Regelunterricht der Grundschule: Eine Evaluation der Generativen Textproduktion* (Doktorarbeit). • Laarmann-Quante, R., Ortmann, K., Ehlert, A., Betken, C., Dipper, S., & Knichel, L. (2017). *Guidelines for the Manual Transcription and Orthographic Normalization of Handwritten German Texts Produced by Primary School Children.* Bochumer Linguistische Arbeitsberichte (BLA), Vol. 20.

Ausgangsformat:

Das Korpus besteht aus einer zweispaltigen csv-Datei pro Text: Die linke Spalte enthält die Transkription des Textes, d. h. die originale Schreibung, die rechte Spalte enthält die Zielhypothese. Korrekturen können sowohl in der Transkription als auch in der Zielhypothese durch Sonderzeichen wie Unterstrich _, senkrechter Strich |, Fragezeichen ? oder Tilde ~ markiert sein.

Das folgende Beispiel illustriert das Ausgangsformat:

ver	Der	^	
kleine	kleine	klebte	klebte
Hund	Hund	sie	sie
Dodo	Dodo	auf	auf
\h	\h	die	die
Lea	Lea	Wänder	~Wänder
suchte	suchte	Plakate	Plakate
ihren	ihren	^	
Hund	Hund	mit	mit
Dodo	Dodo	Dodos	Dodos
,	,	Fotos	Fotos
dafür	dafür		

2.4 Grundschulwiki

Das Grundschulwiki wurde von der Zentrale für Unterrichtsmedien im Internet e.V. mit der Intention der Schulung der Medienkompetenz Heranwachsender gegründet. Im Rahmen verschiedener Unterrichtsprojekte werden seit 2005 Sachtexte verschiedener Themengebiete von Grundschülern des deutschsprachigen Raumes verfasst und anschließend der Öffentlichkeit zur Verfügung gestellt. Die im Rahmen dieser Arbeit genutzten Texte wurden im April 2017 der Webseite entnommen. Das Korpus umfasst 200 Texte mit 24.574 Token (5.583 Types ohne Groß- und Kleinschreibung). Das Korpus enthält keine Annotationen. Die folgende Tabelle gibt einen strukturierten Überblick über alle Informationen.

Korpus	
Erstellt von	Zentrale für Unterrichtsmedien im Internet e.V.
Erhebungszeitraum	Dezember 2005 bis heute
Textsorte	Sachtexte von Kindern für Kinder
Format	XML-Format
Anzahl Texte	200 (Stand: April 2017)
Anzahl Token/Types	Orthographische Zielhypothese: • Token: 24.574 • Types: 5.820 (5.583 ohne Groß-/Kleinschreibung)

	Orthographische Zielhypothese ohne Satzzeichen (POS-Tag enthält kein "$"): • Token: 20.843 • Types: 5.801 (5.564 ohne Groß-/Kleinschreibung) (alle Werte ohne \h)

Verfasser der Texte	
Anzahl Kinder	nicht eindeutig bestimmbar
Klassenstufe(n)	1.-4. Klasse
Region(en)	Deutschsprachiger Raum
Besonderheiten	Texte werden durch Erzieher und Lehrkräfte kontrolliert

Zusätzliche Annotationen und Informationen	
Annotationen	Es liegen keine Annotationen vor.
Metadaten	Verfügbar für das Korpus: • Titel • ID • Revision-ID • Parent-ID • Datum • Autor (ID, Benutzername) • Format • Bilddateien • Kategorien • Referenzen • Kommentar

Quellen	
Verfügbarkeit	Via CC BY-SA 3.0 DE unter https://grundschulwiki.zum.de/wiki/

Ausgangsformat:

Das Korpus besteht aus einer XML-Datei, die alle Texte enthält. Innerhalb des XML-Formats liegen die Texte in Form von <page>-Elementen vor, diesen sind Titel, ID, Revision-ID, Parent-ID, Zeitpunkt, Verfasser und dessen ID sowie Hinweise zu entsprechenden Bearbeitungen und dem verwendeten Format untergeordnet. Innerhalb des Textes befinden sich in eckigen Klammern Referenzen zu anderen Texten und ebenso Informationen zu zugehörigen Bilddateien.

Der folgende Ausschnitt zeigt das Element des Textes "Achkarren":

```
  <page>
    <title>Achkarren</title>
    <ns>0</ns>
    <id>2210</id>
    <revision>
      <id>9939</id>
      <parentid>6151</parentid>
      <timestamp>2008-12-17T13:54:26Z</timestamp>
      <contributor>
        <username>J.Leupold</username>
        <id>9</id>
      </contributor>
      <text xml:space="preserve" bytes="360">[[Bild:Achkarren.jpg|thumb]]
'''Achkarren''' ist ein Weindorf, deswegen ist das Dorf auch von Reben umgeben.
Die Rebfläche ist insgesamt ca. 180 Hektar groß. Achkarren ist ein nicht so großes Dorf,
aber es gibt eine Schule, einen Kindergarten und es hat 850 Einwohner.
Ein Besuch lohnt sich immer!
Achkarren ist ein &amp;quot;musikalisches Dorf&amp;quot;.
{{Rücksprung|A}}</text>
      <sha1/>
      <model>wikitext</model>
      <format>text/x-wiki</format>
    </revision>
  </page>
```

Abbildung 3: <page>-Element zum Text "Achkarren" aus der Grundschulwiki-XML-Datei

2.5 Klexikon

Im Auftrag von Wikimedia entwickelten Michael Schulte und Ziko van Dijk das Klexikon, um ein Wikipedia für Kinder zu schaffen. Im Rahmen des Projektes wählten sie ab Dezember 2014 Autoren des deutschsprachigen Raumes aus, um gemeinsam Sachtexte für Kinder zu schreiben. Die im Rahmen dieser Arbeit genutzten Texte wurden im April 2017 der Webseite entnommen, das Korpus umfasst 924 Texte mit 308.349 Token (20.493 Types ohne Groß- und Kleinschreibung). Es enthält keine Annotationen. Die folgende Tabelle gibt einen strukturierten Überblick über alle Informationen.

Korpus	
Erstellt von	Michael Schulte & Ziko van Dijk
Erhebungszeitraum	Dezember 2014 bis heute
Textsorte	Sachtexte für Kinder (verfasst von ausgewählten Autoren)
Format	XML-Format
Anzahl Texte	924 (Stand: April 2017)
Anzahl Token/Types	Orthographische/grammatische Zielhypothese: • Token: 308.349 • Types: 21.695 (20.493 ohne Groß-/Kleinschreibung)

	Orthographische/grammatische Zielhypothese ohne Satzzeichen (POS-Tag enthält kein "$"): • Token: 266.180 • Types: 21.669 (20.468 ohne Groß-/Kleinschreibung)

Zusätzliche Annotationen und Informationen	
Annotationen	Es liegen keine Annotationen vor.
Metadaten	Verfügbar für das Korpus: • Titel • ID • Revision-ID • Parent-ID • Datum • Autor (ID, Benutzername) • Format • Bilddateien • Kategorien • Referenzen • Kommentar

Quellen	
Publikationen	Schulte, M., & van Dijk, Z. (2015). *Projekt Freies Kinderlexikon. Konzept für eine Kindgerechte Wiki-Enzyklopädie*.
Verfügbarkeit	Via CC BY-SA 3.0 DE unter https://klexikon.zum.de/wiki/

Ausgangsformat:

Ebenso wie das Grundschulwiki besteht das Korpus aus einer XML-Datei, die alle Texte enthält. Innerhalb des XML-Formats liegen die Texte als <page>-Knoten des Hauptbaumes vor, welchen Titel, ID, Revision-ID, Parent-ID, Zeitpunkt, Verfasser und dessen ID sowie Hinweise zu entsprechenden Bearbeitungen und dem verwendeten Format untergeordnet sind. Innerhalb des Textes befinden sich in eckigen Klammern Referenzen zu anderen Texten und ebenso Informationen zu zugehörigen Bilddateien.

Der folgende Ausschnitt zeigt das Element des Textes "Rathaus":

```
  <page>
    <title>Rathaus</title>
    <ns>0</ns>
    <id>639</id>
    <revision>
      <id>49787</id>
      <parentid>36001</parentid>
      <timestamp>2017-03-06T20:10:25Z</timestamp>
      <contributor>
        <username>Ziko van Dijk</username>
        <id>7</id>
      </contributor>
      <text xml:space="preserve" bytes="2254">[[Datei:Berner Rathaus 2007-09.jpg|mini|Das Rathaus der [[Stadt]] [[Bern]]]
Rathaus nennt man das [[Haus]], in dem die [[Politiker]] und Mitarbeiter einer [[Stadt]] oder [[Gemeinde]] " [[Politik]]
machen“ und arbeiten. Hier trifft sich der Rat, das heißt die Volksvertretung der Stadt. So ein [[Parlament]] entscheidet
über die wichtigsten Dinge in der Stadt. Außerdem arbeiten dort der [[Bürgermeister]] und die Verwaltung: der
Chef der Stadt und seine Mitarbeiter.

In manchen Städten sind die Politiker und Mitarbeiter in ein neues Gebäude umgezogen.
Das alte Gebäude heißt dann vielleicht immer noch " Rathaus“ oder " Altes Rathaus“.
Manchmal sagt man " Rathaus“ und meint damit nicht wirklich das Gebäude, sondern die Politiker der Stadt oder die Stadtverwaltung.

== Was passiert in einem Rathaus? ==
In [[Europa]] gibt es schon seit dem [[Mittelalter]] solche Rathäuser,
wie man sie heute kennt. Allerdings hat man sie oft nicht nur für den Rat verwendet.
Ein Ort war eine Stadt unter anderem, wenn man dort einen Markt abhalten durfte.
Daher fand der Markt oft im Rathaus statt. Im Rathaus gab oder gibt es außerdem vielleicht ein Restaurant.

Ein Rathaus braucht einen großen Saal, in dem der Rat sich treffen kann.
Bei einer großen Stadt wie [[Hamburg]] sind das 121 Ratsmitglieder.
Der Rat kommt aber nicht ständig zusammen. Darum nutzen viele Rathäuser ihren Saal noch für etwas anderes.
Zum Beispiel spielt man in Konzerten [[Musik]], oder es finden Treffen von Wissenschaftlern oder Künstlern statt.

&lt;gallery&gt;
Datei:Wilhelmshaven townhall front facing north Wilhelmshaven Germany 03.jpg|Das Rathaus von Wilhelmshaven, in
[[Niedersachsen]], ist gleichzeitig ein Wasserturm.
Datei:Rathaus and Marienplatz from Peterskirche - August 2006.zugeschnitten.jpg|Neues Rathaus in [[München]]:
Vor über hundert Jahren hat man es im Stil der [[Gotik]] gebaut.
Datei:City Hall, Toronto, Ontario.jpg|Dieses moderne Rathaus steht in der Stadt Toronto in [[Kanada]].
Datei:Ratusz2noc.jpg|Das Rathaus von Breslau, in [[Polen]], stammt aus dem [[Mittelalter]].
Datei:Festsaal Rathaus Vienna 2007.jpg|Festsaal im Rathaus von [[Wien]].
&lt;/gallery&gt;
{{Mehr}}
[[Kategorie:Klexikon-Artikel]]
[[Kategorie:Politik und Gesellschaft]]</text>
      <sha1>8yvha21b2unzfwrhq7s7zppug24xtqe</sha1>
      <model>wikitext</model>
      <format>text/x-wiki</format>
    </revision>
  </page>
```

Abbildung 4: <page>-Element zum Text "Rathaus" aus der Klexikon-XML-Datei

3. Zielformat

Wie eingangs erwähnt, war ein Hauptziel des Projekts die Bereitstellung verschiedener Korpora in einem einheitlichen Format. Bislang sind für die Nutzung jedes Subkorpus eigene Methoden und Programme nötig, da das eine Korpus als csv-Datei vorliegt, das nächste in einem XML-Format und das dritte in verschiedenen Textdateien. Durch die Vereinheitlichung soll eine einfache Nutzbarkeit und Vergleichbarkeit aller Korpora ermöglicht werden.

Als gemeinsames Zielformat wurde das speziell für die Darstellung von Lerner-Texten entwickelte LearnerXML-Format (Laarmann-Quante, Knichel, Dipper & Betken, 2016) ausgewählt. Dieses bietet zwei wesentliche Vorteile gegenüber den Ausgangsformaten der Subkorpora. Erstens ermöglicht es die strukturierte Speicherung sehr vieler über den Text hinausgehender Annotationen inklusive Bezügen unterhalb der Wortebene (wie z. B. Phoneme), was in einem reinen Text- oder csv-Format kaum zu bewältigen ist. Zweitens besticht es durch seine logische, eindeutige Struktur und ermöglicht

anders als bspw. das XML-Format des Osnabrücker Bildergeschichtenkorpus eine einfache Abfrage aller Informationen.

Da in diesem Projekt alle Korpora mit bisher nicht vorhandenen Annotationen (POS-Tags, Dependenzen, Meta-Informationen etc.) angereichert wurden, musste das LearnerXML-Format jedoch zunächst erweitert werden, um auch diese Informationen aufnehmen zu können. Für eine einheitliche Verarbeitung aller Subkorpora wurde außerdem ein genormtes csv-Format entwickelt, das alle neu annotierten Informationen aufnehmen kann, bevor diese in das LearnerXML-Format eingebunden werden. Das folgende Kapitel beschreibt zunächst den Aufbau dieses csv-Formats, bevor das erweiterte LearnerXML-Format und die korpusspezifische Struktur des neuen Meta-Headers erläutert werden.

Einheitliche Zwischenspeicherung der Daten – die csv-Datei

Wie bereits in Kapitel 2 erläutert, liegen die Subkorpora alle in unterschiedlichen Ausgangsformaten vor und auch die verschiedenen Annotationen (Zielhypothesen, POS-Tags, Dependenzen, etc.) sind infolge der eingesetzten Tools (s. Kapitel 4) auf verschiedene Dateien verteilt. Um den Aufwand bei der Übertragung in das LearnerXML-Format zu minimieren, wurde daher festgelegt, dass alle vorhandenen Informationen zunächst in Form einer genormten csv-Datei abgespeichert werden. Die Tabulator-getrennten Spalten der Datei enthalten dabei folgende Informationen (hier der Lesbarkeit halber vertikal dargestellt):

1. Token-ID (tok1 bis tokN)
2. Transkription
3. Orthographisches Zielwort
4. Grammatisches Zielwort
5. STTS-POS-Tag
6. Satzanfang des Stanford-Parsers (true/false)
7. Satzanfang nach grammatischer Zeichensetzung (true/false)
8. Direkte Rede (begin/end)
9. Dependenz-Relation Dipper
10. Head-Token-ID Dipper (tokN oder tok0)
11. Dependenz-Relation Tüba
12. Head-Token-ID Tüba (tokN oder tok0)
13. Dependenz-Relation Universal-Dependencies
14. Head-Token-ID Universal-Dependencies (tokN oder tok0)

Da nicht jedes Subkorpus über Informationen aller Kategorien verfügt, werden entsprechend fehlende Spalten oder Zellen leer gelassen. Die folgenden Abbildungen zeigen einen Ausschnitt aus einer csv-

Datei des Osnabrücker Bildergeschichtenkorpus und des Grundschulwikis. Während im Osnabrücker Bildergeschichtenkorpus alle Informationen vorhanden sind, fehlen für das Grundschulwiki die grammatische Zielhypothese und die darauf basierenden Satzgrenzen sowie die Annotation der direkten Rede.

tok63			"	$(	true	true	begin	tok71	punct	tok64	-PUNCT-	tok71	punct
tok64	Na	Na	Na	ITJ	false	false		tok71	xd-cc	tok0	ROOT	tok71	advmod
tok65	gut	gut	gut	ADV	false	false		tok71	xd-mod	tok66	ADV	tok71	advmod
tok66	ich	ich	ich	PPER	false	false		tok71	subj	tok67	SUBJ	tok71	nsubj
tok67	haBe	habe	habe	VAFIN	false	false		tok71	aux	tok64	-UNKNOWN-		tok71	aux
tok68	So	so	so	ADV	false	false		tok69	mod	tok69	ADV	tok71	advmod
tok69	File	viele	viele	PIAT	false	false		tok71	det	tok71	DET	tok71	amod
tok70	^												
tok71	maten	Matten	Matten	NN	false	false		tok0	root	tok67	OBJA	tok0	root
tok72			.	$.	false	false		tok71	punct	tok71	-PUNCT-	tok71	punct
tok73	er	er	er	PPER	true	true		tok76	subj	tok74	SUBJ	tok76	nsubj
tok74	kan|	kann	kann	VMFIN	false	false		tok76	aux	tok0	ROOT	tok76	aux
tok75	sie	sie	sie	PPER	false	false		tok76	subj	tok76	OBJA	tok76	nsubj
tok76	Behalte	behalten		behalten		VVINF	false	false		tok0	root	tok74	AUX	tok0	root
tok77	^														
tok78			.	$.	false	false		tok76	punct	tok76	-PUNCT-	tok76	punct
tok79			"	$(	false	false	end	tok76	punct	tok76	-PUNCT-	tok76	punct
tok80	ENDE	ENDE	ENDE	NN	true	true		tok0	root	tok0	ROOT	tok0	root

Abbildung 5: Ausschnitt aus einer csv-Datei des Osnabrücker Bildergeschichtenkorpus

tok6	Man	Man	PIS	true	tok7	subj	tok7	SUBJ	tok7	nsubj
tok7	braucht	braucht	VVFIN	false	tok0	root	tok0	ROOT	tok0	root
tok8	lange	lange	ADV	false	tok7	mod	tok7	ADV	tok7	advmod
tok9	,	,	$,	false	tok7	punct	tok8	-PUNCT-	tok7	punct
tok10	um	um	KOUI	false	tok13	pcase	tok13	PP	tok13	mark
tok11	das	das	PDS	false	tok13	obj	tok10	PN	tok13	dobj
tok12	zu	zu	PTKZU	false	tok13	ccase	tok13	PART	tok13	aux
tok13	lernen	lernen	VVINF	false	tok7	mod	tok7	OBJI	tok7	advcl
tok14	.	.	$.	false	tok7	punct	tok13	-PUNCT-	tok7	punct
tok15	Es	Es	PPER	true	tok16	subj	tok16	SUBJ	tok18	nsubj
tok16	ist	ist	VAFIN	false	tok0	root	tok0	ROOT	tok18	cop
tok17	so	so	ADV	false	tok18	mod	tok18	ADV	tok18	advmod
tok18	ähnlich	ähnlich	ADJD	false	tok16	pred	tok16	PRED	tok0	root
tok19	wie	wie	KOKOM	false	tok20	pcase	tok18	KOM	tok20	case
tok20	Tennis	Tennis	NN	false	tok18	mod	tok19	CJ	tok18	nmod
tok21	.	.	$.	false	tok16	punct	tok20	-PUNCT-	tok18	punct

Abbildung 6: Ausschnitt aus einer csv-Datei des Grundschulwikis

Einheitliches Zielformat – erweitertes LearnerXML

Mithilfe eines Skriptes wurden die Informationen aus den genormten csv-Dateien ins LearnerXML-Format übertragen. Da die Korpora in diesem Projekt mit bislang nicht verfügbaren Kategorien wie POS-Tags und Dependenzen annotiert wurden, musste das LearnerXML-Format dafür zunächst erweitert werden, um auch die neuen zusätzlichen Informationen aufnehmen zu können. Die folgende Abbildung zeigt die Darstellung des Wortes *<fäld> im originalen LearnerXML-Format:

```
<?xml version="1.0" ?>
<tokens id="test">
   <token id="tok1" orig="fäld" target="fällt"
          foreign_target="false" exist_orig="false">
      <characters_orig>
         <char_o id="o1">f</char_o>
         <char_o id="o2">ä</char_o>
         <char_o id="o3">l</char_o>
         <char_o id="o4">d</char_o>
      </characters_orig>
      <characters_target>
         <char_t id="t1">f</char_t>
         <char_t id="t2">ä</char_t>
         <char_t id="t3">l</char_t>
         <char_t id="t4">l</char_t>
         <char_t id="t5">t</char_t>
      </characters_target>
      <characters_aligned>
         <char_a id="a1" o_range="o1" t_range="t1"/>
         <char_a id="a2" o_range="o2" t_range="t2"/>
         <char_a id="a3" o_range="o3" t_range="t3..t4"/>
         <char_a id="a4" o_range="o4" t_range="t5"/>
      </characters_aligned>
      <phonemes_target>
         <phon_t id="p1" t_range="t1">f</phon_t>
         <phon_t id="p2" t_range="t2">E</phon_t>
         <phon_t id="p3" t_range="t3..t4">l</phon_t>
         <phon_t id="p4" t_range="t5">t</phon_t>
      </phonemes_target>
      <graphemes_target>
         <gra id="g1" range="t1"/>
         <gra id="g2" range="t2"/>
         <gra id="g3" range="t3"/>
         <gra id="g4" range="t4"/>
         <gra id="g5" range="t5"/>
      </graphemes_target>
      <syllables_target>
         <syll id="s1" range="t1..t5" type="stress" plausible_orig ="true"/>
      </syllables_target>
      <morphemes_target>
         <mor id="m1" range="t1..t4" type="NN"/>
         <mor id="m2" range="t5..t5" type="INFL"/>
      </morphemes_target>
      <errors>
         <err range="a3" cat="SL:Cdouble_beforeC" phon_orig_ok="true"
              morph_const="neces"/>
         <err range="a4" cat="MO:hyp_final_devoice" phon_orig_ok="true"
              morph_const="neces"/>
      </errors>
   </token>
</tokens>
```

Abbildung 7: Annotation des Wortes *<fäld> im originalen LearnerXML-Format (Laarmann-Quante et al., 2016, S.39)

Aus diesem Projekt sind fünf neue Annotationskategorien hervorgegangen, die auf verschiedenen Ebenen in das gegebene LearnerXML-Format eingefügt wurden. Im Folgenden werden zunächst die Entscheidungen und Bestimmungen bzgl. jeder Annotation kurz thematisiert, bevor ein Beispiel für das erweiterte LearnerXML-Format folgt.

- **Grammatische Zielhypothese**

  Die gesamte Darstellung im LearnerXML-Format basiert auf der originalen Schreibung (*orig*) und der orthographischen Zielhypothese (*target*). Die grammatische Zielhypothese wurde analog hierzu als Attribut *gram_target* zum Token hinzugefügt. Der Wert des Attributs kann, wenn das Korpus über diese Information verfügt, ein grammatisches Zielwort oder ein leerer String sein. Ist ein Korpus nicht mit grammatischen Zielhypothesen annotiert, kann das Attribut standardmäßig wegfallen.

- **STTS-POS-Tags**

  Die Annotation der POS-Tags, die im Unterkapitel 4.4 beschrieben wird, erfolgte zum ersten Mal im Rahmen dieses Projekts. Da das Tagging nach dem STTS-Tagset erfolgt, wurde als Attributname *stts* ausgewählt und die Annotation zu den Attributen des Tokens hinzugefügt. In Anlehnung an die Arbeit in diesem Projekt ist dies mittlerweile standardmäßig vorgesehen.

- **Satzgrenzen**

  Die Annotation der Satzgrenzen wurde (ebenso wie diejenige der direkten Rede und der Dependenzen) am Token durchgeführt. Hier wurde bewusst gegen die Annotation auf der Satzebene entschieden, da dies eine Veränderung des gesamten tokenbasierten LearnerXML-Formats bedeuten würde. Die bisher aus einer Wurzel <tokens> und den einzelnen Tochterelementen <token> bestehende Aufreihung der Wörter müsste so abgeändert werden, dass zusätzliche Satzelemente eingefügt und die Token diesen entsprechend untergeordnet werden. Dies erfordert jedoch die einheitliche Annotation der Satzgrenzen in allen Subkorpora, welche aktuell nicht gegeben ist. Abschnitt 4.6 erläutert die Schwierigkeiten der Satzgrenzenbestimmung gerade in den von Kindern verfassten Texten genauer. Hier kann nur kurz angedeutet werden, dass eine Zuordnung der Wörter zu Sätzen häufig nicht trivial ist, weshalb eine sinnvolle Annotation der Satzgrenzen (ebenso wie der direkten Rede und der Dependenzen) aktuell auf der Token-Ebene erfolgen muss.

  Im Rahmen dieses Projekts wurden für jedes Subkorpus ein bis zwei verschiedene Satzgrenzenannotationen erzeugt. Diese wurden aus den genannten Gründen als Attribute den Token-Elementen hinzugefügt. Es gibt hierbei zum einen das Attribut *SentBeginStanford*, welches die Satzgrenzen des Stanford-Parsers enthält, und zum anderen das Attribut *SentBeginPunct*, welches auf der grammatischen Interpunktion basiert. Beide Attribute können den Wert "true" haben, wenn ein Token der Satzanfang ist. Ansonsten ist der Wert "false" bzw. das Attribut wird der Übersichtlichkeit halber ausgelassen. Ist eine Satzgrenzenannotation für ein Korpus nicht vorhanden, bspw. weil keine grammatische Interpunktion existiert, entfällt das entsprechende Attribut ebenfalls.

- **Direkte Rede**

  Aus den unter "Satzgrenzen" angeführten Gründen wurden auch der Anfang bzw. das Ende der direkten Rede als Attribut am Token, das die direkte Rede einleitet oder beendet, realisiert. Da es sich um eine manuelle Annotation handelt, wurde das Attribut als *directSpeechMan* bezeichnet und es kann die Werte "begin" oder "end" haben.

- **Dependenzen nach drei versch. Guidelines**

  Die Annotation der Dependenzen wurde aus den unter "Satzgrenzen" genannten Gründen ebenfalls am Token gespeichert. Hierfür wurde analog zur Annotation der Phoneme, Silben, Fehler, etc. ein neues Tochterelement <dependencies> eingeführt, das die drei Sub-Elemente <dep_dipper>, <dep_tueba> und <dep_ud> für die Annotation nach den verschiedenen Dependenzguidelines enthalten kann. Jedes der Sub-Elemente hat wiederum die Attribute *head* und *rel*, in denen die Dependenzrelation und die ID des Head-Tokens gespeichert werden. Annotationen nach weiteren Guidelines könnten entsprechend ergänzt werden. Für das Root-Token ist die Head-ID "tok0" vorgesehen, da die Nummerierung der Token in LearnerXML mit "tok1" beginnt. Das einzige Token, das aktuell keine Dependenzannotation erhält, ist das Meta-Sonderzeichen ^ für das Zeilenende. In Zukunft wird dieses Sonderzeichen (und ebenso das Überschriftsende \h) möglicherweise durch ein Attribut am Token ersetzt, sodass hier keine Sonderregelung mehr zu beachten wäre.

Die folgende Abbildung zeigt das erweiterte LearnerXML:

```
<text id="Dateiname">
    <meta subcorpus="Korpusname">
        <text>
            ...
        </text>
        <author>
            ...
        </author>
    </meta>
    <tokens>
        <token id="tok1" orig="ain" target="ein" gram_target="Eine" exist_orig="false" stts="ART"
        beginSentStanford="true" beginSentPunct="true">
            <characters_orig>
                <char_o id="o1">a</char_o>
                ...
            </characters_orig>
            <characters_target>
                <char_t id="t1">e</char_t>
                ...
            </characters_target>
            <characters_aligned>
                <char_a id="a1" o_range="o1" t_range="t1"/>
                ...
            </characters_aligned>
            <phonemes_target>
                <phon_t id="p1" t_range="t1..t2">aI</phon_t>
                ...
            </phonemes_target>
            <graphemes_target>
                <gra id="g1" range="t1"/>
                ...
            </graphemes_target>
            <syllables_target>
                <syll id="s1" range="t1..t3" type="stress"/>
            </syllables_target>
            <morphemes_target>
                <mor id="m1" range="t1..t3" type="ART"/>
            </morphemes_target>
            <errors>
                <err cat="PGI:literal" morph_const="na" phon_orig_ok="true" range="t1..t2">
            </errors>
            <dependencies>
                <dep_dipper head="tok3" rel="det"/>
                <dep_tueba head="tok3" rel="DET"/>
                <dep_ud head="tok2" rel="det"/>
            </dependencies>
        </token>
        ...
    </tokens>
</text>
```

Abbildung 8: Erweitertes LearnerXML-Fomat am Beispiel der Schreibung *<ain>

Einführung des Meta-Headers

Ebenfalls erweitert wurde das ursprüngliche LearnerXML durch individuelle Meta-Header. Da je nach Subkorpus andere Informationen zur Verfügung stehen, konnte kein vollständig einheitlicher Header erstellt werden. Um dennoch möglichst viele der vorhandenen Informationen in dem Header zu speichern, wurde dieser Abschnitt des XML-Baums individuell für jedes Subkorpus angepasst. Dabei wurden Templates verwendet, die von einem Skript automatisch mit den vorhandenen Informationen gefüllt werden können. Daraus folgt, dass Elemente im Meta-Header, wenn die Information für den Text nicht vorhanden ist, leer sein können. Dies entspricht auch dem Vorgehen im restlichen LearnerXML und bietet den Vorteil einer einheitlichen Struktur des XML-Baums über alle Texte eines Subkorpus hinweg.

Der grundlegende Aufbau des Meta-Headers für die drei Korpora Osnabrücker Bildergeschichtenkorpus, H1 Children's Writing und Litkey sieht wie folgt aus:

```
<meta subcorpus="Korpusname">
        <text>
                ...
        </text>
        <author>
                ...
        </author>
</meta>
```

Die Sub-Elemente <text> und <author> enthalten hierbei korpusspezifische Informationen zu dem jeweiligen Text und seinem Verfasser. Aufgrund ihres anderen Ursprungs wurde beim Grundschulwiki und beim Klexikon auf das <author>-Element verzichtet, da der Benutzername bzw. die ID hier weder Rückschluss auf das Alter noch auf den tatsächlichen Namen des Verfassers gibt. Das Grundgerüst des Meta-Headers ist für diese beiden Korpora identisch:

```
<meta subcorpus="Korpusname">
        <text>
                …
        </text>
</meta>
```

Allen Meta-Headern gemein ist das Attribut *subcorpus* im <meta>-Element, welches den Korpusnamen enthält. Der Vollständigkeit halber folgt eine Abbildung der Templates zu allen Meta-Headern inklusive aller Sub-Elemente und ggf. Default-Werte:

Meta-Header Osnabrücker Bildergeschichtenkorpus

```
<meta subcorpus="Osnabrücker Bildergeschichtenkorpus">
    <text>
        <topic></topic>
        <date></date>
        <federal_state></federal_state>
        <grade></grade>
    </text>
    <author>
        <id></id>
        <gender></gender>
        <origin></origin>
        <languages></languages>
        <class></class>
        <additional_tutoring></additional_tutoring>
        <comments></comments>
    </author>
</meta>
```

Meta-Header H1 Children's Writing

```
<meta subcorpus="H1 Children's Writing">
    <text>
        <topic></topic>
        <date>2015</date>
        <test_time></test_time>
        <federal_state>Baden-Württemberg</federal_state>
        <grade></grade>
    </text>
    <author>
        <id></id>
        <gender></gender>
        <age></age>
        <languages></languages>
    </author>
</meta>
```

Meta-Header Litkey-Korpus

```
<meta subcorpus="Litkey">
    <text>
        <topic></topic>
        <date></date>
        <federal_state>Nordrhein-Westfalen</federal_state>
        <grade></grade>
    </text>
```

```
    <author>
        <id></id>
        <gender></gender>
        <age></age>
        <languages></languages>
        <school></school>
        <class></class>
        <additional_tutoring></additional_tutoring>
    </author>
</meta>
```

Meta-Header Grundschulwiki

```
<meta subcorpus="Grundschulwiki">
    <text>
        <topic></topic>
        <date></date>
        <revision_id></revision_id>
        <parent_id></parent_id>
        <categories></categories>
        <links></links>
    </text>
</meta>
```

Meta-Header Klexikon

```
<meta subcorpus="Klexikon">
    <text>
        <topic></topic>
        <date></date>
        <revision_id></revision_id>
        <parent_id></parent_id>
        <categories></categories>
        <links></links>
    </text>
</meta>
```

4. Annotationen

Existierende deutschsprachige L1-Korpora enthalten i. d. R. nur wenige Annotationen, die über die Transkription, d. h. die originale Schreibung der Schüler, hinausgehen. Dazu zählen Zielhypothesen orthographischer und/oder grammatischer Natur sowie Fehlerannotationen (z. B. Fay, 2010). Eine Annotation mit linguistischen Merkmalen wie Phonemen, Silben, Morphemen, Wortarten, syntaktischen Relationen, etc. existiert in Korpora dieser Art bislang nicht und wurde hier zum ersten Mal auf die verschiedenen Subkorpora angewendet.

Der Auswahl der spezifischen Annotationen für dieses Korpus liegen verschiedene Überlegungen zugrunde. Allen Annotationen ist gemein, dass sie (fast) vollständig automatisch durchführbar und somit auf große Textmengen anwendbar sind. Die Annotation von Wortarten und Dependenzen ist bei standardsprachlichen Korpora wie dem TüBa-D/Z-Korpus (Telljohann, Hinrichs & Kübler, 2004) die Regel und eine Anreicherung der besonderen Textsorte Kindertext mit diesen Standard-Annotationen ermöglicht einen Vergleich mit anderen (standardsprachlichen) Textsorten z. B. bzgl. der Satzkomplexität. Die Annotation von Worteigenschaften wie Phonemen, Graphemen, etc. kann wiederum bspw. für die Fehleranalyse verwendet werden, wodurch neue Analysen wie z. B. zur Entwicklung der Rechtschreibkompetenz ermöglicht werden, die andernfalls mit großem manuellen Aufwand verbunden wären.

Dieses Kapitel gibt einen Überblick über die im Korpus enthaltenen Annotationen und erläutert im Detail, wie sie erstellt wurden und welche Probleme es u. a. aufgrund der Textsorte ggf. mit den einzelnen Annotationen gibt. Folgende Annotationen werden in dieser Reihenfolge thematisiert:

4.1 Transkription
4.2 Orthographische Zielhypothese
4.3 Grammatische Zielhypothese
4.4 POS-Tags
4.5 Dependenz-Parses
4.6 Satzgrenzen
4.7 direkte Rede
4.8 Phoneme, Grapheme, Silben, Morpheme
4.9 Rechtschreibfehler
4.10 Metadaten

4.1 Transkription

Der folgende Abschnitt thematisiert die Transkription (lat. *trans* ‚hinüber' und *scribere* ‚schreiben'). Der Begriff meint hier die originale Schreibung des Textes und rührt daher, dass von Grundschulkindern geschriebene Texte i. d. R. zunächst von einer analogen in eine digitale Form überführt, d. h. abgeschrieben, werden müssen. Bei genuin digitalen Korpora wie dem Grundschulwiki oder Klexikon entfällt eine solche Überführung. Um eine einheitliche Terminologie beizubehalten, wird jedoch im Folgenden auch bei diesen Korpora die Originalschreibung als Transkription bezeichnet.

Die nachfolgenden Abschnitte erläutern für die einzelnen Subkorpora, wie die Transkription erzeugt und/oder für die Verwendung in diesem Korpus aufbereitet wurde. Dabei wird auch die Tokenisierung der Texte angesprochen, d. h. wie insbesondere in den digitalen Korpora die Wortgrenzen ermittelt wurden und auch wie in den von Kindern verfassten Korpora die Tokengrenzen, sprich falsche

Getrennt- und Zusammenschreibungen, korrigiert und mit spezifischen Sonderzeichen markiert wurden. Die Verwendung dieser und weiterer in Tabelle 1 abgebildeter Sonderzeichen in der Transkription folgt den Guidelines von Laarmann-Quante et al. (2017).

SONDERZEICHEN	BEDEUTUNG	AUSNAHME
_ (Unterstrich)	Ein Wort war fälschlich getrennt geschrieben.	Klexikon
| (vertikaler Strich)	Zwei Wörter waren fälschlich zusammengeschrieben.	Klexikon
*	Nicht lesbares Zeichen	H1, Osnabrücker Bildergeschichtenkorpus, Grundschulwiki, Klexikon
^	Zeilenumbruch	H1, Grundschulwiki, Klexikon
\h	Ende einer Überschrift	H1, Klexikon

Tabelle 1: Meta-Sonderzeichen in der Transkription nach Laarmann-Quante et al. (2017)

4.1.1 Osnabrücker Bildergeschichtenkorpus

Das Osnabrücker Bildergeschichtenkorpus liegt, wie bereits in Kapitel 2.1 beschrieben, in Form von 36 XML-Dateien vor, die jeweils die Texte einer Schulklasse enthalten. Die Transkription, d. h. die originale Schreibung, ist in den XML-Dateien i. d. R. als Text-Wert der Tochterelemente aller <text>-Knoten gespeichert. In dem folgenden Ausschnitt entspricht die Transkription den in schwarz dargestellten Wörtern:

```
dann <f ortho="musste">muste</f> es
<versuch schreibung="woll">wohl</versuch>
oder <versuch schreibung="uhbel">übel</versuch> auf
<f ortho="harten">haten</f> &ze; <f typ="gks">boden</f> liegen
```

Abbildung 9: Ausschnitt aus einer XML-Datei des Osnabrücker Bildergeschichtenkorpus

Insbesondere die undurchsichtige Verwendung von <versuch>-Tags erschwert jedoch die Extraktion der Original-Schreibung aus den XML-Dateien, da wie in den folgenden Beispielen nicht immer klar ist, welche Zeichen tatsächlich vom Kind geschrieben wurden und worum es sich bei den anderen Schreibweisen handelt.

```
als der <f typ="gks">mann</f> &ze;
<versuch schreibung="angekomen"><f ortho="angekommen">an gekomen</f></versuch>
ist <komma vorhanden="nein"/> <f ortho="sieht">siet</f> er
```

Abbildung 10: <versuch>-Tag in einer XML-Datei des Osnabrücker Bildergeschichtenkorpus

```
<sanf>Er</sanf> <f ortho="nimmt">nimt</f> den
<versuch schreibung="Fuss"><f ortho="Fußabstreifer">Fusabstreifer</f></versuch> &ze;
mit <punkt/>
```

Abbildung 11: <versuch>-Tag in einer XML-Datei des Osnabrücker Bildergeschichtenkorpus

Interpunktionszeichen, die in den XML-Dateien nur in Form von Tags oder Attributen enthalten sind wie z. B. <punkt vorhanden="nein" noetig="ja"/> oder <punkt zeichen="ausruf" noetig="nein"/>, wurden automatisch rekonstruiert und – wenn im Originaltext vorhanden – als tatsächliche Interpunktionszeichen dargestellt. Dasselbe gilt für Anführungszeichen, die ebenfalls in Form von XML-Tags bzw. Attributen in den XML-Dateien enthalten sind (z. B. <rede markiert="ja">).

Fehlende Interpunktionszeichen und Wörter wurden als leere Strings übernommen. Eine Ausnahme sind fehlende Anführungszeichen zur Markierung direkter Rede. Damit später noch erkennbar ist, über welche Token sich die direkte Rede erstreckt, wurden zwei vorläufige Platzhalter-Marker für Anfang bzw. Ende der direkten Rede übernommen, die später im Zielformat als *direct_speech*="begin" bzw. *direct_speech*="end" wieder aufgenommen wurden (s. Abschnitt 4.7).

Das Zeilenumbruchszeichen *&ze;* wurde durch das von Laarmann-Quante et al. (2017) verwendete Meta-Zeichen ^ ersetzt. Das mit dem XML-Tag <titel> markierte Ende der Überschrift wurde mit dem dort gebrauchten \h dargestellt. Der Satzanfangs-Tag <sanf> wurde wie die direkte Rede mit einem vorläufigen Marker gespeichert. Die Information wurde hinterher jedoch als unzuverlässig und deshalb irrelevant erachtet und nicht in die LearnerXML-Dateien übernommen.

Außerdem wurden Transkription sowie orthographische und grammatische Korrektur automatisch so aligniert, dass die Darstellung den Regeln aus Laarmann-Quante et al. (2017) entspricht, d. h. ein Unterstrich _ im Originalwort bedeutet, dass es fälschlich getrennt geschrieben war, ein senkrechter Strich | im Originalwort bedeutet, dass es fälschlich zusammengeschrieben war, usw.

Der Anfang der Transkription des ersten Textes aus dem Korpus sieht wie folgt aus, wobei #b das temporäre, später wieder entfernte Symbol für einen manuell annotierten Satzanfang ist:

#b	Haus
Ein	
armer	er
Hund	sieht
\h	ein
#b	^
Ein	Hund
alter	
man	er
ckugt	gehd
aus	zur
dem	tür

4.1.2 H1 Children's Writing

Beim H1 Children's Writing Korpus ist die Transkription zu jedem Text in einer eigenen Datei enthalten. Darüber hinaus kann die Datei die in Abbildung 2 im Abschnitt 2.2 dargestellten Annotationen enthalten. Eine Transkriptionsdatei kann demnach z. B. wie folgt aussehen:

> Theo kletert auf den Klodeckel . Auf einmal hat er denn Rasirschaum und denn Rasirer in der hand . Er Rasirt sich aus_fersen eine Glatze . Alz Mama in das Badezimmer [§ kommt] muss sie lachen . Sie sagt hir ist ein Fisör ein§gekert .

Um die originale Schreibung der Kinder zu rekonstruieren, wurden die Annotationen automatisch auf die tatsächliche Schreibung zurückgeführt. Beispielsweise wurde [§ *abc*] gelöscht, [*abc* §] durch *abc* ersetzt, *abc*{3} durch *abc abc abc* etc. Dadurch kann die Transkription ggf. leere Zeilen enthalten, wenn das Kind Interpunktionszeichen oder Wörter ausgelassen hat. Die Annotationen von falscher Getrennt- oder Zusammenschreibung wurden auf die Darstellung nach Laarmann-Quante et al. (2017) abgebildet, d. h. ein Unterstrich _ im Originalwort bedeutet, dass es fälschlich getrennt geschrieben war, ein senkrechter Strich | im Originalwort bedeutet, dass es fälschlich zusammengeschrieben war, etc.

Die Markierung von Fremdwörtern und Eigennamen wurde ebenfalls aus den Dateien entnommen. Die Eigennamenmarkierung wurde später verwendet, um sicherzustellen, dass diese Wörter als NE getaggt sind. Die Annotation ist allerdings nur für 161 Namen vorhanden, während das Korpus letztlich fast 1.800 als NE getaggte Wörter enthält. Die Markierung der Fremdwörter betrifft ebenfalls lediglich 84 Wörter und wurde als unzuverlässig und unvollständig erachtet und deshalb nicht in die finalen LearnerXML-Dateien übertragen.

Zeilenenden sind im Korpus nicht annotiert und eine nachträgliche manuelle Ergänzung basierend auf den Original-Scans der Kindertexte ist vom Aufwand her nicht verhältnismäßig, dasselbe gilt für Überschriften.

Teilweise wurden Fehler in der Transkription wie bspw. fehlerhafte Tokenisierung manuell korrigiert, um eine korrekte Alignierung mit der Zielhypothese zu ermöglichen und die Qualität der Daten zu verbessern. Die Transkription zu obigem Text sieht wie folgt aus:

Theo	und	eine
kletert	denn	Glatze
auf	Rasirer	.
den	in	Alz
Klodeckel	der	Mama
.	hand	in
Auf	.	das
einmal	Er	Badezimmer
hat	Rasirt	
er	sich	muss
denn	aus|	sie
Rasirschaum	fersen	lachen

.	hir	Fisör
Sie	ist	ein_gekert
sagt	ein	.

4.1.3 Litkey

Das Litkey-Korpus liegt im csv-Format vor, wobei die Transkription der Kindertexte in der linken Spalte der csv-Dateien mit je einem Wort pro Zeile enthalten ist. Die Transkriptionen wurden im Litkey-Projekt manuell nach den Guidelines von Laarmann-Quante et al. (2017) erstellt und bilden die Verschriftungen der Kinder so exakt wie möglich ab. Die in Tabelle 1 in Abschnitt 4.1 dargestellten Sonderzeichen dienen dabei zur Korrektur der Tokenisierung der Kinder (| und _), zur Markierung von Überschriften (\h) und Zeilenenden (^) sowie zur Darstellung nicht lesbarer oder nicht existierender Zeichen (*). Das folgende Beispiel zeigt die Transkription zu einem Ausschnitt aus einem Kindertext.

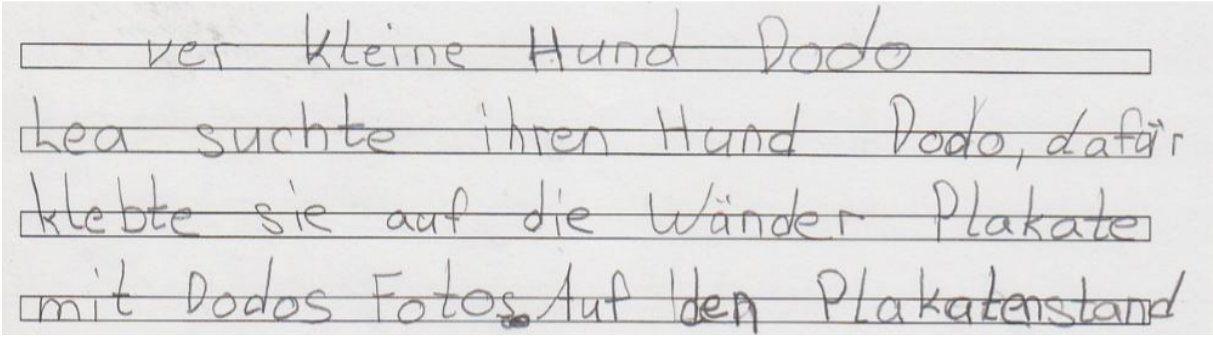

Abbildung 12: Ausschnitt aus dem Original-Scan eines Textes aus dem Litkey-Korpus

ver	sie
kleine	auf
Hund	die
Dodo	Wänder
\h	Plakate
Lea	^
suchte	mit
ihren	Dodos
Hund	Fotos
Dodo	.
,	Auf
dafür	den
^	Plakaten|
klebte	stand

4.1.4 Grundschulwiki

Mithilfe der Spezialseite des Grundschulwikis "Seiten exportieren"[3] war es möglich, die Texte des Grundschulwikis zu entnehmen und in Form einer XML-Datei herunterzuladen. Anschließend wurde die XML-Datei eingelesen, das jeweils einem Text zugehörige Element ausgelesen und die Meta-Informationen sowie der Knoten <text> separat abgespeichert.

Der ausgelesene Text in Form eines Strings wurde folgend auf angegebene Kategorien, Referenzen sowie Bildmaterial überprüft. Enthielt der String betreffende Informationen, wurden diese aus dem Text entfernt und der Metaverarbeitung übergeben, die in Unterkapitel 4.10 thematisiert wird. Anschließend wurden die verwendeten Anführungszeichen einheitlich auf das Zeichen " abgebildet und HTML-Metazeichen wie z. B. "{" und "}" bspw. in {{Mehr}}, "<" und ">" in <gallery> sowie "=" und "#" in BACKGROUND = #AADDA entfernt.

Mithilfe des NTLK-Moduls *nltk.tokenize* und dessen Funktion *word_tokenize* wurde der String schließlich tokenisiert und jeder Text in eine Datei mit je einem Token pro Zeile ausgegeben. Da das Modul *word_tokenize* bei der Tokenisierung Probleme mit Satzgrenzen und Internetadressen zeigte, wurden diese zum Teil manuell korrigiert. Probleme bereiteten bspw. Texte wie "SpongeBob Schwammkopf", der folgend dargestellt wird. Hier wurden z. B. die Internetadresse "Spongepedia.org" und die Geburtsdaten "(* 14.Juli 1986)" nicht richtig getrennt. Ebenso Probleme verursachte das Zeichen "*", welches hier zum Teil abgetrennt wurde, zum Teil jedoch an dem Token erhalten blieb.

[3] Zu erreichen unter https://grundschulwiki.zum.de/wiki/Spezial:Exportieren

```
  <page>
    <title>SpongeBob Schwammkopf</title>
    <ns>0</ns>
    <id>3823</id>
    <revision>
      <id>15305</id>
      <parentid>15304</parentid>
      <timestamp>2012-01-22T14:07:16Z</timestamp>
      <contributor>
        <username>Leupold</username>
        <id>3</id>
      </contributor>
      <minor/>
      <comment>babel rücksprung</comment>
      <text xml:space="preserve" bytes="943">{{Babel-1|Schüler_Verfasser}}
'''SpongeBob Schwammkopf''' ist eine Zeichentrickserie. Sie wurde von [[Stephen
Hillenburg]] erschaffen.
==Charaktere==
* Robert &quot;SpongeBob&quot; Schwammkopf (* 14.Juli 1986) ist der Hauptcharakter
der Serie. Er ist ein Meeresschwamm.
* Patrick Star (*26. Februar 1986) ist der Beste Freund von SpongeBob. . Er ist ein
Seestern. Häufig wird er als total Dumm dargestellt.
* Thaddäus Q. Tentakel (* 23. Oktober 1975) ist ein Tintenfisch. Er hasst SpongeBob.
* Eugene Herbert ( Mr.) Krabs (* 30. November 1942) sit eine sehr geizige Krabbe. Sein
Erzfeind Plankton versucht immer, ihm die Geheimformel zu stehlen.
* Sandra &quot;Sandy&quot; Cheeks (*4. April 1983) ist eine Freundin von SpongeBob.
Sie ist ein Grauhörnchen.
* Gary ist die Hausschnecke von SpongeBob.
* Sheldon J. Plankton (* 30 November 1942) ist der Erzfeind von Mr. Krabs.
Er ist ein sehr kleiner Plankton.

==Quelle(n)==
Spongepedia.org



{{Rücksprung|S}}</text>
      <sha1/>
      <model>wikitext</model>
      <format>text/x-wiki</format>
    </revision>
  </page>
```

Abbildung 13: Text "SpongeBob Schwammkopf aus dem Grundschulwiki im XML-Format

Bei der weiteren Verwendung dieses Programms entfällt diese manuelle Korrektur, da dazu eine automatische Variante verwendet wird, die in Kapitel 5 kurz thematisiert wird. Auf die Verwendung des Moduls *nltk.tokenize* wird dabei verzichtet.

Im Anschluss an die Tokenisierung wurden die Dateien mithilfe des Programms BoNKid normalisiert (s. Abschnitt 4.2.4). Das Tool nahm dabei Modifikationen an der Transkription vor. So wurden vereinzelt \h, der senkrechte Strich | und der Unterstrich _ eingefügt, um fälschlicherweise getrennte oder zusammengeschriebene Token zu korrigieren und Überschriften zu kennzeichnen. Die entsprechende Bearbeitung geschah semi-automatisch und erforderte anschließend manuelle Korrekturen.

Das folgende Beispiel zeigt einen Ausschnitt der finalen Transkription des Textes "SpongeBob Schwammkopf":

SpongeBob	.
Schwammkopf	Sie
ist	wurde
eine	von
Zeichentrickserie	Stephen

Hillenburg	Er
erschaffen	ist
.	ein
Charaktere	Meeresschwamm
*	.
Robert	*
"	Patrick
SpongeBob	Star
"	(
Schwammkopf	*
(	26.
*	Februar
14.	1986
Juli	)
1986	ist
)	der
ist	Beste
der	Freund
Hauptcharakter	von
der	SpongeBob
Serie	.
.	

4.1.5 Klexikon

Wie bei der Entnahme des Grundschulwikis wurden mithilfe der Spezialseite des Klexikons "Seiten exportieren"[4] die Texte der Webseite entnommen und in Form einer XML-Datei heruntergeladen. Die anschließende Verarbeitung wurde analog zum Grundschulwiki durchgeführt. So wurden die entsprechenden Meta-Informationen herausgefiltert, abgespeichert und aus dem Text entfernt sowie alle Anführungszeichen auf " abgebildet und HTML-Metazeichen gelöscht.

Allein die Durchführung der Normalisierung und die damit zusammenhängende Modifikation der Tokengrenzen durch den senkrechten Strich | und den Unterstrich _ entfällt, da nicht von Fehlern im Text auszugehen ist.

Da die Überschriften nur teilweise durch Meta-Markierungen wie "==" gekennzeichnet sind, war eine automatische Erkennung dieser Information nicht möglich und es wurde entsprechend auf die Annotation mit \h verzichtet.

Das folgende Beispiel zeigt einen Ausschnitt der XML-Datei und den Anfang der Transkription des Textes "Rathaus":

Rathaus	Haus
nennt	,
man	in
das	dem

[4] Zu erreichen unter https://klexikon.zum.de/wiki/Spezial:Exportieren

die	Gemeinde
Politiker	"
und	Politik
Mitarbeiter	machen
einer	"
Stadt	und
oder	arbeiten
.	

```
  <page>
    <title>Rathaus</title>
    <ns>0</ns>
    <id>639</id>
    <revision>
      <id>49787</id>
      <parentid>36001</parentid>
      <timestamp>2017-03-06T20:10:25Z</timestamp>
      <contributor>
        <username>Ziko van Dijk</username>
        <id>7</id>
      </contributor>
      <text xml:space="preserve" bytes="2254">[[Datei:Berner Rathaus 2007-09.jpg|mini|Das
Rathaus der [[Stadt]] [[Bern]]]]
Rathaus nennt man das [[Haus]], in dem die [[Politiker]] und Mitarbeiter einer [[Stadt]]
oder [[Gemeinde]] " [[Politik]] machen“ und arbeiten. Hier trifft sich der Rat, das heißt
die Volksvertretung der Stadt. So ein [[Parlament]] entscheidet über die wichtigsten Dinge
in der Stadt. Außerdem arbeiten dort der [[Bürgermeister]] und die Verwaltung: der Chef der
Stadt und seine Mitarbeiter.

In manchen Städten sind die Politiker und Mitarbeiter in ein neues Gebäude umgezogen. Das
alte Gebäude heißt dann vielleicht immer noch " Rathaus“ oder " Altes Rathaus“. Manchmal
sagt man " Rathaus“ und meint damit nicht wirklich das Gebäude, sondern die Politiker der
Stadt oder die Stadtverwaltung.

== Was passiert in einem Rathaus? ==
In [[Europa]] gibt es schon seit dem [[Mittelalter]] solche Rathäuser, wie man sie heute
kennt. Allerdings hat man sie oft nicht nur für den Rat verwendet. Ein Ort war eine Stadt
unter anderem, wenn man dort einen Markt abhalten durfte. Daher fand der Markt oft im
Rathaus statt. Im Rathaus gab oder gibt es außerdem vielleicht ein Restaurant.

Ein Rathaus braucht einen großen Saal, in dem der Rat sich treffen kann. Bei einer großen
Stadt wie [[Hamburg]] sind das 121 Ratsmitglieder. Der Rat kommt aber nicht ständig
zusammen. Darum nutzen viele Rathäuser ihren Saal noch für etwas anderes. Zum Beispiel
spielt man in Konzerten [[Musik]], oder es finden Treffen von Wissenschaftlern oder
Künstlern statt.

&lt;gallery&gt;
Datei:Wilhelmshaven townhall front facing north Wilhelmshaven Germany 03.jpg|Das Rathaus
von Wilhelmshaven, in [[Niedersachsen]], ist gleichzeitig ein Wasserturm.
Datei:Rathaus and Marienplatz from Peterskirche - August 2006.zugeschnitten.jpg|Neues
Rathaus in [[München]]: Vor über hundert Jahren hat man es im Stil der [[Gotik]] gebaut.
Datei:City Hall, Toronto, Ontario.jpg|Dieses moderne Rathaus steht in der Stadt Toronto in
[[Kanada]].
Datei:Ratusz2noc.jpg|Das Rathaus von Breslau, in [[Polen]], stammt aus dem [[Mittelalter]].
Datei:Festsaal Rathaus Vienna 2007.jpg|Festsaal im Rathaus von [[Wien]].
&lt;/gallery&gt;
{{Mehr}}
[[Kategorie:Klexikon-Artikel]]
[[Kategorie:Politik und Gesellschaft]]</text>
      <sha1>8yvha21b2unzfwrhq7s7zppug24xtqe</sha1>
      <model>wikitext</model>
      <format>text/x-wiki</format>
    </revision>
```

Abbildung 14: Text "Rathaus" aus dem Klexikon im XML-Format

4.2 Orthographische Zielhypothese

Die Korrektur der Rechtschreibfehler in der Transkription wird als orthographische Zielhypothese oder auch als Normalisierung des Textes bezeichnet. In vielen Lerner-Korpora wird die orthographische Korrektur mit der grammatischen und/oder lexikalischen Korrektur der Texte vermischt und nur eine einzige korrigierte Form annotiert. Hier sollen die orthographische und grammatische Ebene jedoch getrennt bleiben, d. h. die orthographische Zielhypothese enthält (idealerweise) nur Korrekturen eindeutig orthographischer Fehler, wie in Laarmann-Quante et al. (2017) beschrieben. So sollte der Satz in (1) auf der orthographischen Ebene lediglich zu (2), nicht jedoch bspw. zu (3) korrigiert werden.

(1) Lea sein Hund Dodo ist weg sie **hengt** überall **steck briefe**.

(2) Lea sein Hund Dodo ist weg sie **hängt** überall **Steckbriefe**.

(3) **Leas** Hund Dodo ist weg**. Sie hängt** überall **Steckbriefe auf**.

Auch die orthographische Zielhypothese kann wie die Transkription einige Sonderzeichen enthalten, die gemäß Laarmann-Quante et al. (2017) in Tabelle 2 dargestellt sind:

SONDERZEICHEN	BEDEUTUNG	AUSNAHMEN
\h	Ende der Überschrift	Klexikon, H1
?wort	Unbekanntes Zielwort	Grundschulwiki, Klexikon
*	Nicht lesbares Zeichen in unbekanntem Zielwort	H1, Osnabrücker Bildergeschichtenkorpus, Grundschulwiki, Klexikon
~wort	Nicht existierendes Zielwort (durch fehlerhafte Flexion)	Grundschulwiki, Klexikon

Tabelle 2: Meta-Sonderzeichen in der orthographischen Zielhypothese nach Laarmann-Quante et al. (2017)

Die folgenden Unterkapitel beschreiben, wie die orthographischen Zielhypothesen für die verschiedenen Subkorpora erstellt bzw. aufbereitet wurden und welche problematischen Aspekte es dabei ggf. gab.

4.2.1 Osnabrücker Bildergeschichtenkorpus

Da das Osnabrücker Bildergeschichtenkorpus in einem sehr komplexen, z. T. unübersichtlichen XML-Format vorliegt, ist das Auslesen der orthographischen Zielwörter vergleichsweise kompliziert. Die Korrektur der orthographischen Fehler ist in <f>-Tags oder <versuch>-Tags enthalten. Zum Auslesen der orthographischen Zielhypothese aus den <f>-Tags wurden anhand der dtd-Datei und Einzelfallbetrachtungen folgende Regeln erarbeitet und verwendet:

if <f> hat das Attribut *typ* und der *typ ist* "o" und es gibt ein Attribut *ortho*

➔ Zielhypothese = Wert von *ortho*

else if <f> hat das Attribut *typ* und der *typ* ist "g" und es gibt ein Attribut *ortho*

➔ Zielhypothese = Text-Wert von <f>

else if <f> hat das Attribut *typ* und der *typ* ist "og" und es gibt die Attribute *ortho* und *fortho*

➔ Zielhypothese = Wert von *fortho* bzw. wenn dieser nicht in childLex steht ~ + Wert von *fortho*

else if <f> hat ein Attribut *ortho*

➔ Zielhypothese = Wert von *ortho*

else if "gks" kommt in den Werten der Attribute von <f> vor

➔ Zielhypothese = der Text-Wert von <f> in umgekehrter Groß/-Kleinschreibung

Ist das orthographische Zielwort in einem <versuch>-Element enthalten, wurde es wie folgt ausgewertet:

- Ist die Transkription im *schreibung*-Attribut des <versuch>-Elements enthalten, ist das orthographische Zielwort der Text-Wert des <versuch>-Elements.
- Ist das orthographische Zielwort im *schreibung*-Attribut enthalten und steht es in childLex, wird es übernommen.
- Ist das orthographische Zielwort im *schreibung*-Attribut enthalten und steht es nicht in childLex, wird es mit führender Tilde ~ übernommen.

Die orthographische Korrektur der Interpunktionszeichen wurde wie bereits bei der Transkription aus den Tags rekonstruiert. Unbekannte orthographische Zielwörter wie <f ortho="???">wug</f> wurden den Regeln aus Laarmann-Quante et al. (2017) folgend mit einem führenden Fragezeichen (z. B. "?wug") versehen. Genau wie die Transkription kann auch die orthographische Zielhypothese ein leerer String sein, wenn das Kind bspw. ein Satzzeichen oder Wort vergessen hat.

Die Transkription wurde, wie bereits in Abschnitt 4.1.1 beschrieben, mittels Meta-Sonderzeichen so mit der Normalisierung aligniert, dass jede orthographische Zielhypothese in einer eigenen Zeile steht. Das folgende Beispiel zeigt Transkription und orthographische Zielhypothese, wobei auch die Zielhypothese die später wieder entfernten Satzgrenzenmarkierungen #b enthält:

#b	#b
Ein	Ein
armer	armer
Hund	Hund
\h	\h
#b	#b
Ein	Ein
alter	alter
man	Mann
ckugt	guckt
aus	aus
dem	dem
Haus	Haus
er	er
sieht	sieht
ein	ein
^	
Hund	Hund
er	er
gehd	geht
zur	zur
tür	Tür

Eine überblicksartige Kontrolle der orthographischen Zielhypothesen zeigt zwei Probleme auf: Grundsätzlich wird bei der Annotation im Osnabrücker Bildergeschichtenkorpus, wie in den Guidelines von Laarmann-Quante et al. (2017) gefordert, strikt zwischen orthographischer und grammatischer Korrektur unterschieden. Es gibt dennoch einige problematische Fälle, die nicht immer konsequent annotiert wurden, z. B.:

ORIG	ORTH	GRAM		ORIG	ORTH	GRAM
das	**das**	dass	vs.	das	**dass**	dass
im	**im**	in	vs.	im	**in**	in
den	**den**	dem	vs.	den	**dem**	dem

Es handelt sich bei diesen uneinheitlichen Annotationen jedoch zahlenmäßig um Einzelfälle. Deutlich häufiger besteht ein Problem bei der Annotation mit Tilden zur Markierung nicht existierender Wortformen durch fehlerhafte Flexion nach Laarmann-Quante et al. (2017). Der Grund hierfür liegt darin, dass die entsprechenden Wörter im Osnabrücker Bildergeschichtenkorpus häufig uneinheitlich annotiert sind. Beispielsweise findet sich für die Form *<laufte> zweimal die Annotation in (4) und zweimal die Annotation in (5), die deutlich macht, dass die orthographische Zielhypothese eine nicht existierende Form ist.

(4) <f typ="g" ortho="lief">laufte</f>

(5) <f typ="og" ortho="lief" fortho="laufte">Laufte</f>

Nur letztere Fälle wurden nach den obigen Regeln mit einer Tilde annotiert. Zudem wurden Wörter aufgrund der Regeln mit einer Tilde versehen, die nach Laarmann-Quante et al. (2017) nicht zu der entsprechenden Kategorie zählen, darunter z. B. "dawischte ~dawischte erwischte" sowie "kham ~kahm kam". Auch wurden ähnliche Fälle im Korpus häufig unterschiedlich annotiert wie zum Beispiel "Hunden_Haus Hundehaus Hundehaus" vs. "Hund-Haus ~Hundhaus Hundehaus", was bei Anwendung der Regeln ebenfalls zu unterschiedlichen Ergebnissen führt.

Die Annotation mit Tilden ist demnach nicht durchweg zuverlässig. Da jedes Nomen, Verb oder Adjektiv auf verschiedene Weisen fehlerhaft flektierbar ist, ist die Anzahl betroffener Formen allerdings potenziell sehr groß, weshalb eine automatische Erkennung kaum zuverlässig möglich und der Aufwand einer manuellen Durchsicht des Korpus nicht verhältnismäßig ist.

4.2.2 H1 Children's Writing

Beim H1 Children's Writing Korpus ist die orthographische Zielhypothese teilweise in der Transkriptions- und teilweise in der Zielhypothesendatei enthalten. Sie wurde analog zur Transkription automatisch auf die tatsächliche Korrekturschreibung zurückgeführt, indem die Annotations-metazeichen aus Abbildung 2 in Abschnitt 2.2 ersetzt wurden. Ist in der Transkriptionsdatei bspw.

abc{G} und in der Zielhypothesendatei *cde*{G} annotiert, ist die orthographische Zielhypothese *abc*. Ebenso wurde *a=b* als *a-b* übernommen, usw.

Genau wie die Transkription kann auch die orthographische Zielhypothese leere Strings enthalten, wenn das Kind Interpunktionszeichen oder Wörter ausgelassen hat. Unbekannte Zielwörter wurden größtenteils manuell mit einem führenden Fragezeichen versehen, sind jedoch im Ausgangsformat i. d. R. nicht gesondert markiert, weshalb nicht garantiert werden kann, dass alle betroffenen Wörter gefunden wurden. Zudem wurden einige Rechtschreib- bzw. Tippfehler in der orthographischen Zielhypothese korrigiert, doch auch dies geschah nicht für alle Texte, weshalb noch zahlreiche dieser Fehler im Korpus zu vermuten sind.

Die Transkription wurde mittels Meta-Sonderzeichen so aligniert, dass jedes orthographische Zielwort in einer eigenen Zeile steht. Folgendes Beispiel zeigt die Transkription neben der orthographischen Zielhypothese:

Theo	Theo
kletert	klettert
auf	auf
den	den
Klodeckel	Klodeckel
.	.
Auf	Auf
einmal	einmal
hat	hat
er	er
denn	den
Rasirschaum	Rasierschaum
und	und
denn	den
Rasirer	Rasierer
in	in
der	der
hand	Hand
.	.
Er	Er
Rasirt	Rasiert
sich	sich
aus|	aus

fersen	versehen
eine	eine
Glatze	Glatze
.	.
Alz	Als
Mama	Mama
in	in
das	das
Badezimmer	Badezimmer
muss	muss
sie	sie
lachen	lachen
.	.
Sie	Sie
sagt	sagt
hir	hier
ist	ist
ein	ein
Fisör	Frisör
ein_gekert	eingekehrt
.	.

Bei der orthographischen Zielhypothese gibt es im H1 Children's Writing Korpus genau wie beim Osnabrücker Bildergeschichtenkorpus zwei grundlegende Probleme. Erstens ist die orthographische Ebene nicht, wie in Laarmann-Quante et al. (2017) gefordert, konsequent auf reine Orthographiefehler beschränkt. Zwar wird auch im H1 Children's Writing Korpus nach orthographischer und grammatischer Korrektur unterschieden, die Trennung ist jedoch noch deutlich unzuverlässiger als

beim Osnabrücker Bildergeschichtenkorpus. So wird bspw. <ihn> auf der orthographischen Ebene immer zu <ihm> korrigiert, wenn dies das grammatische Zielwort ist, obwohl es sich offenkundig um einen Grammatikfehler handelt. Selbiges gilt für die Korrektur von <ein> zu <einen> etc. Auch noch drastischere Korrekturen wie <Gegengesetzen> zu <entgegengesetzten> sind im Korpus auf der orthographischen Ebene annotiert.

Zweitens sind Formen, die nach Laarmann-Quante et al. (2017) mit einer Tilde markiert werden müssten, im Korpus entweder schon auf der orthographischen Ebene korrigiert oder nicht gesondert markiert und dadurch nicht automatisch erkennbar. Lediglich bei der manuellen Korrektur wurden einige vereinzelte Tilden eingefügt. Dies geschah jedoch aufgrund des enormen Aufwands nicht für das gesamte Korpus, wodurch sich im Korpus nun bspw. zu der Form *<schlafte> bzw. *<schlaften> die vier orthographischen Zielhypothesen <~schlaften>, <schlaften>, <schlafte> und <schlief> finden. Die Annotation mit Tilden kann für dieses Subkorpus somit als eigentlich nicht vorhanden betrachtet werden.

4.2.3 Litkey

Die orthographische Zielhypothese ist im Litkey-Korpus in der rechten Spalte der csv-Dateien mit je einem Zielwort pro Zeile enthalten. Die Normalisierungen wurden genau wie die Transkription im Litkey-Projekt manuell nach den Guidelines von Laarmann-Quante et al. (2017) erstellt und enthalten ausschließlich die strikte Korrektur rein orthographischer Fehler in den Kindertexten. Dabei werden die in Tabelle 2 in Abschnitt 4.2 dargestellten Sonderzeichen zur Markierung nicht existierender und unbekannter Zielwörter sowie darin enthaltener nicht lesbarer Zeichen verwendet. Auch die Überschriftenmarkierung ist in der Zielhypothese enthalten. Das folgende Beispiel zeigt einen Ausschnitt aus einer Transkription mit orthographischer Normalisierung.

ver	Der
kleine	kleine
Hund	Hund
Dodo	Dodo
\h	\h
Lea	Lea
suchte	suchte
ihren	ihren
Hund	Hund
Dodo	Dodo
,	,
dafür	dafür
^	
klebte	klebte
sie	sie
auf	auf
die	die
Wänder	~Wänder
Plakate	Plakate
^	
mit	mit
Dodos	Dodos
Fotos	Fotos
.	.
Auf	Auf
den	den
Plakaten|	Plakaten
stand	stand

4.2.4 Grundschulwiki

Die Transkription des von Kindern verfassten Grundschulwikis kann aufgrund vorhandener Fehler nicht als orthografische Zielhypothese genutzt werden. Da keine weitere Annotation vorhanden ist, bestand der Bedarf, die Texte zu normalisieren. Um den manuellen Aufwand möglichst gering zu halten, wurde hierfür ein System für die automatische Korrektur der Rechtschreibfehler von Kindern eingesetzt.

Da herkömmliche Korrektursysteme wie Hunspell[5] häufig Probleme mit den von Kindern produzierten Kompetenzfehlern haben, die stärkere Abweichungen zum Zielwort aufweisen als z. B. Tippfehler, wurde für dieses Projekt das Programm BoNKid, der Bochumer Normalisierer für Kinderschreibungen, ausgewählt. Das Tool ist eine Weiterentwicklung des in Laarmann-Quante (2015) dargestellten Algorithmus und generiert seine Korrekturvorschläge in der Annahme, dass Kinder ein eingeschränkteres Vokabular haben als Erwachsene, basierend auf dem Kinderbuchkorpus childLex (Schroeder, Würzner, Heister, Geyken & Kliegl, 2015). Durch ein eigens entwickeltes Sprach- und Wortartmodell sowie Regeln zur Korrektur der Groß- und Kleinschreibung, Getrennt- und Zusammenschreibung und von Real-Word Errors, d. h. Fehlern, die zufällig wieder gültige Wörter der Sprache ergeben, ist BoNKid zudem in der Lage, kontextabhängige Fehler zu erkennen und zu korrigieren. Eine Evaluation des Systems auf Texten aus dem Litkey-Korpus hat gezeigt, dass damit eine deutliche Verbesserung des Gesamtfehleranteils in den Texten von 17% auf nur 4% möglich ist. Da bei der Evaluation 83% aller Korrekturen mit dem ersten Vorschlag, 96% jedoch mit den ersten drei Vorschlägen korrigiert werden konnten, verfügt das System zusätzlich über eine GUI, welche den Textkontext anzeigt und den Nutzer befähigt, die Korrektur manuell zu beeinflussen – sowohl durch die Auswahl des korrekten automatischen Korrekturvorschlags, als auch durch die Eingabe einer eigenen Korrektur (s. Kapitel 5).

Die erstmalige Nutzung der GUI bei der Annotation aller Texte des Grundschulwikis zeigte, dass die Ergebnisse in vielen Fällen korrekt waren, bedingt durch das zugrunde liegende childLex-Wörterbuch, welches nicht optimal für diese Textdomäne geeignet ist, konnten jedoch nicht alle Token automatisch normalisiert werden (z. B. "Feuerpfote"). An dieser Stelle wurden die Ergebnisse manuell bearbeitet.

Angelehnt an die Anforderung der Guidelines von Laarmann-Quante et al. (2017) besteht der Output aus einer zweispaltigen Textdatei, welche die Transkription und die orthographische Zielhypothese enthält. Beispielhaft wird dieses Format anhand des folgenden Ausschnittes des Textes "Feuerpfote" gezeigt:

Blaustern	Blaustern	ConnerClans	DonnerClans
die	die	bot	bot
derzeitige	derzeitige	Sammy	Sammy
Anführerin	Anführerin	(	(
des	des	der	der

[5] hunspell.github.io

dann	dann	zu	zu
zu	zu	kommen	kommen
Feuerpfote	Feuerpfote	,	,
wurde	wurde	um	um
)	)	einen	einen
an	an	Wakdkatze	Waldkatze
in	in	zu	zu
den	den	werden	werden
Wald	Wald	.	.

4.2.5 Klexikon

Da das Klexikon von geschulten Autoren verfasst wurde, ist davon auszugehen, dass hier keine Fehler existieren. Somit wurde die Transkription als orthographische Zielhypothese übernommen.

4.3 Grammatische Zielhypothese

Als grammatische Zielhypothese wird die grammatische Korrektur der Transkription bezeichnet. Während die orthographische Zielhypothese (idealerweise) ausschließlich Fehler korrigiert, die eindeutig auf die Verletzung orthographischer Regeln zurückzuführen sind, korrigiert die grammatische Zielhypothese auch Fehler, die ausschließlich grammatische Regeln verletzen. Wie weit die grammatische Korrektur reicht, d. h. ob bspw. nur Kongruenzfehler oder auch fehlende Wörter o. Ä. korrigiert werden, hängt vom jeweiligen Korpus ab. So könnte der Satz in (6) z. B. zu (7) oder (8) korrigiert werden.

(6) Lea sein Hund Dodo ist weg sie **hengt** überall **steck briefe**.

(7) Lea **ihr** Hund Dodo ist weg**. Sie hängt** überall **Steckbriefe**.

(8) **Leas** Hund Dodo ist weg**. Sie hängt** überall **Steckbriefe auf**.

Im vorliegenden Korpus ist eine grammatische Zielhypothese nur für diejenigen Subkorpora enthalten, in denen sie bereits annotiert war: im Osnabrücker Bildergeschichtenkorpus, im H1 Children's Writing Korpus und – wenn auch auf andere Art und Weise – im Klexikon. Die folgenden Abschnitte beschreiben, wie die Annotation jeweils erzeugt bzw. für dieses Korpus aufbereitet wurde. Die grammatische Zielhypothese kann folgende Meta-Sonderzeichen enthalten:

SONDERZEICHEN	BEDEUTUNG	AUSNAHMEN
\h	Ende der Überschrift	H1, Litkey, Grundschulwiki, Klexikon
?wort	Unbekanntes Zielwort	Litkey, Grundschulwiki, Klexikon
|	Zielwort musste für die Alignierung mit der orthographischen Zielhypothese getrennt werden	H1, Litkey, Grundschulwiki, Klexikon

Tabelle 3: Meta-Sonderzeichen in der grammatischen Zielhypothese

Für die restlichen zwei Subkorpora, das Litkey-Korpus und das Grundschulwiki, in denen eine grammatische Normalisierung nicht von vornherein vorhanden war, konnte diese nicht einfach analog erstellt werden, da eine solche Annotation nur mit hohem manuellen Aufwand möglich ist und zunächst eindeutige Annotationsregeln nötig wären. Derartige Regeln müssten tendenziell noch deutlich komplexer sein als diejenigen zur orthographischen Normalisierung und da auch zu keinem der anderen Korpora entsprechende Guidelines verfügbar sind, enthalten die zwei Subkorpora keine grammatische Zielhypothese.

4.3.1 Osnabrücker Bildergeschichtenkorpus

Wie schon das Auslesen der orthographischen Zielhypothese ist auch das Extrahieren der grammatischen Zielwörter aufgrund des unübersichtlichen XML-Formats komplizierter. Die grammatische Korrektur ist in den XML-Dateien in <f>-Tags oder <versuch>-Tags enthalten. Wie für die orthographische Zielhypothese wurden anhand der dtd-Datei und Einzelfallbetrachtungen folgende Regeln zum Auslesen der Informationen erarbeitet und verwendet:

if "gks" kommt in den Werten der Attribute von <f> vor

➔ Zielhypothese = der Text-Wert von <f> in umgekehrter Groß-/Kleinschreibung

else

➔ Zielhypothese = Wert von *ortho*

Ist das grammatische Zielwort in einem <versuch>-Element enthalten, wird es wie folgt ausgewertet:

- Ist die Transkription im *schreibung*-Attribut des <versuch>-Elements enthalten, ist das grammatische Zielwort der Text-Wert des <versuch>-Elements.
- Ist das orthographische Zielwort im *schreibung*-Attribut enthalten, wird das grammatische Zielwort mittels der obigen Regeln für Fehler in <f>-Elementen bestimmt.

Die grammatische Korrektur der Interpunktionszeichen wurde wie bereits bei der Transkription und der orthographischen Normalisierung rekonstruiert. Unbekannte grammatische Zielwörter wie in <f ortho="???">wug</f> wurden auch in der grammatischen Zielhypothese mit einem führenden Fragezeichen ("?wug") versehen. Genau wie die Transkription und die orthographische Normalisierung kann auch die grammatische Zielhypothese ein leerer String sein, wenn das Kind ein überflüssiges Satzzeichen oder Wort eingefügt hat.

Da sich die Alignierung der Wörter an der orthographischen Zielhypothese orientiert, kann die grammatische Zielhypothese Leerzeichen enthalten. Dies ist im Osnabrücker Bildergeschichtenkorpus insgesamt sechsmal der Fall bei:

ORIG	ORTH	GRAM
im	im	aus dem
in	in	aus dem
ogott	ogott	oh Gott (2 x)
ins	ins	in den
ausm	ausm	aus dem

Außerdem kann die grammatische Zielhypothese das Sonderzeichen | enthalten, wenn sie zur Alignierung getrennt werden musste. Dies tritt im Osnabrücker Bildergeschichtenkorpus allerdings nur ein einziges Mal auf:

ORIG	ORTH	GRAM
Schwarzen	schwarzen	schwarz-|
weisen	weißen	weißen
Pademantel	Bademantel	Bademantel

Das folgende Beispiel zeigt Transkription, orthographische und grammatische Zielhypothese, wobei alle drei Spalten die später wieder entfernten Satzgrenzenmarkierungen #b enthalten:

#b	#b	#b			.
Ein	Ein	Ein	er	er	er
armer	armer	armer	sieht	sieht	sieht
Hund	Hund	Hund	ein	ein	einen
\h	\h	\h	^		
#b	#b	#b	Hund	Hund	Hund
Ein	Ein	Ein			.
alter	alter	alter	er	er	er
man	Mann	Mann	gehd	geht	geht
ckugt	guckt	guckt	zur	zur	zur
aus	aus	aus	tür	Tür	Tür
dem	dem	dem			.
Haus	Haus	Haus			

Da zum Osnabrücker Bildergeschichtenkorpus keine Guidelines für die Erstellung der Zielhypothesen existieren, kann deren Qualität nur schlecht beurteilt werden und es ist basierend auf stichprobenartigen Beobachtungen von gewissen Inkonsequenzen bei der Annotation auszugehen.

4.3.2 H1 Children's Writing

Die grammatische Korrektur ist in den Zielhypothesendateien des H1 Children's Writing Korpus enthalten. Sie wurde genau wie die orthographische Zielhypothese automatisch auf die tatsächliche Korrekturschreibung zurückgeführt, indem Annotationsmetazeichen aus Abbildung 2 in Abschnitt 2.2 ersetzt wurden. Ist in der Transkriptionsdatei bspw. *abc*{G} und in der Zielhypothesendatei *cde*{G} annotiert, ist die grammatische Zielhypothese *cde*.

Ebenso wie die Transkription und die orthographische Zielhypothese kann auch die grammatische Zielhypothese leere Strings enthalten, wenn das Kind überflüssige Interpunktionszeichen oder Wörter eingefügt hat. Unbekannte grammatische Zielwörter wurden genau wie die orthographischen Zielwörter größtenteils manuell mit einem führenden Fragezeichen versehen, sind jedoch im Ausgangsformat i. d. R. nicht gesondert markiert, weshalb nicht garantiert werden kann, dass alle betroffenen Wörter gefunden wurden. Zudem wurden einige Rechtschreib- bzw. Tippfehler in der grammatischen Zielhypothese korrigiert, doch auch dies geschah nicht für alle Texte, weshalb noch zahlreiche dieser Fehler im Korpus zu vermuten sind.

Da sich die Alignierung an der orthographischen Transkription orientiert, wäre es prinzipiell möglich, dass in der grammatischen Zielhypothese Leerzeichen oder Trennzeichen vorkommen. Dies ist jedoch im H1 Children's Writing Korpus nicht der Fall. Möglicherweise ist dies auf die ohnehin hohe Übereinstimmung zwischen den beiden Korrekturebenen zurückzuführen. Das nachfolgende Beispiel zeigt die Transkription neben der orthographischen und grammatischen Zielhypothese:

Theo	Theo	Theo
kletert	klettert	klettert
auf	auf	auf
den	den	den
Klodeckel	Klodeckel	Klodeckel
.	.	.
Auf	Auf	Auf
einmal	einmal	einmal
hat	hat	hat
er	er	er
denn	den	den
Rasirschaum	Rasierschaum	Rasierschaum
und	und	und
denn	den	den
Rasirer	Rasierer	Rasierer
in	in	in
der	der	der
hand	Hand	Hand
.	.	.
Er	Er	Er
Rasirt	Rasiert	Rasiert
sich	sich	sich
aus|	aus	aus
fersen	versehen	versehen
eine	eine	eine
Glatze	Glatze	Glatze
.	.	.
Alz	Als	Als
Mama	Mama	Mama
in	in	in

das	das	das
Badezimmer	Badezimmer	Badezimmer
		kommt
muss	muss	muss
sie	sie	sie
lachen	lachen	lachen
.	.	.
Sie	Sie	Sie
sagt	sagt	sagt
hir	hier	hier
ist	ist	ist
ein	ein	ein
Fisör	Frisör	Frisör
ein_gekert	eingekehrt	eingekehrt
.	.	.

Da zum H1 Children's Writing Korpus keine Guidelines für die Erstellung der Zielhypothesen existieren, kann deren Qualität nur schwer beurteilt werden und es ist basierend auf stichprobenartigen Beobachtungen von nicht unerheblichen Inkonsequenzen bei der Annotation auszugehen.

4.3.3 Litkey

Für das Litkey-Korpus existiert aus den bereits in Abschnitt 4.3 beschriebenen Gründen keine grammatische Zielhypothese.

4.3.4 Grundschulwiki

Da BoNKid nicht über eine Funktion verfügt, eine grammatische Zielhypothese zu erstellen, wurde von dieser Möglichkeit Abstand genommen und das Grundschulwiki nicht grammatisch normalisiert.

4.3.5 Klexikon

Da davon auszugehen ist, dass sich keine grammatischen Fehler im Klexikon befinden, wurde auch hier die Transkription als grammatische Zielhypothese übernommen.

4.4 POS-Tags

Der folgende Abschnitt thematisiert das Tagging mit Part-Of-Speech-Tags nach dem STTS. Wie bei standardsprachlichen Korpora wie dem TüBa-D/Z-Korpus (Telljohann et al., 2004) üblich, wurden die Token mit dem jeweiligen POS-Tag versehen und so Informationen über ihre Wortart abgespeichert. Um eine Vergleichbarkeit mit anderen Korpora zu gewährleisten und ein Standard-Tagset des Deutschen zu nutzen, wird hier auf das Stuttgart-Tübingen-TagSet (kurz STTS; Schiller, Teufel, Stöckert, & Thielen, 1999) zurückgegriffen. Diese Wahl vereinfacht das Tagging, da die meisten Taggermodelle auf diesem trainiert sind und auch die zum Training eines Taggers erforderlichen getaggten Texte

dieses Tagset verwenden. Andere Tagsets wie z. B. die Universal POS-Tags[6] sind für das Deutsche bislang nicht sehr verbreitet und können auch nicht einfach auf das STTS abgebildet werden, da zwischen den Tags häufig 1:n-Beziehungen bestehen, d. h. ein Tag aus dem Universal POS-Tagset entspricht im STTS mehreren feingliedrigeren und für das Deutsche erprobten Tags (z.B. ADJ = ADJA/ADJD, ADP = APPR/ APPRART/APPO/APZR, VERB = VVFIN/VVINF/…, etc.). Eine umgekehrte Abbildung ist dagegen möglich, sodass das Korpus in seinem jetzigen Zustand auch ohne erneutes Taggen mit Universal POS-Tags angereichert werden könnte.

Zum Taggen der Korpora wurde der Stanford-POS-Tagger (Toutanova, Klein, Manning & Singer, 2003) verwendet[7]. Zwei der Subkorpora, das Osnabrücker Bildergeschichtenkorpus und das H1 Children's Writing Korpus waren bereits vor Projektbeginn getaggt (mehr hierzu in den Abschnitten 4.4.1 und 4.4.2). Das auf diesen beiden Korpora trainierte case-sensitive bidirektionale Modell wurde nun zum Tagging der Zielhypothesen des Litkey- und Grundschulwiki-Korpus verwendet. Die bidirektionale Architektur des Modells wurde gewählt, um eine höhere Tagging-Genauigkeit zu erreichen, da der Tagger hierbei nicht nur vorhergehende, sondern auch nachfolgende Wörter und ihre Tags berücksichtigen kann. Da das Modell auf den Zielhypothesen arbeitet, wurde zudem ein case-sensitives Modell gewählt, das von den darin enthaltenen Groß- und Kleinschreibungsinformationen profitieren kann. Es ist zu erwarten, dass das Modell besser mit den häufig grammatisch fehlerhaften Texten aus den Kinderkorpora umgehen kann als ein Standardmodell, da es auf ähnlich strukturierten Texten trainiert ist.

Da das Klexikon-Korpus von Erwachsenen verfasst ist, ist in diesen Texten eine standardsprachliche Satzstruktur zu erwarten, sodass für dieses Korpus ein Standard-Tagger-Modell eingesetzt werden kann. Es wurde ein bidirektionales case-sensitives Modell gewählt, das auf dem Tiger- (Version 2.1, 24.08.2006; Brants, et al., 2004) und dem TüBa-D/Z-Korpus (Release 10.0, 08/2015; Telljohann et al., 2004) trainiert wurde.

Die Texte aus allen Korpora wurden als kontinuierliche Liste von Zielwörtern an den Tagger übergeben, der über das Interface des NLTK (Natural Language Toolkit) aufgerufen wurde. Eine solche Liste kann bspw. wie folgt aussehen: ["Der", "kleine", "Hund", "Dodo", "\h", "Lea", "suchte", …].

Probleme bereitete vor allem der enorme Arbeitsspeicherverbrauch des Taggers. Wenn der Tagger mehr als 12 GB Arbeitsspeicher für einen Text benötigte, wurden problematische Sätze einzeln oder manuell getaggt. Anschließend wurden für alle Subkorpora weitere manuelle Korrekturen der Tags durchgeführt. Diese betreffen z. B. Satzzeichen, die nicht als solche getaggt wurden ("-" als XY oder "*" als NN), aber auch weitere falsch getaggte Token (u. a. APPR statt PTKVZ, kleingeschriebene Wörter als NN, usw.).

[6] universaldependencies.org/u/pos/
[7] http://nlp.stanford.edu/software/tagger.shtml (Download am 05.03.2017)

Wie die manuelle Korrektur einiger Tags gezeigt hat, ist die Qualität des automatischen Taggings bei nicht-standardsprachlichen Texten oft nicht ausreichend, um die Tags ohne Weiteres zu verwenden. Aus diesem Grund wurde eine stichprobenartige Evaluation durchgeführt. Um eine Vergleichbarkeit für alle Subkorpora zu gewährleisten, wurden automatisch ca. 1.000 orthographische bzw. grammatische Zielwörter pro Subkorpus mithilfe eines Programms entnommen und manuell annotiert. Die jeweiligen Abschnitte der Texte wurden dabei zufällig mithilfe des python-Moduls *random* und der Anzahl aller Sätze ausgewählt. Die Textdateien wurden ebenfalls zufällig und ohne vorhergehende Sortierung der Liste eingelesen. Das manuelle Tagging wurde von zwei Annotatoren durchgeführt. Jeder Annotator hat eine Hälfte der Sätze getaggt und anschließend den Part des zweiten Annotators korrigiert. Problemfälle wurden mit einem dritten Annotator besprochen. Im Anschluss wurden die Ergebnisse mithilfe eines Skriptes evaluiert, welches die manuellen Tags mit denen des Stanford-Taggers aus dem Korpus vergleicht und Unterschiede auswertet. Bei der Interpretation der Ergebnisse ist zu beachten, dass dies nicht unbedingt die original vom Tagger produzierten Tags, sondern bereits in unterschiedlichem Maße manuell korrigierte Tags sind. Die Ergebnisse sind dementsprechend nicht geeignet, um Auskunft über die allgemeine Tagging-Genauigkeit des Stanford-Taggers oder der verwendeten Modelle zu geben.

Es ist außerdem zu berücksichtigen, dass bei von Kindern geschriebenen Texten – insbesondere wenn keine vollständig grammatische Zielhypothese vorliegt – regelmäßig Fälle auftreten können, in denen es nicht möglich ist, einen korrekten Tag zu bestimmen. Hierzu zählen zum Beispiel die folgenden Sätze:

(9) Lea und Lars gehen **hinter** her.
(10) Ja **sag** Jan und auch Lars **sag** ich will auch dieser Spiel spielen.
(11) Lea klebt Poster **auf** die Poster steht
(12) Dann wartet sie auf einen **anruft**.
(13) da kam der kleine Jakob vorbei und **Hund** eine Idee

In vielen Fällen liegt das Problem wie bei (9) und (10) infolge der Normalisierung nach den Guidelines von Laarmann-Quante et al. (2017) in der Getrenntschreibung oder der Flexionsform. Hier wurde „im Zweifel für das Kind“ annotiert, sodass die Strukturen vergleichsweise standardsprachlichem, möglichst korrektem Deutsch entsprechen. Dies gilt auch für Wörter, die eigentlich mehrfach geschrieben werden müssten wie in (11). In Fällen, in denen eine größere Abweichung von der Norm vorhanden ist wie bspw. in (12), wurde wenn möglich nach dem Kontext getaggt, da so weniger Ambiguitäten zu erwarten sind, als wenn nur das isolierte Wort betrachtet wird. Ist dies aufgrund der ungrammatischen Struktur des Satzes nicht sinnvoll möglich wie im Fall von (13), wurde nach der Form des Wortes getaggt.

Die nachfolgenden Abschnitte zeigen für jedes Subkorpus auf, wie das Tagging im Einzelnen erfolgte, welche Wörter getaggt wurden und welche Korrekturen vorgenommen wurden. Ebenso werden einzeln für jedes Subkorpus die Ergebnisse der Evaluation dargestellt und Probleme beim Tagging aufgezeigt.

4.4.1 Osnabrücker Bildergeschichtenkorpus

Die Texte aus dem Osnabrücker Bildergeschichtenkorpus waren vor Beginn des Projekts bereits im Zuge einer Hausarbeit semi-automatisch getaggt worden. Dafür wurden der Tree-Tagger (Schmid, 1995)[8] und der Stanford-hgc-POS-Tagger (Toutanova et al., 2003)[9] verwendet. Der korrekte Tag wurde bei Uneinigkeit der Tagger manuell oder semi-automatisch ausgewählt, indem Bereiche identifiziert wurden, in denen einer der Tagger konsistent bessere Ergebnisse erzielt hat. So war bspw. der Tree-Tagger besser in der Lage, Demonstrativ- und Indefinitpronomen von Artikeln zu unterscheiden. Anschließend wurden weitere manuelle Korrekturen an den Tags durchgeführt, wobei v. a. unübliche Tag-Folgen wie ART-ART kontrolliert und ggf. korrigiert wurden.

Die im csv-Format vorliegenden Tags wurden nun im Rahmen dieses Projekts den korrekten Wörtern aus dem neu extrahierten Korpus zugeordnet und ggf. manuell korrigiert oder ergänzt, wenn es Abweichungen zwischen den Wörtern aus den beiden Quellen gab. Zudem wurden die neun Erlebnisberichte wie die anderen Korpora neu automatisch mit dem Stanford-Tagger und dem case-sensitiven bidirektionalen Kinder-Modell getaggt, da diese Texte in den anderen Daten nicht enthalten waren.

Bei den in diesem Projekt durchgeführten Korrekturen wurden die Tags weiterhin manuell so korrigiert, dass kein Wort als Satzzeichen und kein Satzzeichen als Wort getaggt ist. Außerdem wurden bei der Korrektur des Parsings vereinzelt weitere fehlerhafte Tags manuell korrigiert.

Für die Evaluation des Taggings wurden 1.013 Token aus dem Osnabrücker Bildergeschichten-korpus manuell getaggt. Dies entspricht 280 verschiedenen Zielwörtern bei Nichtberücksichtigung der Groß- und Kleinschreibung. Folgende Probleme (fett gedruckt) haben sich beim Tagging ergeben und wurden wie in Klammern angegeben gelöst:

- ... spielt mit **sein** (PPOSAT) seinem Schiff
- **er** (PPER) **Hause** (NN) **nach** (APPO) **ist** (VAFIN)
- Er **rennen** (VVFIN) weg **rennt** (VVFIN)
- da kam der kleine Jakob vorbei und **Hund** (NN) eine Idee
- Dann kam **eine** (PIS) Rocki lief davon.
- **Der** (PDS) sein Herrchen ist drinnen.

[8] http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/ (Download am 05.03.2017)
[9] Download ebenfalls am 05.03.2017

- Fußmatte **mit** (ADV).
- **es** (PPER) **spät** (ADV) **Abend** (NN)

Die Evaluation der Tags ergibt für das Osnabrücker Bildergeschichtenkorpus eine Accuracy von 96,15%, was 39 falschen auf insgesamt 1.013 Tags entspricht. Fehler traten bei den folgenden Tags auf (die Werte in Klammern geben den Accuracy-Wert für die einzelnen Tags an):

- NE (96.0%)
- APPRART (95.24%)
- APPR (94.44%)
- ADV (94.34%)
- VVINF (75.0%)
- PIS (50.0%)
- PWS (50.0%)
- PDS (25.0%)
- PTKVZ (11.54%)
- APPO (0%)

Hierbei traten APPO (1), PWS (2), PIS (2), PDS (4) und VVINF (8) bei der Evaluation weniger als zehn Mal auf. Die folgende Tabelle gibt einen Überblick über die verwechselten Tags im Osnabrücker Bildergeschichtenkorpus:

Gold-Tag	POS-Tag	Frequenz	Token
PTKVZ	ADV	23	nach heraus raus rein hinterher weg vorbei hinaus
APPR	APPRART	2	zu
VVINF	VVFIN	2	stehen gucken
PDS	ART	2	den Der
NE	NN	2	Hund Zwengelmann
ADV	PTKVZ	2	auf mit
PDS	PRELS	1	Der
PIS	ART	1	eine
APPRART	APPR	1	zur
APPO	APPR	1	nach
PWS	PIS	1	was
ADV	ADJD	1	spät

Tabelle 4: Tagging-Evaluationsergebnisse für das Osnabrücker Bildergeschichtenkorpus

Es ist aus der Tabelle ersichtlich, dass die verbliebene Haupt-Fehlerquelle mit 23 von 39 Fehlern die Verwechslung von Verbpartikeln mit Adverbien betrifft, wobei dies vor allem komplexere Partikel wie „heraus“, „hinterher“ oder „vorbei“ sind. Und auch insgesamt existieren Verwechslungen nur zwischen funktional ähnlichen Wortarten wie Nomen und Eigennamen oder Infinitiven und finiten Verben, die als relativ unproblematisch angesehen werden können. Für das Osnabrücker Bildergeschichtenkorpus kann somit insgesamt eine sehr hohe Qualität der Tags angenommen werden.

4.4.2 H1 Children’s Writing

Auch die Texte aus dem H1 Children’s Writing Korpus waren wie das Osnabrücker Bildergeschichtenkorpus bereits vor Beginn des Projekts im Rahmen einer Hausarbeit semi-automatisch getaggt worden (s. Abschnitt 4.4.1). Die im csv-Format vorliegenden Tags wurden für dieses Projekt den korrekten Wörtern aus dem neu extrahierten Korpus zugeordnet und ggf. manuell korrigiert oder ergänzt, wenn es Abweichungen zwischen den Wörtern aus den beiden Quellen gab.

Bei den in diesem Projekt durchgeführten Korrekturen wurden die Tags weiterhin so korrigiert, dass kein Wort als Satzzeichen und kein Satzzeichen als Wort getaggt ist. Außerdem wurden bei der Korrektur des Parsings vereinzelt fehlerhafte Tags manuell korrigiert. Auch wurde sichergestellt, dass die im Korpus als Eigennamen annotierten Wörter (s. Abschnitt 4.1.2) als NE getaggt sind.

Für die Evaluation wurden 1.006 Token aus dem Korpus manuell getaggt. Dies waren 376 verschiedene Zielwörter bei Nichtberücksichtigung der Groß- und Kleinschreibung. Folgende Probleme (fett gedruckt) haben sich beim Tagging ergeben und wurden wie in Klammern angegeben gelöst:

- … geben sie **?dira** (XY) hat gesagt …
- danach gehen sie **nachhause** (ADV)
- Es gibt so ein Spiel **der** (PDS) **was** (PRELS) als erstes den Schatz gefunden hat hat dann auch gewonnen und bekommt eine Millionen Goldbarren.
- Sie ist auf einem Besen und auch **welche** (PIAT) Bienen und da ist ein Mann.

Die Evaluation der Tags ergibt für das H1 Children’s Writing Korpus eine Accuracy von 97,42%, was 26 falschen auf insgesamt 1.006 Tags entspricht. Fehler kamen bei den folgenden Tags vor (die Werte in Klammern geben den Accuracy-Wert für die einzelnen Tags an):

- VAFIN (97.67%)
- VVFIN (97.06%)
- ADJA (94.12%)
- NE (93.75%)
- ADJD (87.5%)
- PIS (81.82%)

- VVINF (80.0%)
- PRELS (60.0%)
- PTKVZ (45.45%)
- PDAT (0%)
- ITJ (0%)
- PTKANT (0%)
- PDS (0%)

Hierbei traten PTKANT (1), ITJ (1), PDAT (1) und PDS (4) bei der Evaluation weniger als fünf Mal auf. Die folgende Tabelle gibt einen Überblick über die verwechselten Tags im H1 Children's Writing Korpus:

Gold-Tag	POS-Tag	Frequenz	Token
PTKVZ	ADV	6	weg los raus herum
PDS	ART	4	den der
VVINF	VVFIN	2	gehen
PIS	ART	2	einem eine
PTKANT	ITJ	1	okay
NE	NN	1	Mark
ITJ	NN	1	Tchüss
VVFIN	VVINF	1	tragen
VVFIN	NN	1	Spielen
VVFIN	VVPP	1	erwischt
ADJD	VVPP	1	aufgeräumt
VAFIN	VVFIN	1	hats
PRELS	PIS	1	was
PRELS	PDS	1	denen
PDAT	ADJA	1	selben
ADJA	PIDAT	1	anderen

Tabelle 5: Tagging-Evaluationsergebnisse für das H1 Children's Writing Korpus

Die Tabelle zeigt, dass Verwechslungen lediglich zwischen funktional ähnlichen Wortarten wie Verbpartikeln und Adverbien oder Infinitiven und finiten Verben vorkommen, die als relativ unproblematisch angesehen werden können. Für das H1 Children's Writing Korpus kann somit ebenfalls eine sehr hohe Qualität der Tags angenommen werden.

4.4.3 Litkey-Korpus

Die orthographischen Zielhypothesen aus dem Litkey-Korpus wurden für dieses Projekt mit dem Stanford-POS-Tagger getaggt. Hierfür wurde das case-sensitive Kinder-Modell verwendet, das auf dem Osnabrücker Bildergeschichtenkorpus und dem H1 Children's Writing Korpus trainiert ist. Die Texte wurden als kontinuierliche Liste von Zielwörtern [Der, kleine, Hund, Dodo, \h, Lea, suchte, …] an den Tagger übergeben.

Die Tags wurden anschließend teilweise manuell korrigiert. Es wurde sichergestellt, dass kein Wort als Satzzeichen und kein Satzzeichen als Wort getaggt ist. Außerdem wurden kleingeschriebene Zielwörter, die als NN getaggt waren, überprüft und der Tag ggf. korrigiert. Zudem wurden bei der Korrektur des Parsings vereinzelt weitere fehlerhafte Tags manuell korrigiert.

Für die Evaluation wurden 1.013 Token aus dem Korpus manuell getaggt. Dies entspricht 315 verschiedenen Zielwörtern bei Nichtberücksichtigung der Groß- und Kleinschreibung. Folgende Probleme (fett gedruckt) haben sich beim Tagging ergeben und wurden wie in Klammern angegeben gelöst:

- kuck mal ich hab mein Hund Dodo **mit** (ADV) genommen
- Anna und Lars rennen **hinter** (ADV) **her** (ADV)
- Plötzlich ~rufte Dodo **steht** (VVIMP) auf
- sie wartet auf einen **anruft** (NN)
- Dann hatte Lars in den Beutel **nach** (ADV) **kuckt** (VVPP)
- Lars freut sich auch **wieder** (ADV) **zu** (PTKZU) **sehen** (VVINF) und er gab Dodo seine Hand.
- Lars legt ein Baustein **ein** (PTKVZ) auf Holzburg

Die Evaluation der Tags ergibt für das Litkey-Korpus eine Accuracy von 95,85%, was 42 falschen auf insgesamt 1.013 Tags entspricht. Fehler traten bei den folgenden Tags auf (die Werte in Klammern geben den Accuracy-Wert für die einzelnen Tags an):

- ART (98.53%)
- NN (98.32%)
- KON (96.15%)
- VVFIN (94.35%)
- ADV (94.34%)
- VVPP (93.1%)
- KOUS (90.48%)
- PDS (80.0%)
- ADJD (80.0%)
- PWS (60.0%)

- PTKVZ (52.0%)
- VVIMP (50.0%)
- ADJA (50.0%)
- VAINF (0%)

Hiervon traten VAINF (1), ADJA (4), PWS (5), PDS (5) und VVIMP (6) bei der Evaluation weniger als zehn Mal auf. Die folgende Tabelle gibt einen Überblick über die verwechselten Tags im Litkey-Korpus:

Gold-Tag	POS-Tag	Frequenz	Token
PTKVZ	APPR	7	an auf
VVFIN	VVINF	4	tropfen setzen sehen
VVIMP	VVFIN	3	rufen steht kuck
PWS	PWAV	2	wo
PTKVZ	ADV	2	hinterher rein
PTKVZ	ADJD	2	fest
KON	ADV	2	aber
VVFIN	VVIMP	2	komm nimm
ART	PDS	1	die
PDS	ART	1	den
ADV	KON	1	Aber
ADV	APPR	1	hinter
ADV	PTKVZ	1	mit
PTKVZ	PAV	1	drauf
ADJD	VVPP	1	verschwunden
ADJD	ADV	1	übersetzt
VVFIN	VVPP	1	befestigt
KOUS	KOKOM	1	Als
KOUS	APPR	1	Bis
VAINF	VAFIN	1	haben
VVPP	VVFIN	1	kuckt
VVPP	VVINF	1	wehgetan
ADJA	VVFIN	1	letzten

ADJA	PIAT	1	anderem
NN	NE	1	Holzburg
NN	VVFIN	1	anruft

Tabelle 6: Tagging-Evaluation im Litkey-Korpus

Die Tabelle zeigt, dass der Tagger u. a. Probleme mit der Unterscheidung von Präpositionen und Verbpartikeln hat. Dies betrifft v. a. Fälle wie in (14) und (15), in denen keine Satzgrenzen vorhanden sind und die Verbpartikel unmittelbar vor dem nächsten Nomen stehen.

(14) … Lars sagt pass **auf** Lea stolpert über Dodo und …

(15) ... aber niemand ruft **an** Lea kommen die Tränen.

Auch Verwechslungen der Verbformen lassen sich häufig auf fehlerhafte Satzstrukturen und mangelnde Interpunktion zurückführen wie in den Beispielen (16) oder (17), die auch für einen menschlichen Annotator Probleme bereiten.

(16) Plötzlich ~rufte Dodo **steht** auf.

(17) Dann hatte Lars in den Beutel nach **kuckt** …

Insgesamt ist jedoch festzuhalten, dass die Qualität der Tags mit einer Accuracy von fast 96% angesichts der Probleme beim Tagging orthographischer Zielhypothesen sehr hoch ist.

4.4.4 Grundschulwiki

Das Tagging des Grundschulwikis erfolgte durch den Stanford-POS-Tagger, bei welchem das case-sensitive Kinder-Modell verwendet wurde. Als Input wurden dabei die normalisierten Texte in Form einer Liste genutzt. Der Tagger erzeugte daraus einen Output in Form einer csv-Datei, welche zeilenweise aus der normalisierten Wortform und dem jeweiligen zugehörigen Tag besteht. Das folgende Beispiel zeigt einen Ausschnitt des getaggten Textes "Lego Strategie ":

Man PIS
kann VMFIN
auch ADV
Cheats NE
verwenden VVINF
($(
wenn KOUS
du PPER
nicht PTKNEG
weiß VVFIN
was PWS
das PDS
ist VAFIN

guck VVIMP
mal ADV
bei APPR
Cheats NE
) $(
. $.
Es PPER
gibt VVFIN
6 CARD
verschiedene ADJA
Storys NN
. $.
Die ART

König	NN
,	$,
Magier	NN
,	$,
Piraten	NN
,	$,
Gouvernour	NN
,	$,
Astronauten	NN
und	KON
Alien	FM
Story	FM
.	$.
Man	PIS
kann	VMFIN
es	PPER
auch	ADV
in	APPR
Multiplayer	NN
spielen	VVINF
.	$.
Neue	ADJA
Sachen	NN
sich	PRF
kaufen	VVINF
und	KON
freischalten	VVINF
.	$.
Es	PPER
ist	VAFIN
sehr	ADV
sehr	ADV
cool	ADJD
.	$.
XD	XY

Das abgebildete Beispiel zeigt nicht nur das entsprechende Format, sondern deutet ebenso die Probleme des Taggings von Kindertexten an. Kinder verwenden in ihren Texten nicht nur Wörter wie z.B. "cool" oder "Cheats", sondern auch Emoticons "XD" oder doppelte Ausdrücke "sehr sehr".

Während dieses Beispiel nur einen Ausschnitt möglicher Probleme zeigt, verweist die Evaluation von 1.021 Token auf die falsche Annotation von insgesamt 84 Token, 937 Token wurden richtig annotiert. Dies entspricht einer Gesamtgenauigkeit von 91,77%.

Beim manuellen Tagging traten unter anderem folgende Probleme auf (das Problem ist fettgedruckt und die anschließende Annotation steht in Klammern dahinter):

- Im **Mai** (NN) verlor der BVB gegen den **FC** (NE) Bayern München im Champions League Finale durch das Tor von Frank Ribéry.
- Die meisten Menschen benutzen das Wort **cool** (ADJD) Heutzutage sollten alle Menschen wissen was **cool** (ADJD) heißt.
- DJ Antoine hat sich mit DJs **zusammen** (ADV) getan.
- In Frankreich bezahlt man mit dem **Euro** (NN).
- Griechenland hat das Kennzeichen **GR** (NE).

Fehler fanden sich bei den folgenden Tags (in Klammern befinden sich die Accuracy-Werte des jeweiligen Tags und dahinter die jeweiligen Frequenzen):

- ART (98.86%) 88
- NN (96.02%) 176
- ADJA (95.65%) 23
- APPR (94.12%) 68
- VVPP (94.12%) 17
- ADV (91.11%) 45
- $. (84.21%) 76
- VVINF (83.33%) 12
- NE (80.0%) 110
- ADJD (76.92%) 26
- PRELS (75.0%) 4
- KOKOM (75.0%) 4
- CARD (72.97%) 37
- PWS (66.67%) 3
- KOUS (66.67%) 6
- PDS (50.0%) 2
- VVIMP (50.0%) 4
- PIAT (33.33%) 6
- FM (0%) 2

Die folgende Tabelle zeigt die verwechselten Tags mit ihrer Frequenz. In der ersten Spalte befinden sich die Tags der Goldannotation, in der zweiten die Lösung des Taggers, in der dritten Spalte die Frequenzen sowie in der vierten Spalte die betroffenen Token.

Gold-Tag	POS-Tag	Frequenz	Token
NE	NN	17	Achkarren Dumbledore Lohner-Porsche Arsenal Ärzten Hochstaden Voldemort ISBN Urlaub Altenmünster Bozen-Brixen FSV FC
$.	$(	12	! ? :
CARD	ADJA	8	1. 21. 11. 24. 1940er 7. 13. 6.
NN	NE	7	Burg St. Wicht Cheats Graf Monde
PIAT	PIDAT	4	alle viele
ADV	ADJD	3	natürlich Früher lang
NE	ADJA	3	Kölner Bonner Poppelsdorfer
ADJD	ADV	3	Bestimmt südlich östlich

FM	NE	2	Champions League
ADJD	NN	2	Himmelblau Gold/Braun
APPR	KON	2	bis
ADV	PIAT	1	Viel
PDS	ART	1	Die
PRELS	ART	1	die
NE	XY	1	B
NE	VVINF	1	Wertingen
ADJA	NN	1	Bekannte
VVINF	VVFIN	1	wissen
VVINF	VVPP	1	getan
ADJD	VVPP	1	erhalten
VVIMP	VVFIN	1	weine
VVIMP	PRF	1	sing
KOKOM	KOUS	1	wie
APPR	PTKZU	1	zu
APPR	KOKOM	1	als
KOUS	KOKOM	1	als
KOUS	KOUI	1	um
VVPP	ADJD	1	umgekehrt
CARD	ADJD	1	3-87067-735-x

Tabelle 7: Tagging-Evaluation im Grundschulwiki

Wie in der Tabelle dargestellt, kommt es vor allem zu Problemen bei der Unterscheidung der Tags NE und NN. Verwechslungen wie "Urlaub" und "Ärzten" sind jedoch dem Kontext geschuldet, da diese hier auf "Die Ärzte" und "Farin Urlaub" referieren. Die Accuracy von knapp 92% ist angesichts der Probleme beim Tagging von Zielhypothesen sowie dem ungewöhnlichen Vokabular aufgrund des großen Themenspektrums als akzeptabel zu bewerten.

4.4.5 Klexikon

Da das Klexikon nicht von Kindern verfasst wurde, erfolgte das Tagging mit dem standardsprachlichen auf dem TüBa-D/Z- und dem Tiger-Korpus trainierten Modell des Stanford-POS-Taggers. Auch diesem Tagger wurden die Texte in Form einer Liste als Input übergeben und der Output in Form einer zweispaltigen csv-Datei erzeugt. Das folgende Beispiel stammt aus dem Text "Bochum".

Bochum	NE	der	ART
ist	VAFIN	20	CARD
eine	ART	größten	ADJA
Stadt	NN	Städte	NN
in	APPR	Deutschlands	NE
Nordrhein-Westfalen	NE	.	$.
.	$.	Sie	PPER
Sie	PPER	liegt	VVFIN
hat	VAFIN	im	APPRART
etwa	ADV	Ruhrgebiet	NE
360.000	CARD	zwischen	APPR
Einwohner	NN	Essen	NE
und	KON	und	KON
ist	VAFIN	Dortmund	NE
damit	PAV	.	$.
eine	PIS		

Bei der Durchführung der in Kapitel 4.4 beschriebenen Evaluation ergaben sich auch beim Tagging der von Erwachsenen geschriebenen Texte einige Probleme (das Problem ist fettgedruckt und die anschließende Annotation steht in Klammern dahinter):

- Die Mündung, **wo** (PWAV) das Flusswasser ins Meer fließt, liegt in Brasilien.
- Der Buchstabe **A** (NN) zum Beispiel für den Laut **A** (NN), wie am Anfang des Wortes "Affe".
- Das Wort AIDS spricht man wie **Äjts** (NN) aus.

Die Evaluation der Tags zeigt eine Accuracy von 93,84%. Dies umfasst bei insgesamt 1.006 Tags, 62 falsch getaggte Token und 944 korrekt klassifizierte Token. Bei den folgend dargestellten Tags traten Fehler auf (in Klammern befindet sich die Accuracy und nachstehend die Frequenz):

- ART (98.08%) 104
- APPR (95.83%) 72
- NN (95.77%) 189
- $. (94.94%) 79
- CARD (92.31%) 13
- ADJD (90.0%) 20
- VVPP (89.47%) 19
- ADV (82.86%) 70
- PTKVZ (80.0%) 10
- NE (77.5%) 40
- VVINF (76.47%) 17
- VAINF (75.0%) 4

- PWAV (57.14%) 7
- PDS (50.0%) 2
- PIAT (36.36%) 11
- VMINF (0%) 1

Die folgende Tabelle zeigt eine Übersicht der verwechselten Tags:

Gold-Tag	POS-Tag	Frequenz	Token
NE	NN	9	AIDS Pyrenäen Portugiesisch Afghanistan Handler Syrakus Elisabeth-Turm Krill Deutschlands
ADV	ADJD	7	Weltweit lang später Später unterschiedlich
PIAT	PIDAT	7	Viele allen beiden viele alle vielen
NN	NE	6	Äjts DDR Deutschland Mittelmeer
$.	$(	4	: ?
VVINF	VVFIN	4	kennen reisen wiedererkennen wegziehen
PWAV	KOUS	3	wie
VVPP	VVFIN	2	regiert weitererzählt
ADV	PIAT	2	Mehr mehr
ADV	PAV	2	dorthin stattdessen
PTKVZ	ADV	2	hinein weh
ADJD	VVPP	2	benannt verheiratet
NN	XY	2	A
CARD	ART	1	eine
ADV	NN	1	Jahrtausendelang
VAINF	VAFIN	1	werden
VMINF	VMFIN	1	wollen
ART	PIS	1	eine
ART	PRELS	1	die
PDS	ART	1	das
APPR	ART	1	der
APPR	KOKOM	1	als
APPR	ADV	1	über

Tabelle 8: Tagging-Evaluation im Klexion

Anders als bei den anderen Subkorpora ist die Qualität der Tags im Klexikon im Vergleich zu State-of-the-Art-Ergebnissen nicht gänzlich zufriedenstellend. Da es sich hier nicht um Texte von Kindern handelt, war ein besseres automatisches Ergebnis zu erwarten. Der Grund für das dennoch eher mittelmäßige Resultat könnte die vorliegende Textsorte sein. Während der Tagger auf Zeitungsartikeln trainiert wurde, handelt es sich hier um Lexikonartikel, die zudem für eine sehr spezielle Zielgruppe geschrieben sind. Die Tabelle zeigt allerdings auch, dass die häufigsten Verwechslungen zwischen ähnlichen Tags (NN und NE, ADV und ADJD, PIAT und PIDAT) bestehen und damit als vergleichsweise unproblematisch bewertet werden können.

4.5 Dependenz-Parses

Um die Kinderkorpora für dieses Projekt mit Dependenzannotationen anzureichern, wurde der Stanford-Neural-Dependency-Parser aus der Stanford-Core-NLP-Pipeline[10] ausgewählt. Er basiert auf einem neuronalen Netz und erzielt mit Accuracy-Raten von 90% oder mehr sehr gute Ergebnisse für englischsprachige Korpora (Chen & Manning, 2014). Da der Dependenzparser nicht über das NLTK nutzbar ist, wurde er über die Kommandozeile bzw. das *subprocess*-Modul von Python angesteuert.

Ebenso wie der Tagger benötigt der Parser zunächst ein trainiertes Modell, um damit neue Texte annotieren zu können. Im Stanford-Core-NLP-Paket ist ein auf Universal Dependencies[11] trainiertes deutsches Parsing-Modell *UD_German_official* enthalten. Analog zum Universal-POS-Tagset sollen die Universal Dependencies für die Annotation jeder beliebigen Sprache genutzt werden können und es existiert eine wachsende Anzahl von Korpora, die nach diesen Guidelines mit Dependenzen annotiert sind.

Darüber hinaus ist es möglich, eigene Parsing-Modelle für die Nutzung mit dem Stanford-Parser zu trainieren. Die Trainingsdaten müssen dafür im CoNLL-X-Format vorhanden sein, d. h. neben den eigentlichen Wörtern werden POS-Tags sowie Dependenzrelationen für jeden Satz benötigt. Dies kann zum Beispiel wie folgt aussehen:

1	Dodo	_	_	NE	_	2	subj	_	_
2	bellt	_	_	VVFIN	_	0	ROOT	_	_
3	den	_	_	ART	_	4	det	_	_
4	Eismann	_	_	NN	_	2	obj	_	_
5	an	_	_	PTKVZ	_	2	vpart	_	_
6	.	_	_	$.	_	2	punct	_	_

[10] Version 3.8.0 vom 09.06.2017, Download von https://stanfordnlp.github.io/CoreNLP/download.html
[11] http://universaldependencies.org/

Zusätzlich zu den Trainingsdaten wird ein TreebankLanguagePack benötigt, das weitere Informationen zum Trainingskorpus und der entsprechenden Sprache enthält. Fürs Deutsche kann dafür das im Stanford-Core-NLP-Paket enthaltene TueBaDZLanguagePack genutzt werden.

Im Rahmen dieses Projekts wurde zunächst ein Parsing-Modell auf dem TüBa-D/Z-Korpus (Release 10.0, 08/2015; Telljohann et al., 2004) trainiert, welches mit Dependenzen nach Foth (2006) annotiert ist. Zudem wurde ein weiteres Modell auf 1.882 manuell nach den Guidelines von Dipper (2017) annotierten Sätzen trainiert. Die Dependenzguidelines orientieren sich an den Universal Dependencies, haben jedoch den Anspruch weniger komplex und auch auf Nicht-Standarddaten gut anwendbar zu sein. Von den 1.882 Trainingssätzen entstammen 1.488 dem Litkey-Korpus und 394 dem Klexikon. Aus dem Litkey-Korpus wurden vor dem Training die Meta-Sonderzeichen aus den Zielhypothesen entfernt. Die Satzgrenzen waren im Rahmen einer Hausarbeit manuell bestimmt worden, ebenso wie die STTS-POS-Tags.

Mit den drei verschiedenen Parser-Modellen (Dipper, Tüba und Universal Dependencies) wurden in diesem Projekt alle fünf Subkorpora geparst. Dabei wurde die grammatischste Version jedes Textes geparst, d. h. die grammatischen Zielhypothesen, sofern diese vorhanden waren. Gab es für ein Subkorpus nur orthographische Zielhypothesen oder war das grammatische Zielwort ein leerer String, weil das Wort oder Satzzeichen im Text des Kindes überflüssig war, wurde das orthographische Zielwort geparst. War dieses ebenfalls ein leerer String, wurde die Transkription eines Wortes geparst.

Da der Stanford-Dependenz-Parser und die Stanford-Core-NLP-Pipeline keine bereits tokenisierten oder annotierten Input-Daten akzeptieren, wurden die Zielhypothesen mit Leerzeichen zusammengefügt und als zusammenhängender String an den Parser übergeben. Eine Ausnahme sind Überschriften, die gesondert geparst und anschließend wieder mit dem Rest des Textes zusammengesetzt wurden, um eine Satzgrenze am Ende der Überschrift zu garantieren. Das Meta-Zeichen \h für das Ende der Überschrift wurde nicht an den Parser übermittelt, sondern anschließend von einem Skript als Dependent der Wurzel der Überschrift mit der Relation für Nicht-Wörter aus dem entsprechenden Tagset (xy, -UNKNOWN- bzw. dep) annotiert.

Da der Parser keine Vorannotation annimmt, benötigt er zum Parsen einen eigenen Tagger für die Ermittlung der POS-Tags. Hierfür wurde, wenn möglich, derselbe Tagger wie für das Tagging in diesem Korpus verwendet, d. h. das standardsprachliche Modell für die Klexikon-Texte sowie das Kinder-Modell für die restlichen Subkorpora. Einzelne Sätze, die sich aufgrund zu hohen Arbeitsspeicherverbrauchs des Taggers nicht automatisch parsen ließen, wurden entweder mit dem standardsprachlichen Tagger-Modell getaggt und dann geparst oder, wenn dies nicht möglich war, manuell annotiert.

Das Ergebnis des Parsings mit dem Universal-Dependencies-Modell sieht für einen Satz aus dem Klexikon wie folgt aus:

nsubj(im-7, Alexander-1)
det(Große-3, der-2)
det(Alexander-1, Große-3)
cop(im-7, war-4)
det(im-7, ein-5)
amod(im-7, Feldherr-6)
root(ROOT-0, im-7)
amod(Griechenland-9, Alten-8)
det(im-7, Griechenland-9)
punct(im-7, .-10)

Da der Parser nicht die bereits vorhandene Tokenisierung nutzt, mussten die Parsing-Ergebnisse für alle Korpora anschließend korrigiert werden, um eine Re-Alignierung mit den Input-Texten zu ermöglichen. Zudem mussten einzelne Zeichen aus dem Output auf den Input abgebildet werden, wie z. B. "-LRB-" auf "(" oder "$" auf "€". Des Weiteren wurden in allen Subkorpora Sätze korrigiert, in denen laut Universal-Dependencies-Modell eine Root-Relation zwischen zwei oder mehr Token und nicht nur zwischen einem Token und der Wurzel besteht, z. B.:

root(ROOT-0, Lea-1)
appos(Lea-1, hat-2)
det(Eis-4, ein-3)
dobj(hat-2, Eis-4)
cc(Eis-4, aber-5)
root(aber-5, es-6)
root(es-6, hat-7)
det(Loch-9, ein-8)
dobj(hat-7, Loch-9)
punct(hat-7, .-10)

Auch Wörter, die als Satzzeichen geparst wurden, und umgekehrt wurden manuell korrigiert. Zudem wurde sichergestellt, dass kein Token als Dependent von einem Satzzeichen abhängt. Darüber hinaus wurde dafür gesorgt, dass Satzzeichen bei der Annotation mit dem Dipper-Modell immer Dependenten der Wurzel des Satzes sind, wie es auch in den Trainingsdaten des Modells, nicht jedoch in den geparsten Texten der Fall war.

Eine Evaluation wäre für das Dependenzparsing anders als für die Annotation mit POS-Tags nur schwierig und mit großem Aufwand möglich. Zunächst müssten hierfür die z. T. sehr umfangreichen Guidelines der verschiedenen Modelle studiert und ihre Anwendbarkeit auch auf ungrammatische oder „Bandwurm“-Sätze geprüft werden. Die Probleme, die bereits beim POS-Tagging aufgetreten sind wie bspw. Wörter, die mit unterschiedlichen Funktionen zu zwei Sätzen gehören müssten wie in den Beispielen (18) und (19), stellen auch auf der Parsing-Ebene wieder ein Problem dar, das zunächst gelöst werden müsste.

(18) Lea klebt Poster **auf** die Poster steht …

(19) Lars kommt **mit** Dodo bellt.

Weiterhin besteht das Problem, dass eine sinnvolle Evaluation nur mit korrekter Tokenisierung und Satzgrenzenerkennung möglich ist, die hier jedoch nicht gegeben ist. Somit sind aktuell nur folgende Einschätzungen zur Qualität des Dependenzparsings möglich:

- Da der Parser nicht auf die vorhandenen korrigierten POS-Tags zugreift, sondern die Texte vollständig neu taggt, muss angenommen werden, dass die Qualität der Parses schlechter ist als diejenige der Tags.
- Stichproben vermitteln den Eindruck, dass die Annotation mit Dependenzen nach den Guidelines von Dipper (2017) trotz des erheblich kleineren Trainingskorpus mindestens genauso korrekt sind wie mit den anderen Modellen, was möglicherweise auf die geringere Komplexität des Tagsets zurückzuführen ist.
- Es kann nicht angenommen werden, dass kurze einfache Sätze grundsätzlich eine höhere Parsing-Accuracy aufweisen als lange Sätze oder Sätze mit fehlender Interpunktion. So hat der Parser bspw. nur mit einem der drei Modelle den Drei-Wort-Satz „Lea vermisst Dodo“ korrekt geparst.

Insgesamt ist festzuhalten, dass das Dependenzparsing von nicht-standardsprachlichen Texten wie jenen aus diesem Korpus bislang nicht annähernd so zuverlässig funktioniert wie das Parsing von Standardsprache. Die Parsing-Ergebnisse in diesem Korpus sind deshalb mit der gebotenen Skepsis zu behandeln.

Die folgenden Unterkapitel gehen genauer auf die einzelnen Verarbeitungs- und Korrekturschritte beim Dependenzparsing der verschiedenen Subkorpora ein.

4.5.1 Osnabrücker Bildergeschichtenkorpus

Aus dem Osnabrücker Bildergeschichtenkorpus wurde wie beschrieben jeweils die grammatischste Version jedes Satzes geparst, d. h. die grammatischen Zielwörter. Eine Ausnahme sind Fälle, in denen das grammatische Zielwort ein leerer String ist, weil das Wort oder Satzzeichen im Text des Kindes überflüssig war. In dem Fall wurde das orthographische Zielwort oder, wenn dieses ebenfalls leer war, die Transkription geparst, damit jedes Token (abgesehen vom Zeilenumbruchszeichen ^) in der finalen LearnerXML-Datei in einer Dependenzrelation zu einem anderen Token oder dem Wurzelknoten ROOT bzw. tok0 steht.

Die Überschriften aus dem Osnabrücker Bildergeschichtenkorpus wurden gesondert an den Parser übergeben und anschließend wieder mit dem Rest des Textes zusammengesetzt, um eine Satzgrenze am Ende der Überschrift zu garantieren.

Texte, die sich aufgrund mangelnden Arbeitsspeichers nicht automatisch parsen ließen, wurden entweder satzweise automatisch oder manuell annotiert. Fehlerhafte Tokenisierung des Parsers bspw. von Zahlen, Wörtern mit Meta-Sonderzeichen, Interpunktionszeichen, usw. wurde ebenfalls manuell korrigiert. Zudem wurden problematische root-Relation des Universal-Dependencies-Modells manuell korrigiert, sodass die root-Relation nur noch zwischen der Wurzel des Satzes und dem imaginären Wurzelobjekt und nicht mehr zwischen einzelnen Wörtern eines Satzes besteht. Es ist bei dieser Korrektur auffällig, dass nicht vorhersagbar ist, in welchen Fällen und aus welchem Grund der Parser diese fehlerhaften Relationen annotiert, weshalb sich dies auch nicht verhindern lässt. Schon ein einziges verändertes Zeichen kann, wie die folgenden Beispiele zeigen, eine fehlerhafte Annotation auslösen.

nsubj(kaufe-2, ich-1)
root(ROOT-0, kaufe-2)
iobj(kaufe-2, mir-3)
dobj(kaufe-2, eine-4)
root(eine-4, neue-5)
punct(neue-5, .-6)

nsubj(kaufe-2, ich-1)
root(ROOT-0, kaufe-2)
iobj(kaufe-2, mir-3)
det(neue-5, eine-4)
dobj(kaufe-2, neue-5)
punct(kaufe-2, !-6)

Dabei ist teilweise nur der Tag der Relation falsch oder wie in diesem Beispiel die gesamte Relation. Dementsprechend wurde bei der Korrektur entweder nur der Tag ersetzt oder die Annotation eines Satzteils oder des ganzen Satzes wurde korrigiert.

4.5.2 H1 Children's Writing

Auch aus dem H1 Children's Writing Korpus wurde wie beschrieben jeweils die grammatischste Version jedes Satzes geparst, d. h. die grammatischen Zielwörter. Eine Ausnahme sind Fälle, in denen das grammatische Zielwort ein leerer String ist, weil das Wort oder Satzzeichen im Text des Kindes überflüssig war. In dem Fall wurde das orthographische Zielwort, oder wenn dieses ebenfalls leer war, die Transkription geparst, damit jedes Token in der finalen LearnerXML-Datei in einer Dependenz-relation zu einem anderen Token oder dem Wurzelknoten ROOT bzw. tok0 steht.

Texte, die sich aufgrund mangelnden Arbeitsspeichers nicht automatisch parsen ließen, wurden entweder satzweise automatisch oder manuell annotiert. Fehlerhafte Tokenisierung des Parsers bspw. von Zahlen, Wörtern mit Meta-Sonderzeichen, Interpunktionszeichen, o. Ä. wurde ebenfalls manuell korrigiert. Zudem wurden problematische root-Relationen des Universal-Dependencies-Modells manuell korrigiert, sodass die root-Relation nur noch zwischen der Wurzel des Satzes und dem imaginären Wurzelobjekt ROOT bzw. tok0 und nicht mehr zwischen einzelnen Wörtern eines Satzes besteht.

4.5.3 Litkey-Korpus

Beim Dependenzparsing wurde die orthographische Zielhypothese aus dem Litkey-Korpus geparst. Überschriften wurden gesondert übergeben, um zu garantieren, dass sie einen eigenständigen Satz ergeben. Da das Korpus als Letztes der fünf Subkorpora geparst wurde, waren bereits die enormen Probleme mit der fehlerhaften Tokenisierung von Aufzählungen und Wörtern mit Sonderzeichen bekannt. Um die Tokenisierungsfehler des Parsers zu verhindern und den manuellen Korrekturaufwand zu verringern, wurden problematische Zeichen vor dem Parsen durch ein alphabetisches Zeichen „Y“ ersetzt, das der Parser nicht von den Wörtern abtrennt. Anschließend wurde die Änderung wieder rückgängig gemacht. Vorübergehend ersetzt wurden:

- Tilden ~ vor Zielwörtern
- Fragezeichen ? vor Zielwörtern
- * in Wörtern
- Nicht-alphanumerische Zeichen in Wörtern, die mind. ein alphanumerisches Zeichen enthalten (z.B. "4." = "4Y" oder "17,5:9=?" = "17Y5Y9YY")

Zwar wurde das Parsing-Ergebnis so vermutlich etwas verfälscht. Die Auswirkungen sind jedoch als gering einzuschätzen im Vergleich mit den Auswirkungen der fehlerhaften Tokenisierung, die an all diesen Zeichen eine Tokengrenze und/oder Satzgrenze bewirkt hätte.

Einzelne Sätze, die sich aufgrund zu hohen Arbeitsspeicherverbrauchs der Stanford-Pipeline nicht automatisch parsen ließen, wurden manuell annotiert, z. B.:

(20) Dodo dachte : " Wuff wuff wuff wuff wuff "

Außerdem wurden alle Sätze korrigiert, in denen eine root-Relation zwischen zwei oder mehr Token des Satzes und nicht nur zwischen einem Token und der Wurzel bestand wie z. B. bei dem folgenden Satz:

root(ROOT-0, Lea-1)
appos(Lea-1, hat-2)
det(Eis-4, ein-3)
dobj(hat-2, Eis-4)
cc(Eis-4, aber-5)
root(aber-5, es-6)
root(es-6, hat-7)
det(Loch-9, ein-8)
dobj(hat-7, Loch-9)
punct(hat-7, .-10)

Hierbei wurde teilweise die Annotation des gesamten Satzes oder eines Satzteils korrigiert, teilweise nur die entsprechende Dependenzrelation zwischen den beiden Token, wenn lediglich der Dependenz-Tag fehlerhaft war.

4.5.4 Grundschulwki

Bei der Annotation der Dependenzen des Grundschulwikis wurde die orthographische Zielhypothese des Korpus geparst. Die annotierten Überschriften wurden separat geparst und anschließend wieder dem Text hinzugefügt. Texte, die sich mangels Arbeitsspeichers nicht automatisch parsen ließen, wurden satzweise oder manuell annotiert.

Beim Parsing des Grundschulwikis war besonders die Verarbeitung von Zahlen und Interpunktionszeichen problematisch, da Geburtsdaten oder Abkürzungen vom Parser als Satzgrenzen interpretiert wurden und so zu neuen Sätzen führten. Um diese Fehler zu beheben, wurden die erzeugten Ergebnisse automatisch mit dem bereits korrigierten tokenisierten Korpus aligniert. Das erstellte Programm vergleicht dafür das Token der geparsten Version mit dem Token der Transkription. Findet sich hier ein Unterschied, wird das nachfolgende Token des Parses dem zu vergleichenden Token hinzugefügt und anschließend wieder ein Vergleich der beiden Token durchgeführt. Dies geschieht so lange, bis die korrekt tokenisierte Version des Transkripts wiederhergestellt ist.

Durch die Zusammenführung der Token mussten sowohl die Relationen des betroffenen Tokens verändert als auch ggf. die nun zusammengeführten Sätze angepasst werden. Dies geschah zum einen durch die Übernahme der Relation, des Indexes und des Heads des nun gelöschten Tokens, bzw. wenn es sich dabei um ein Satzzeichen handelte, durch die Übernahme der Relation, des Indexes und des Heads des hinzugefügten Tokens. Zum anderen wurden, wenn nötig, die Indizes der weiteren Token des Satzes aktualisiert.

Bei der Modifikation der Relationen und Head-Token waren die durch das Zusammenfügen entstandenen mehrfachen Wurzeln in einem Satz problematisch. Um dieses Problem zu beheben, wurden die Wurzeln in ihrer Wertigkeit miteinander verglichen und die höherwertige ausgewählt. Dies geschah mithilfe eines entwickelten Rankings der Tags (absteigend sortiert nach favorisiertem Tag):

VVFIN -> VVPP -> VVINF -> VVIMP -> VVIZU -> VMFIN -> VMPP -> VMINF -> VAFIN -> VAIMP -> VAINF -> ADJD -> NN -> NE -> PDS -> PPER -> PIS -> ADJA

Fehlte einer der beiden Wurzel-Tags in dem gegebenen Ranking, wurde die Wurzel mit dem in der Liste vorhandenen Tag ausgewählt. Befanden sich beide Wurzeln nicht in der Liste, wurde die erste Wurzel favorisiert. Anschließend wurde die höchstwertige Wurzel als Head-Token der anderen, nicht mehr akzeptierten Wurzeln des Satzes und im Falle der Dipper-Depenzen als Head der Satzzeichen gesetzt. Die nun nicht mehr gültige Relation ROOT bzw. root der Token wurde durch dep, xy bzw. -UNKNOWN- ersetzt.

Die mehrfachen root-Relationen des Universal-Dependencies-Modell und die als Root fungierenden Satzzeichen wurden manuell korrigiert, die automatische Korrektur erzielte hier aufgrund der Satzkomplexität keine akzeptablen Ergebnisse.

Die folgende Abbildung illustriert die Modifikation der Satzgrenzen und Dependenzen. Bedingt durch die zuerst durchgeführte Modifikation der nicht als Satzgrenze fungierenden Satzzeichen, konnten vor der Nutzung des Rankings bereits drei vermeintliche Wurzeln gelöscht werden.

```
xy(ist-3, Galileo-1)
subj(ist-3, Galilei-2)
root(ROOT-0, ist-3)
pcase-det(15-5, am-4)
mod(ist-3, 15-5)
punct(ist-3, .-6)

root(ROOT-0, 2-1)
punct(2-1, .-2)

xy(geboren-4, 1564-1)
pcase(Pisa-3, in-2)
mod(geboren-4, Pisa-3)
root(ROOT-0, geboren-4)
cc(gestorben-6, und-5)
conj(geboren-4, gestorben-6)
aux(geboren-4, ist-7)
subj(ist-7, er-8)
pcase-det(8-10, am-9)
mod(er-8, 8-10)
punct(geboren-4, .-11)

root(ROOT-0, 1-1)
punct(1-1, .-2)

mod(Florenz-3, 1642-1)
mod(Florenz-3, nahe-2)
root(ROOT-0, Florenz-3)
punct(Florenz-3, .-4)
```

```
xy(ist-3, Galileo-1)
subj(ist-3, Galilei-2)
aux(geboren-10, ist-3)
pcase-det(15-5, am-4)
mod(2.-6, 15.-5)
mod(1564-7, 2.-6)
mod(geboren-10, 1564-7)
pcase(Pisa-9, in-8)
mod(geboren-10, Pisa-9)
root(ROOT-0, geboren-10)
cc(gestorben-10, und-11)
conj(geboren-10, gestorben-12)
aux(gestorben-12, ist-13)
subj(ist-13, er-14)
pcase-det(8.-16, am-15)
mod(gestorben-12, 8.-16)
mod(8.-16, 1.-17)
mod(Florenz-20, 1642-18)
mod(Florenz-20, nahe-19)
dep(geboren-10, Florenz-20)
punct(geboren-10, .-21)
```

Abbildung 15: Satzgrenzen des Textes "Galileo Galilei" vor und nach der Korrektur

4.5.5 Klexikon

Beim Dependenzparsing des Klexikons wurde, wie dem Abschnitt 4.5 zu entnehmen, die grammatische Zielhypothese des Korpus geparst. Auch in diesem Korpus verursachte der Parser Fehler. Sind diese mangels Arbeitsspeicher entstanden, wurden die betreffenden Texte satzweise automatisch oder manuell annotiert. Bei der fehlerhaften Tokenisierung fand sowohl eine automatische als auch eine manuelle Korrektur Anwendung.

Ebenso wie bei der im Abschnitt 4.5.4 beschriebenen Modifikation mussten die Resultate des Parsers mit der bestehenden Tokenisierung aligniert und die Satzgrenzen modifiziert werden. Der entwickelte automatische Algorithmus fügte dabei, wie bereits beschrieben, zur Wiederherstellung der ursprünglichen Tokenisierung Token zu ihren vorhergehenden Token hinzu, modifizierte Head-Token und Relationen und passte die Indizes der Sätze an. Bei der Modifikation der Heads bzw. der Auswahl der Wurzel wurde hier ebenso das erstellte Ranking der Tags hinzugezogen und die nicht favorisierten Wurzeln wurden der favorisierten untergeordnet.

Nur manuell korrigierbar war die problematische root-Relation des Universal-Dependencies-Modell, d.h. die root-Relation zwischen zwei gewöhnlichen Token, da eine automatische Korrektur aufgrund der Satzkomplexität keine akzeptablen Ergebnisse erzielte.

Das folgende Beispiel zeigt die ursprünglich geparste Version des Textes "Abitur" und die anschließende korrigierte Version.

```
det(Abitur-2, Das-1)
root(ROOT-0, Abitur-2)
punct(Abitur-2, ,-3)
mark(kennen-8, wie-4)
nsubj(kennen-8, wir-5)
dobj(kennen-8, es-6)
advmod(kennen-8, heute-7)
advcl(19-12, kennen-8)
punct(19-12, ,-9)
auxpass(19-12, wurde-10)
advmod(19-12, im-11)
ccomp(Abitur-2, 19-12)
punct(Abitur-2, .-13)

dep(erfunden-2, Jahrhundert-1)
root(ROOT-0, erfunden-2)
punct(erfunden-2, .-3)
```

```
det(Abitur-2, Das-1)
dep(erfunden-14, Abitur-2)
punct(Abitur-2, ,-3)
mark(kennen-8, wie-4)
nsubj(kennen-8, wir-5)
dobj(kennen-8, es-6)
advmod(kennen-8, heute-7)
advcl(19-12, kennen-8)
punct(19-12, ,-9)
auxpass(19-12, wurde-10)
advmod(19-12, im-11)
ccomp(Abitur-2, 19.-12)
dep(erfunden-14, Jahrhundert-13)
root(ROOT-0, erfunden-14)
punct(erfunden-14, .-15)
```

Abbildung 16: Satzgrenzen im Text "Abitur" vor und nach der Korrektur

4.6 Satzgrenzen

Über die tokenbasierte Betrachtung der Korpora hinausgehend wurden die möglichen Satzgrenzen in Form der Boolean-Attribute *SentBeginStanford* und *SentBeginPunct* an den Satzanfängen annotiert, um nicht nur eine bessere Nutzung der Dependenzen zu gewährleisten, die sich immer nur auf einen einzelnen Satz beziehen können, sondern auch weitere Forschung z. B. hinsichtlich der Satzkomplexität zu ermöglichen. Während *SentBeginStanford* Bezug auf die vom Stanford-Dependenz-Parser erzeugten Satzgrenzen in Form von Leerzeilen in seinem Output nimmt, wird die Annotation von *SentBeginPunct* ausschließlich durch die Satzzeichen im Text bestimmt. Als Satzende wird so das letzte als $. getaggte Token gesetzt, als Satzanfang das nächste nicht als $., $, oder $(getaggte Token, welches nicht eines der Meta-Sonderzeichen ^ oder \h ist. Dabei werden Doppelpunkte nicht als Satzende gewertet, öffnende Klammern nach dem Satzende jedoch als Satzanfang. Die Annotation der Satzgrenzen wird noch vor der Konvertierung ins LearnerXML-Format durchgeführt.

Während *SentBeginStanford* in allen Subkorpora vorhanden ist, da jedes Korpus mit dem Stanford-Parser geparst wurde, konnte die Annotation des Attributs *SentBeginPunct* nur bei Existenz einer grammatischen Zielhypothese durchgeführt werden, da nur so die Korrektheit der Interpunktion vorausgesetzt werden kann. *SentBeginPunct* wurde aus diesem Grund nur im Klexikon-Korpus, im Osnabrücker Bildergeschichtenkorpus und im H1 Children's Writing Korpus annotiert.

Das folgende Beispiel zeigt einen Ausschnitt der csv-Datei des Textes "Totes Meer" aus dem Grundschulwiki, wobei sich das Attribut *SentBeginStanford* in Spalte 6 befindet. Spalte 7 für das Attribut *SentBeginPunct* ist dagegen leer.

tok5	Das	Das	ART	true		tok7	det	tok7	DET	tok7	det	
tok6	Tote	Tote	ADJA	false		tok7	mod	tok7	ATTR	tok7	amod	
tok7	Meer	Meer	NN	false		tok11	subj	tok8	SUBJ	tok9	nsubj	
tok8	ist	ist	VAFIN	false		tok11	aux	tok0	ROOT	tok9	cop	
tok9	weit	weit	ADJD	false		tok11	subj	tok8	PRED	tok0	root	
tok10	und	und	KON	false		tok11	cc	tok8	KON	tok9	cc	
tok11	breit	breit	ADJD	false		tok0	root	tok10	CJ	tok13	advmod	
tok12	ganz	ganz	ADV	false		tok13	mod	tok13	ADV	tok13	advmod	
tok13	still	still	ADJD	false		tok11	mod	tok11	ADV	tok9	conj	
tok14	.	.	$.	false		tok11	punct	tok13	-PUNCT-	tok9	punct	
tok15	Es	Es	PPER	true		tok16	subj	tok16	SUBJ	tok16	nsubj	
tok16	gibt	gibt	VVFIN	false		tok0	root	tok0	ROOT	tok0	root	
tok17	keine	keine	PIAT	false		tok18	det	tok18	DET	tok18	neg	
tok18	Lebewesen	Lebewesen	NN	false				tok16	obj	tok16	OBJA	tok16 dobj
tok19	.	.	$.	false		tok16	punct	tok18	-PUNCT-	tok16	punct	

Abbildung 17: Ausschnitt der csv-Datei zu "Totes Meer"

Ebenso wie das Tagging und das Dependenz-Parsing kann das Annotieren der Satzgrenzen von Kindertexten nur mit Einschränkungen durchgeführt werden, da in vielen Fällen wie in (21) und (22) keine oder nur fehlerhafte Satzzeichen existieren und die Lokalisierung der Satzgrenzen durch grammatische Fehler und fehlende oder überflüssige Wörter z. T. stark erschwert wird.

(21) Lea hat ihren Hund verloren sie Plakate sie . Tut alles was sie tun . Kann

(22) Lea klebt Poster auf die Poster steht …

Darüber hinaus verursachte der Parser vereinzelt falsche Satzgrenzen z. B. in Bezug auf Geburtsdaten ("26. Februar 1986"). So wurde in einigen Fällen der entsprechende Punkt als Satzgrenze gesetzt. Im Rahmen der Korrektur dieser Fehler wurden sowohl automatisch Sätze zusammengefügt als auch manuell Fehler bereinigt.

In den nachfolgenden Abschnitten wird beschrieben, wie die Annotation der Satzgrenzen einzelner Subkorpora durchgeführt wurde, welches der Satzgrenzen-Attribute für die jeweiligen Korpora existiert und welche Korrekturen manuell oder automatisch durchgeführt wurden.

4.6.1 Osnabrücker Bildergeschichtenkorpus

Für das Osnabrücker Bildergeschichtenkorpus existieren sowohl die vom Stanford-Parser annotierten *beginSentStanford*-Satzgrenzen als auch jene, die aus der Interpunktion der grammatischen Zielhypothese abgeleitet sind. Die vom Stanford-Parser produzierten Satzgrenzen wurden bei der manuellen Nachbearbeitung vereinzelt korrigiert, um die korrekte Tokenisierung der Texte wiederherzustellen oder eine korrekte Dependenzannotation zu ermöglichen.

Für die Ermittlung der interpunktionsbasierten Satzgrenzen wurden die in Abschnitt 4.6 beschriebenen Regeln auf die grammatische Zielhypothese angewendet. Als Sonderregel kommt für dieses Korpus hinzu, dass Anführungszeichen, die die direkte Rede einleiten, der Satzanfang sein können, wenn sie das erste Zeichen nach dem Satzende sind. Die Anwendung dieser Regel wird durch die vorhandene Annotation der direkten Rede im Osnabrücker Bildergeschichtenkorpus ermöglicht (s. Abschnitt 4.7.1). Wie die Satzgrenzen des Stanford-Parsers wurden auch die Interpunktions-Satzgrenzen in vereinzelten Fällen manuell nachkorrigiert.

Die bereits im Osnabrücker Bildergeschichtenkorpus enthaltenen manuell annotierten Satzgrenzen wurden, wie bereits in Abschnitt 4.1.1 beschrieben, als unzuverlässig erachtet und deshalb nicht in die LearnerXML-Dateien übernommen. Da nicht bekannt ist, nach welchen Regeln sie annotiert sind, und die Annotation als Satzgrenze deshalb undurchsichtig und scheinbar inkonsistent ist, bieten sie keinen wirklichen Mehrwert gegenüber den beiden automatischen Satzgrenzenannotationen.

4.6.2 H1 Children's Writing

Für das H1 Children's Writing Korpus existieren wie beim Osnabrücker Bildergeschichtenkorpus sowohl die vom Stanford-Parser annotierten Satzgrenzen als auch jene aus der Interpunktion der grammatischen Zielhypothese abgeleiteten. Auch hier wurden die vom Stanford-Parser produzierten Satzgrenzen bei der manuellen Nachbearbeitung vereinzelt korrigiert, um die korrekte Tokenisierung der Texte zu rekonstruieren oder eine korrekte Dependenzannotation zu ermöglichen. Für die Ermittlung der interpunktionsbasierten Satzgrenzen wurden die in Abschnitt 4.6 beschriebenen Regeln auf die grammatische Zielhypothese angewendet. Auch diese Satzgrenzen wurden in Einzelfällen manuell korrigiert.

4.6.3 Litkey-Korpus

Das Litkey-Korpus ist nur mit den vom Stanford-Parser erzeugten Satzgrenzen annotiert. Da für dieses Subkorpus keine grammatische Zielhypothese vorhanden ist, kann die Interpunktion nicht als ausreichend zuverlässig erachtet werden, um darauf basierend nützliche Satzgrenzen zu bestimmen. Wie bereits an den (hier wiederholten) Beispielen aus Abschnitt 4.6 deutlich wurde, sind die Satzgrenzen selbst für einen menschlichen Annotator in Fällen wie (21) und (22) schwer zu bestimmen.

(21) Lea hat ihren Hund verloren sie Plakate sie . Tut alles was sie tun . Kann

(22) Lea klebt Poster auf die Poster steht …

Eine Annotation basierend auf den Satzzeichen würde für beide Beispiele keine annähernd korrekten Satzgrenzen erzeugen. Da die Satzgrenzenerkennung der Stanford-Core-NLP-Pipeline letztlich jedoch ebenfalls auf Interpunktionszeichen basiert, kann auch der Stanford-Parser in einem Text wie dem folgenden (dargestellt ist die orthographische Zielhypothese) keine sinnvollen Satzgrenzen erkennen:

(23) Lea und Lars holen sich ein Eis mit Dodo wollte auch ein Eis haben aber sie haben nicht gemerkt dass Dodo auch ein Eis wollte und dann hat Lea Eis getropft und Dodo war glücklich dass er auch Eis bekommen hat und dann ist Lea gefallen und dann war Dodo richtig glücklich dann gab Lars ihr was ab und dann war Lea sehr glücklich und Lars und Dodo iss ganz schnell das Eis auf und Lea packte Lars an und dann gingen sie nach Hause und Dodo hinterher .

Vereinzelt wurden die Satzgrenzen des Stanford-Parsers bei der manuellen Korrektur des Parsings nachkorrigiert, um sinnvolle Dependenzanalysen zu ermöglichen. Betroffen waren davon v. a. Satzzeichencluster sowie Anführungszeichen, die beim Parsen häufig dem falschen Satz zugeordnet

wurden. Texte, in denen die Kinder bspw. an jedem Zeilenende einen Punkt gesetzt haben, der vom Parser als Satzende angenommen wurde, konnten aufgrund des enormen damit verbundenen Aufwands sowie der allgemeinen Herausforderungen bei der Satzgrenzenerkennung in Kindertexten nicht korrigiert werden. Auch die *SentBeginStanford*-Satzgrenzen sollten somit im Litkey-Korpus nicht unbedingt als zuverlässig vorausgesetzt werden.

4.6.4 Grundschulwiki

Da im Grundschulwiki keine grammatische Zielhypothese annotiert wurde, bezieht sich die Annotation der Satzgrenzen ausschließlich auf die Satzgrenzen des Stanford-Parsers. Die Grenzen wurden bei der Verarbeitung den geparsten Dateien entnommen, wo sie als Leerzeilen annotiert wurden. Die folgenden Abbildungen zeigen einen Ausschnitt des Textes "Lukas Podolski" sowohl als geparste Datei nach Dipper (2017) als auch als spätere csv-Datei mit annotierten Satzgrenzen, hier gekennzeichnet durch "true" und "false".

```
root(ROOT-0, Lukas-1)
mod(Gewicht-3, Podolski-2)
xy(Lukas-1, Gewicht-3)
punct(Lukas-1, ,-4)
xd-mod(Lukas-1, Aussehen-5)
cc(Geburtstag-7, und-6)
conj(Lukas-1, Geburtstag-7)
punct(Lukas-1, :-8)
xy(Lukas-1, \h-9)

xy(geboren-12, Lukas-1)
subj(geboren-12, Podolski-2)
aux(geboren-12, ist-3)
pcase-det(4.-5, am-4)
mod(ist-3, 4.-5)
subj(geboren-12, Juni-6)
mod(geboren-12, 1985-7)
pcase(Gleiwitz-9, in-8)
mod(geboren-12, Gleiwitz-9)
obj(geboren-12, -LRB--10)
subj(geboren-12, Polen-6)
obj(geboren-12, -RRB--11)
root(ROOT-0, geboren-12)
punct(geboren-12, .-13)
```

tok1	Lukas	Lukas		NE	true
tok2	Podolski	Podolski		NE	false
tok3	Gewicht	Gewicht		NN	false
tok4	,	,		$,	false
tok5	Aussehen	Aussehen		NN	false
tok6	und	und		KON	false
tok7	Geburtstag	Geburtstag		NN	false
tok8	:	:		$(	false
tok9	\h	\h		XY	false
tok10	Lukas	Lukas		NE	true
tok11	Podolski	Podolski		NE	false
tok12	ist	ist		VAFIN	false
tok13	am	am		APPRART	false
tok14	4.	4.		ADJA	false
tok15	Juni	Juni		NN	false
tok16	1985	1985		CARD	false
tok17	in	in		APPR	false
tok18	Gleiwitz	Gleiwitz		NE	false
tok19	(	(		$(	false
tok20	Polen	Polen		NE	false
tok21	)	)		$(	false
tok22	geboren	geboren		VVPP	false
tok23	.	.		$.	false

Abbildung 18: Ausschnitt des Texts "Lukas Podolski" in Form der korrigierten Parsing-Datei und der späteren csv-Datei

Durch die hohe Frequenz von Punkten (z. B. in Geburtsdaten) produzierte die Tokenisierung des Parsers Fehler bei der Abgrenzung der Sätze. Die entstandenen nicht akzeptierten Satzgrenzen wurden automatisch mit dem bereits in Abschnitt 4.5.4 beschriebenen Algorithmus, welcher die geparsten und tokenisierten Texte aligniert, modifiziert und gegebenenfalls manuell ergänzt.

Die folgenden Abbildungen zeigen die gesetzten Satzgrenzen vor der entsprechenden Korrektur (links) und die Satzgrenzen nach der entsprechenden Korrektur (rechts).

```
root(ROOT-0, Lukas-1)
mod(Gewicht-3, Podolski-2)
xy(Lukas-1, Gewicht-3)
punct(Lukas-1, ,-4)
xd-mod(Lukas-1, Aussehen-5
cc(Geburtstag-7, und-6)
conj(Lukas-1, Geburtstag-7
punct(Lukas-1, :-8)
xy(Lukas-1, \h-9)

xy(ist-3, Lukas-1)
subj(ist-3, Podolski-2)
root(ROOT-0, ist-3)
pcase-det(4-5, am-4)
mod(ist-3, 4-5)
punct(ist-3, .-6)

subj(geboren-8, Juni-1)
mod(geboren-8, 1985-2)
pcase(Gleiwitz-4, in-3)
mod(geboren-8, Gleiwitz-4)
obj(geboren-8, -LRB--5)
subj(geboren-8, Polen-6)
obj(geboren-8, -RRB--7)
root(ROOT-0, geboren-8)
punct(geboren-8, .-9)
```

```
root(ROOT-0, Lukas-1)
mod(Gewicht-3, Podolski-2)
xy(Lukas-1, Gewicht-3)
punct(Lukas-1, ,-4)
xd-mod(Lukas-1, Aussehen-5)
cc(Geburtstag-7, und-6)
conj(Lukas-1, Geburtstag-7)
punct(Lukas-1, :-8)
xy(Lukas-1, \h-9)

xy(geboren-12, Lukas-1)
subj(geboren-12, Podolski-2)
aux(geboren-12, ist-3)
pcase-det(4.-5, am-4)
mod(ist-3, 4.-5)
subj(geboren-12, Juni-6)
mod(geboren-12, 1985-7)
pcase(Gleiwitz-9, in-8)
mod(geboren-12, Gleiwitz-9)
obj(geboren-12, -LRB--10)
subj(geboren-12, Polen-6)
obj(geboren-12, -RRB--11)
root(ROOT-0, geboren-12)
punct(geboren-13, .-13)
```

Abbildung 19: Satzgrenzen des Textes "Lukas Podolski" vor und nach der Korrektur

```
xy(ist-3, Galileo-1)
subj(ist-3, Galilei-2)
root(ROOT-0, ist-3)
pcase-det(15-5, am-4)
mod(ist-3, 15-5)
punct(ist-3, .-6)

root(ROOT-0, 2-1)
punct(2-1, .-2)

xy(geboren-4, 1564-1)
pcase(Pisa-3, in-2)
mod(geboren-4, Pisa-3)
root(ROOT-0, geboren-4)
cc(gestorben-6, und-5)
conj(geboren-4, gestorben-6)
aux(geboren-4, ist-7)
subj(ist-7, er-8)
pcase-det(8-10, am-9)
mod(er-8, 8-10)
punct(geboren-4, .-11)

root(ROOT-0, 1-1)
punct(1-1, .-2)

mod(Florenz-3, 1642-1)
mod(Florenz-3, nahe-2)
root(ROOT-0, Florenz-3)
punct(Florenz-3, .-4)
```

```
xy(ist-3, Galileo-1)
subj(ist-3, Galilei-2)
aux(geboren-10, ist-3)
pcase-det(15-5, am-4)
mod(2.-6, 15.-5)
mod(1564-7, 2.-6)
mod(geboren-10, 1564-7)
pcase(Pisa-9, in-8)
mod(geboren-10, Pisa-9)
root(ROOT-0, geboren-10)
cc(gestorben-10, und-11)
conj(geboren-10, gestorben-12)
aux(gestorben-12, ist-13)
subj(ist-13, er-14)
pcase-det(8.-16, am-15)
mod(gestorben-12, 8.-16)
mod(8.-16, 1.-17)
mod(Florenz-20, 1642-18)
mod(Florenz-20, nahe-19)
dep(geboren-10, Florenz-20)
punct(geboren-10, .-21)
```

Abbildung 20: Satzgrenzen des Textes "Galileo Galilei" vor und nach der Korrektur

4.6.5 Klexikon

Im Gegensatz zum Grundschulwiki wurde bei der Annotation des Klexikons die Transkription als orthografische und grammatische Zielhypothese übernommen. Da davon auszugehen ist, dass die Markierung der Satzgrenzen korrekt ist, konnte das Attribut *SentBeginPunct* ebenfalls annotiert werden. Weil dieses Attribut mithilfe des Tags $. annotiert wird, wurde zunächst automatisch sichergestellt, dass Satzzeichen wie <.>, <!> und <?> als $. getaggt sind. Anschließend konnten, wie in Abschnitt 4.6 beschrieben, anhand der entsprechenden Regeln die Satzgrenzen erkannt und das nachfolgende Token als Satzanfang markiert werden.

Um das Attribut *SentBeginStanford* zu annotieren, wurden zuvor, wie dem Abschnitt 4.5.4 zu entnehmen, die bereits tokenisierte Version des Korpus und die Parsingergebnisse automatisch aligniert und die Satzgrenzen modifiziert. Anschließend wurden die Leerzeilen des geparsten Texts eingelesen und die Satzanfänge markiert.

Die folgende Abbildung zeigt in Anlehnung an Abschnitt 4.6.4 eine Übersicht der geparsten Datei, der korrigierten Datei, der getaggten Datei sowie der anschließend entstandenen csv-Datei des Textes "Bern":

```
pcase(19-3, Seit-1)
det(19-3, dem-2)
root(ROOT-0, 19-3)
punct(19-3, .-4)

subj(gelegt-7, Jahrhundert-1)
aux(gelegt-7, wurde-2)
det(Aare-4, die-3)
obj(gelegt-7, Aare-4)
pcase(Kanäle-6, in-5)
mod(gelegt-7, Kanäle-6)
root(ROOT-0, gelegt-7)
cc(umgeleitet-12, und-8)
pcase(Bielersee-11, in-9)
det(Bielersee-11, den-10)
mod(umgeleitet-12, Bielersee-11)
conj(gelegt-7, umgeleitet-12)
punct(gelegt-7, .-13)
```

```
pcase(19-3, Seit-1)
det(19-3, dem-2)
dep(gelegt-10, 19.-3)
subj(gelegt-10, Jahrhundert-4)
aux(gelegt-10, wurde-5)
det(Aare-7, die-6)
obj(gelegt-10, Aare-7)
pcase(Kanäle-9, in-8)
mod(gelegt-10, Kanäle-9)
root(ROOT-0, gelegt-10)
cc(umgeleitet-15, und-11)
pcase(Bielersee-14, in-12)
det(Bielersee-14, den-13)
mod(umgeleitet-15, Bielersee-14)
conj(gelegt-10, umgeleitet-15)
punct(gelegt-10, .-16)
```

```
Seit        APPR
dem         ART
19.         CARD
Jahrhundert         NN
wurde       VAFIN
die         ART
Aare        NN
in          APPR
Kanäle      NN
gelegt      VVPP
und         KON
in          APPR
den         ART
Bielersee           NE
umgeleitet          VVPP
.           $.
```

tok559	Seit	Seit	Seit	APPR	true	true
tok560	dem	dem	dem	ART	false	false
tok561	19.	19.	19.	CARD	false	false
tok562	Jahrhundert	Jahrhundert	Jahrhundert	NN	false	false
tok563	wurde	wurde	wurde	VAFIN	false	false
tok564	die	die	die	ART	false	false
tok565	Aare	Aare	Aare	NN	false	false
tok566	in	in	in	APPR	false	false
tok567	Kanäle	Kanäle	Kanäle	NN	false	false
tok568	gelegt	gelegt	gelegt	VVPP	false	false
tok569	und	und	und	KON	false	false
tok570	in	in	in	APPR	false	false
tok571	den	den	den	ART	false	false
tok572	Bielersee	Bielersee	Bielersee	NE	false	false
tok573	umgeleitet	umgeleitet	umgeleitet	VVPP	false	false
tok574	.	.	.	$.	false	false

Abbildung 21: Ausschnitt des Textes "Bern" vor und nach der Korrektur der Satzgrenzen, nach dem Tagging, sowie als csv-Datei

4.7 Direkte Rede

Neben der Markierung von Satzgrenzen ist in diesem Korpus auch die Möglichkeit vorgesehen, direkte Rede im Text zu markieren. Im LearnerXML-Format kann diese Information im Attribut *directSpeech* mit den Werten "begin" und "end" codiert werden. Je nachdem ob im Text Anführungszeichen vorhanden sind oder nicht und abhängig davon, ob eine grammatische oder nur eine orthographische Zielhypothese existiert, ist es möglich, das Attribut dem Anführungszeichen-Token zuzuweisen oder den entsprechenden Token, mit denen die direkte Rede beginnt bzw. endet.

In den fünf Subkorpora ist die direkte Rede lediglich im Osnabrücker Bildergeschichtenkorpus annotiert, da die Information dort bereits enthalten war. Im Abschnitt zu diesem Subkorpus wird somit auf die Verarbeitung der Annotation und einige problematische Aspekte Bezug genommen.

Für Korpora mit einer grammatischen Zielhypothese könnte die direkte Rede basierend auf der Interpunktion ebenfalls automatisch annotiert werden. Die Abschnitte zum H1 Children's Writing Korpus und zum Klexikon-Korpus werden darlegen, warum dies im jeweiligen Fall jedoch nicht sinnvoll oder möglich war.

In Korpora ohne grammatische Zielhypothese, d. h. mit nur einer orthographischen Normalisierung, könnte die direkte Rede lediglich mit hohem Aufwand manuell annotiert werden, was hier nicht durchgeführt wurde. Die Abschnitte zum Litkey-Korpus und Grundschulwiki werden auf die besonderen Herausforderungen und Probleme, die mit direkter Rede in Kindertexten verbunden sind, eingehen.

4.7.1 Osnabrücker Bildergeschichtenkorpus

Das Osnabrücker Bildergeschichtenkorpus ist das einzige Subkorpus, in dem die direkte Rede bereits gesondert markiert ist. Dies geschieht in der XML-Datei durch den <rede>-Tag, der die gesamte direkte Rede einschließt. Dies kann zum Beispiel wie folgt aussehen:

```
er
<f typ="gks"> Hat </f>
&ze;
<f ortho="gefragt"> gefr akt </f>
<rede>
        <f ortho="willst"> wielst </f>
        <f typ="gks"> Du </f>
        mit mir
        <f ortho="kommen"> komen </f>
        &ze;
        <punkt vorhanden="nein" zeichen="frage"/>
</rede>
```

Abbildung 22: Direkte Rede im Osnabrücker Bildergeschichtenkorpus

Zusätzlich kann der Tag durch das Attribut *markiert* mit den Werten "ja", "nein", "anfang" oder "ende" weitere Auskunft über das zur Anzeige der direkten Rede verwendete Zeichen geben. Tatsächlich tritt <rede markiert="ja"> im Korpus insgesamt 27 Mal auf und <rede markiert="anfang"> ein Mal. Fehlt das Attribut, entspricht dies <rede markiert="nein">. Dieser Fall kommt im Korpus 323 Mal vor.

Im Zuge der Transkription wurde die Annotation der direkten Rede, wie bereits in Abschnitt 4.1.1 beschrieben, aus den XML-Dateien extrahiert und in Form eines vorläufigen Markers für Anfang bzw. Ende der direkten Rede vor bzw. nach dem betroffenen Token zwischengespeichert. Anschließend

wurde die Information in die Werte "begin" und "end" umgesetzt und dem zugehörigen Token als Attribut hinzugefügt. In der csv-Datei, die als Input für die Umwandlung in LearnerXML dient, befindet sich die Information, wie im nachfolgenden Beispiel zu erkennen ist, in der achten Spalte. Im LearnerXML-Format befindet sich die Information als Attribut an dem entsprechenden Token, das die direkte Rede einleitet oder beendet.

```
tok63                   "       $(      true    true    begin   tok71   punct   tok64   -PUNCT- tok71   punct
tok64   Na      Na      Na      ITJ     false   false           tok71   xd-cc   tok0    ROOT    tok71   advmod
tok65   gut     gut     gut     ADV     false   false           tok71   xd-mod  tok66   ADV     tok71   advmod
tok66   ich     ich     ich     PPER    false   false           tok71   subj    tok67   SUBJ    tok71   nsubj
tok67   haBe    habe    habe    VAFIN   false   false           tok71   aux     tok64   -UNKNOWN-       tok71   aux
tok68   So      so      so      ADV     false   false           tok69   mod     tok69   ADV     tok71   advmod
tok69   File    viele   viele   PIAT    false   false           tok71   det     tok71   DET     tok71   amod
tok70   ^
tok71   maten   Matten  Matten  NN      false   false           tok0    root    tok67   OBJA    tok0    root
tok72                   .       $.      false   false           tok71   punct   tok71   -PUNCT- tok71   punct
tok73   er      er      er      PPER    true    true            tok76   subj    tok74   SUBJ    tok76   nsubj
tok74   kan|    kann    kann    VMFIN   false   false           tok76   aux     tok0    ROOT    tok76   aux
tok75   sie     sie     sie     PPER    false   false           tok76   subj    tok76   OBJA    tok76   nsubj
tok76   Behalte behalten        behalten        VVINF   false   false           tok0    root    tok74   AUX     tok0    root
tok77   ^
tok78                   .       $.      false   false           tok76   punct   tok76   -PUNCT- tok76   punct
tok79                   "       $(      false   false   end     tok76   punct   tok76   -PUNCT- tok76   punct
tok80   ENDE    ENDE    ENDE    NN      true    true            tok0    root    tok0    ROOT    tok0    root
```

Abbildung 23: csv-Datei eines Textes aus dem Osnabrücker Bildergeschichtenkorpus

Dass die direkte Rede im Osnabrücker Bildergeschichtenkorpus annotiert ist, bedeutet allerdings nicht, dass sicher alle Fälle direkter Rede erfasst sind. So lassen sich auch Gegenbeispiele wie das folgende finden, wo in der grammatischen Zielhypothese keine Anführungszeichen o. Ä. annotiert sind und auch eine Markierung des entsprechenden Satzteils als direkte Rede fehlt:

(24) da hat der Mann den Hund in der Hundehütte gesehen . sagte er kann ich wieder meine Fußmatte wiederhaben , weil ich das möchte .

Wie oft dieses Problem im gesamten Korpus auftritt, könnte nur mit einer vollständigen manuellen Überprüfung festgestellt werden, da die Fälle ohne die fehlenden Anführungszeichen anders nicht auffindbar sind.

4.7.2 H1 Children's Writing

Obwohl das H1 Children's Writing Korpus eine grammatische Zielhypothese beinhaltet, konnte für dieses Korpus keine automatische Annotation der direkten Rede durchgeführt werden. Dies liegt daran, dass in der grammatischen Zielhypothese keine Anführungszeichen annotiert sind und auch eine Markierung der entsprechenden Sätze als direkte Rede nicht vorhanden ist. Dies wird auch an dem bereits angeführten Beispiel sichtbar:

(25) Als Mama in das Badezimmer kommt muss sie lachen . Sie sagt hier ist ein Frisör eingekehrt .

Während andere in der Transkription fehlende Interpunktionszeichen ergänzt wurden, ist dies für Anführungszeichen offensichtlich nicht der Fall. Das gesamte H1 Children's Writing Korpus enthält somit kein einziges Anführungszeichen und lediglich in 97 Fällen hat ein Kind wie im folgenden Beispiel einen Doppelpunkt in seinem Text verwendet, um die direkte Rede anzuzeigen:

(26) Sie sagt : Wir gehen zum Arzt .

Folglich lässt sich die direkte Rede nicht automatisch basierend auf der grammatischen Zielhypothese annotieren. Das gesamte Korpus müsste zunächst manuell mit den entsprechenden Interpunktionszeichen angereichert werden, bevor dies möglich wäre. Dass eine solche Annotation nicht unbedingt trivial ist, da häufig unterschiedliche Interpretationen dahingehend möglich sind, ob es sich um direkte Rede handelt und wo diese beginnt oder endet, zeigt der folgende Textausschnitt (dargestellt ist die grammatische Zielhypothese):

(27) Mama telefoniert immer noch . Teo geht ins Biro ! Mama schrie aaaaah! Was ist denn los ? Da drüben ist ein Gespenst .

4.7.3 Litkey-Korpus

Für das Litkey-Korpus ist keine Annotation der direkten Rede vorhanden. Da das Korpus lediglich mit einer orthographischen Zielhypothese annotiert ist, kann keine zuverlässige automatische Erkennung der direkten Rede für dieses Korpus durchgeführt werden. Häufig verwenden die Kinder Anführungszeichen nur zur Markierung des Anfangs der direkten Rede wie in (28) oder seltener auch am Ende bzw. meistens gar nicht wie in (29).

(28) Er sagte : " Lea ich habe dein Hund . Lea hat sich sehr gefreut .

(29) Sie wartet bis ein Anruf kam da klingelte das Telefon . Lars was ist ich habe dein Hund Dodo gefunden echt das ist ja toll dass du ihn gefunden hast wo war er denn . " Er saß an der Straße . Ok bringst du ihn mir in zehn min ja klar tschüss .

Darüber hinaus gibt es Fälle, in denen die Kinder wie in Beispiel (30) indirekte Rede als direkte Rede markieren.

(30) die Lehrerin hat nicht geschimpft sie hat nur gesagt : " dass sie Dodo nicht mehr mit nehmen soll . "

Zum Teil haben Anführungszeichen zudem gar keinen Bezug zu wörtlicher Rede wie in (31).

(31) Die Lehrerin kommt aber sie schimpft nicht , sie " lacht " .

Auch in Texten von Kindern, die jeden Satz in Anführungszeichen setzen wie in der folgenden Abbildung, würde eine automatische Erkennung der direkten Rede unsinnige Ergebnisse erzielen.

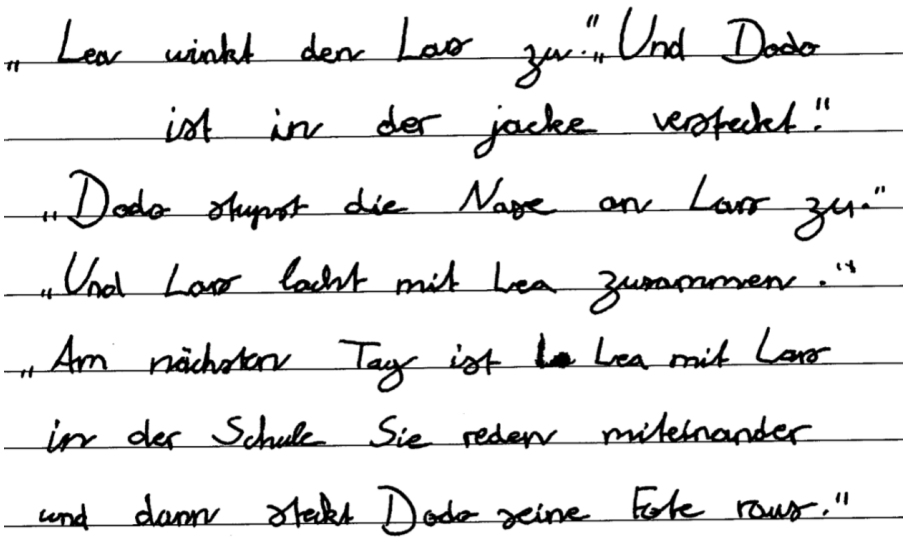

Abbildung 24: Ausschnitt aus einem Original-Scan aus dem Litkey-Korpus, in dem alle Sätze in Anführungszeichen gesetzt sind

Eine Orientierung an weiteren Satzzeichen wie dem Doppelpunkt wäre ebenfalls nicht sehr hilfreich, da viele Kinder fehlerhafte Satzzeichencluster oder Satzzeichen an falschen Stellen produzieren, wenn sie die Zeichensetzungsregeln noch nicht beherrschen wie in (32) bis (34).

(32) Lea ~rufte : " Halt halt meine Tasche ist im Bus : " .

(33) Nach einer Weile kam Lea mit Dodo nach Fundbüro und sagte : " zu den Mann haben Sie ein Tasche

(34) Am nächsten Tag ging Lea mit Dodo zum Fundbüro und sagte : " Die Nummer 38 gehört mir " : sagte Lea und dann sagte der Fundbüromann : " Hier ist sie sagte : " der Mann und Lea sagte : " Vielen Dank Herr Mann sagte : " Lea und ging zurück nach Hause .

Um das Korpus mit direkter Rede zu annotieren, müssten demnach alle Texte manuell bearbeitet werden. Hierbei müsste die direkte Rede identifiziert und entsprechend markiert werden. Dabei wären jedoch zunächst dieselben Probleme zu lösen, die bereits bei der Satzgrenzenerkennung auftreten, und darüber hinaus die häufig nicht triviale Bestimmung, ob es sich um direkte Rede handelt, wo diese beginnt und endet und wo möglicherweise Sprecherwechsel anzunehmen sind. Die Problematik der Annotation direkter Rede in Kindertexten wird an dem folgenden Beispiel sehr deutlich (dargestellt ist die orthographische Zielhypothese):

(35) Lea am Sonntag hat Jan Lea und Lars besucht und haben schön gespielt ein sehr schöner Spiel sollen wir das spielen ja oder nein ja sag Jan und auch Lars sag ich will auch dieser Spiel spielen okay nun setzen wir uns hin und spielen dieses Spiel oh ja mein bester Spiel dieses Spiel mag ich gern das spiele ich immer zu Hause mit meine Oma mein Oma mag dieses Spiel so jetzt spielen wir aber ist mein Lieblingsspiel schreit Jan Sie setzen sich hin und spielen eine nach den andere immer schön langsam fast hat der Turm um runtergefallen Uff Glück gehabt dass das Turm nicht

> runtergefallen ist so fast haben wir das ganze Turm fertig Lars pass auf ganz vorsichtig du sollst sich lieber den Turm fest genauso langsam vorsichtig ganz vorsichtig genau so festhalten vorsichtig es ist bald fertig es fällt langsam uuii Uff das ist nicht runtergefallen Dodo nicht überall ?schnürch ?schnürch ?schnürch der kleiner Dodo ich glaube dass Dodo etwas gerochen hat sagt Lars und und legt den Holzstein ganz Vorsicht ganz Vorsicht genau so ganz Vorsicht fertig ist das Turm ich kann nicht mehr

4.7.4 Grundschulwiki

Da es sich bei den Texten des Grundschulwikis um Sachtexte handelt, ist keine direkte Rede anzunehmen. Stattdessen verwenden die Autoren Anführungszeichen im Grundschulwiki vor allem zur Hervorhebung bestimmter Textstellen, wie an den folgenden Beispielen zu erkennen:

(36) Achkarren ist ein "musikalisches Dorf"

(37) Sein Lieblingshobby ist Schauspielen und er spielt in "Star Trek"

Auch bei Vorkommen der direkten Rede wäre es nur schwer möglich, diese zu annotieren, da das Grundschulwiki nicht über eine grammatische Zielhypothese mit korrigierter Interpunktion verfügt und eine zuverlässige automatische Erkennung der direkten Rede somit nicht gewährleistet werden kann.

4.7.5 Klexikon

Auch die Texte des Klexikons sind den Sachtexten zuzuordnen, sodass die direkte Rede in der Regel nicht vorkommen sollte. Die Autoren setzen Anführungszeichen indes zur Hervorhebung von Textpassagen ein. Folgende Beispiele zeigen Einsatzmöglichkeiten der Anführungszeichen:

(38) Wenn sich die Herzmuskeln zusammenziehen und wieder entspannen, sagt man dazu: "Das Herz schlägt".

(39) Es gibt viele Redewendungen, in denen das Herz vorkommt, zum Beispiel nennen manche Erwachsene ihr Kind "Herzchen" und sagen sich gegenseitig "Ich liebe dich von Herzen". Man lacht auch "herzlich" und sagt, das Leben ist voller "Herz und Schmerz". Es gibt Schnuller in Herzform und Schokoladenherzen auf der Kirmes.

(40) Das wird "Karies" genannt.

Werden Ausdrücke wie "sagen sich gegenseitig ‘Ich liebe dich von Herzen‘" als direkte Rede angesehen, so ist es dennoch nicht möglich, diese im Klexikon automatisch zu annotieren. Zwar liegt eine grammatische Zielhypothese vor, die automatische Unterscheidung der Verwendungszwecke der Anführungszeichen ist jedoch nicht ohne Weiteres möglich, daher wurde auf die Annotation verzichtet.

4.8 Phoneme, Grapheme, Silben, Morpheme

Zusätzlich zu den üblichen Annotationen in standardsprachlichen Korpora wurden die Kinderkorpora in diesem Projekt mit weiteren linguistischen Informationen angereichert, die insbesondere für die Fehleranalyse in Kindertexten nützlich sein können. Die folgende Auflistung nach Laarmann-Quante, Knichel, Dipper & Betken (2016) gibt zunächst einen Überblick über die Annotationen:

Phoneme: Jedes Zeichen bzw. jede Zeichenfolge der orthographischen Zielhypothese wird auf ein Phonem (SAMPA) abgebildet. Für das Wort <fällt> wäre das Ergebnis bspw. `f E l t`. Die Alignierung der Phoneme mit den Buchstaben des Zielwortes wird mit Levenshtein-basierten Skripten von Marcel Bollmann[12] realisiert.

Grapheme: Jedes Zeichen bzw. jede Zeichenfolge der orthographischen Zielhypothese wird auf ein Graphem nach Eisenbergs (2006) Graphem-Definition abgebildet. Letztlich bedeutet dies, dass jedes Zeichen auf sich selbst abgebildet wird, abgesehen von <ie>, <ch>, <sch> und <qu>, die als Mehrgrapheme zählen.

Silben: Jede Silbe des Zielwortes wird als *stress* (kurz für *stressed*), *unstress* (*unstressed*) oder *red* (*reduced*) annotiert.

Morpheme: Alle Morpheme in den Zielwörtern werden mit ihrem Morphemtyp annotiert. Gebundene Morpheme werden dabei unterschieden in Derivations- und Flexionsmorpheme, freie Morpheme nach ihrer Wortart. Die verwendeten Tags entsprechen einer Teilmenge der von BAS[13] aufgelisteten allgemeinen Morphemklassen.

Fremdwort: Für jedes Zielwort wird annotiert, ob es sich nach silbenstrukturellen Gesichtspunkten um ein Wort aus dem deutschen Kernwortschatz handelt. Hierzu zählen alle monosyllabischen und disyllabischen Stämme mit einem trochäischen Betonungsmuster (betonte Silbe + Reduktionssilbe) sowie Flexionen und Derivationen solcher Stämme. Wörter, die keine solche Struktur aufweisen, werden als Fremdwörter gezählt.

Existenz der Originalschreibung: Für jeden Fehler wird annotiert, ob die fehlerhafte Schreibung des Kindes (durch Verwechslung oder Zufall) eine existierende Wortform ergibt, d. h. ein Real-Word-Error ist wie z. B. *<feld> für <fällt>. Groß- und Kleinschreibung wird hierbei ignoriert. Als Grundlage für die Beurteilung, ob ein Wort existiert, wird ein

[12] https://github.com/mbollmann/levenshtein
[13] http://www.phonetik.uni-muenchen.de/Bas/readme_mrp_inventory.txt

Auszug aus dem Kinderbuchkorpus childLex (Schroeder, Würzner, Heister, Geyken & Kliegl, 2015) verwendet.

Plausibilität der Originalschreibung: Für jede Silbe wird annotiert, ob es sich um eine graphotaktisch mögliche Silbe im Deutschen handelt (*true*/*false*). Nicht plausibel wäre bspw. *<Bllätter>, da im Onset keine Doppelkonsonanten auftreten können. Es wird außerdem festgehalten, wenn die Silbenstruktur des Wortes verändert wurde und dadurch eine Silbe fehlt (*miss*) oder hinzugefügt wurde (*sup*).

Die Annotation der linguistischen Informationen wurde mit dem filetoxml-Skript (Stand 18.01.2018) aus dem Litkey-Projekt durchgeführt. Als Input benötigt das Skript die Transkription sowie die orthographische Zielhypothese und erzeugt dann zu jedem Tokenpaar die beschriebenen Annotationen. Die Informationen bzgl. Phonemen, Graphemen, Silben und Morphemen werden hierbei mithilfe des Webservices G2P[14] vom Bavarian Archive of Speech Signals (BAS; Reichel, 2012; Reichel & Kisler, 2014) und einigen heuristischen Abbildungen bestimmt. Zur Verbesserung der ansonsten wortbasierten Morphemannotation wird zudem der auch in diesem Projekt eingesetzte Stanford-Tagger mit dem beschriebenen Kinder-Modell verwendet, um ein-morphemige Wörter sowie zwei-morphemige Wörter mit einem freien und einem Flexionsmorphem (Ausnahme: nominalisierte Verben) unter Berücksichtigung des Textkontexts zu taggen. Die STTS-Tags werden anschließend auf Morphem-Tags abgebildet. Das Ergebnis der Annotation wird im LearnerXML-Format ausgegeben und kann anschließend u. a. bei der Fehlerannotation (s. Abschnitt 4.9) weiterverarbeitet werden.

Eine Evaluation der Annotationen des BAS-Webservices auf 20 Texten aus dem Litkey-Korpus im Litkey-Projekt hat folgende Ergebnisse ergeben (in Klammern ist die absolute Anzahl der Fälle angegeben):

	Korrekt	Falsch	Fehlend	Überflüssig
Phoneme	96,19% (6435)	2,44% (163)	1,38% (92)	0,06% (4)
Silben	91,84% (2184)	8,07% (192)	0,08% (2)	0,13% (3)
Morpheme	82,88% (1888)	13,56% (309)	3,56% (81)	0,97% (22)

Tabelle 9: Evaluation der BAS-Annotationen

Lediglich ein Wort in den 20 evaluierten Texten enthielt einen Fehler in der Graphem-Annotation. Die Zahlen können einen ungefähren Eindruck davon vermitteln, mit welcher Annotationsqualität in den einzelnen Subkorpora gerechnet werden kann. Allerdings wurde im Litkey-Projekt im Anschluss an die Evaluation eine manuelle Korrektur der Silben- und Phonem-Annotation (fast) aller im Litkey-Korpus

[14] https://webapp.phonetik.uni-muenchen.de/BASWebServices/#/services/Grapheme2Phoneme

vorkommenden Wörter vorgenommen, die in dem von filetoxml.py verwendeten Dictionary gespeichert ist. Da dieses Dictionary die Grundlage aller weiteren Annotationen darstellt, sollte die Annotation der Silben und Phoneme der entsprechenden Wörter nun in allen Subkorpora korrekt sein und die Accuracy auf der Silben- und Phonem-Ebene auch außerhalb des Litkey-Korpus höher liegen als hier angegeben.

Die folgenden Abschnitte geben einen kurzen Überblick darüber, wie die einzelnen Subkorpora mit den zusätzlichen linguistischen Informationen annotiert wurden und welche Besonderheiten dabei ggf. vorlagen.

4.8.1 Osnabrücker Bildergeschichtenkorpus

Die Annotation der linguistischen Informationen Phoneme, Grapheme, etc. wurde wie bereits beschrieben automatisch mit dem filetoxml-Skript durchgeführt. Hierbei konnte auf die korrigierten Phonem- und Silbenannotationen aus dem Litkey-Korpus zurückgegriffen werden.

Zudem wurden für das Osnabrücker Bildergeschichtenkorpus circa. 680 neue Wortformen vom BAS-Webservice abgerufen. Hauptsächlich handelte es sich dabei um Namen wie „Jakob“, „Timmy“, „Balduin“, „Fifi“, usw., aber auch viele korpusspezifische Wörter, die eng mit den Bildergeschichten in Verbindung stehen wie z. B. „Wintergeschichte“, „Schuhputzerdieb“ oder „Entenfamilie“.

Bei insgesamt 2.713 verschiedenen orthographischen Zielwort-Types (inkl. Groß-/Kleinschreibung) bedeutet dies, dass etwa ein Viertel des Vokabulars aus dem Osnabrücker Bildergeschichtenkorpus nicht im Litkey-Korpus oder dem von filetoxml.py verwendeten childLex-Auszug enthalten ist.

4.8.2 H1 Children’s Writing Korpus

Auch für das H1 Children’s Writing Korpus wurde die Annotation der Phoneme, Grapheme, etc. wie beschrieben automatisch mit dem filetoxml-Skript durchgeführt und auf die korrigierten Phonem- und Silbenannotationen aus dem Litkey-Korpus zurückgegriffen.

Die Annotation wurde im Anschluss an die Verarbeitung des Osnabrücker Bildergeschichtenkorpus durchgeführt, wobei noch ca. 1.380 neue Wortformen vom BAS-Webservice abgerufen werden mussten. Auch hier handelte es sich dabei häufig um Namen wie „Emma“, „Timo“, „Teo“, „Antonia“, usw. sowie viele korpusspezifische Wörter, die mit den beschriebenen Bildern in Verbindung stehen. Dazu zählen z. B. „Seerobbenschule“, „Ziegenbockteddy“, „Giraffenkuscheltier“, „Heißluftballon“, „Zauberland“ oder „Toilettendeckel“. Darüber hinaus mussten viele Zielhypothesen, die Tippfehler enthalten, abgefragt werden.

Insgesamt sind von den 5.083 verschiedenen orthographischen Zielwort-Types (inkl. Groß-/Kleinschreibung) aus dem H1 Children’s Writing Korpus etwa 27% nicht im Litkey-Korpus, Osnabrücker Bildergeschichtenkorpus oder dem von filetoxml.py verwendeten childLex-Auszug enthalten.

4.8.3 Litkey-Korpus

Wie bei den anderen Korpora erfolgte die Annotation der Phoneme, Silben, usw. automatisch mit filetoxml.py. Wie bereits beschrieben, war die kontextunabhängige Phonem- und Silbenannotation für fast alle Wörter schon manuell korrigiert und im Dictionary gespeichert. Nur knapp 40 Wörter mussten online vom BAS-Webservice abgerufen werden, darunter v. a. Interpunktionszeichencluster und großgeschriebene Formen zu bereits bekannten Wörtern. Da der Service zwischenzeitlich aus unbekannten Gründen nicht erreichbar war, wurden jedoch einige der 40 Wörter ebenfalls manuell annotiert.

4.8.4 Grundschulwiki

Wie bereits beschrieben erfolgte die Annotation der linguistischen Informationen Phoneme, Grapheme, etc. automatisch mit dem filetoxml-Skript. Da das Grundschulwiki zuletzt annotiert wurde, konnten die bereits abgerufenen Wortformen des Klexikons miteinbezogen werden. Dennoch wurden ca. 1.910 neue Wortformen abgefragt. Dabei handelt es sich vor allem um spezifische Begriffe der einzelnen Themengebiete oder Eigennamen der beschriebenen Serien, Städte, Charaktere und Bücher wie z. B. „Attribut“, „Veith“, „Altenmünster“, „Kleintierzuchtverein“, „Bischofsstadt“, „Aquarena“, „Ortsteil“, „Täufer“, „Kickboxen“, „Liedermacherin“, „Schwager“ und „Apollo“. Da insgesamt 1.910 der 5.820 Types abgefragt werden mussten, bedeutet dies, dass ungefähr 33% des Vokabulars des Korpus unbekannt waren.

4.8.5 Klexikon

Ebenso wie bereits beschrieben wurde zur automatischen Annotation der linguistischen Informationen Phoneme, Grapheme etc. im Klexikon das filetoxml-Skript verwendet. Bei der Annotation mussten dabei ungefähr 11.800 Wortformen beim BAS-Webservice abgefragt werden. Dies entspricht bei 21.695 Types ca. 54% des Korpus. Betroffen sind dabei ebenso wie beim Grundschulwiki viele themenspezifische Worte und Eigennamen von Entitäten der Geschichte wie „Damaskus“, „al-Assad“, „Zigeuner“, „Green“, „Pistols“, „Day“, „Kartoffelsorten“, „Politikwissenschaftler“, „Sprungschanze“, „Türkisch“ und „Ostermontag“, aber auch bisher unbekannten Verben oder Verbformen wie „salben“, „verkündigt“, „exportiert“, „verwaltet“, „warben“, „umsegelte“ und „hineingelangen“.

4.9 Rechtschreibfehler

Aufbauend auf der Annotation der Korpora mit Phonemen, Graphemen, Silben und Morphemen wurden die Texte in einem letzten Schritt mit orthographischen Fehlern nach Laarmann-Quante et al. (2016) annotiert. Das Annotationsschema ist eng an der graphematischen Theorie von Eisenberg (2006) orientiert und unterscheidet sich von etablierten orthographischen Analyseschemen v. a. durch seinen mehrschichtigen Aufbau und die feingliedrigen Fehlerkategorien. Dadurch erlaubt es eine sehr genaue Analyse der Rechtschreibfehler in einem Text und kann für die Erstellung detaillierter

Fehlerprofile einzelner Schüler und zur Untersuchung der Entwicklung ihrer orthographischen Kompetenz genutzt werden. Darüber hinaus lässt es sich auch nutzen, um allgemein die orthographischen Eigenschaften deutscher Wörter zu bestimmen.

Das Annotationsschema ist auf die Annotation von Rechtschreibfehlern in Einzelwörtern fokussiert, es deckt allerdings auch einige Phänomene auf textueller Ebene ab. Unterschieden werden Fehler aus fünf Oberkategorien: Phonem-Graphem-Korrespondenzen (PG), Silben (SL), Morphologie (MO), Syntax (SN) und Silbentrennung (HY). Da sich das Schema derzeit noch in der Entwicklung befindet, kann die Anzahl der Fehlerkategorien leicht schwanken. In jedem Fall ist das Annotationssystem so entworfen, dass jedem Fehler nur genau ein Fehler-Tag zugeordnet ist, d. h. mehrere mögliche Fehlerkategorien für einen Fehler sind nicht vorgesehen.

Neben der Annotation der Fehlerkategorie werden für jeden Fehler der Ebenen PG, SL und MO zwei weitere Eigenschaften codiert.

Phonologische Plausibilität (phon_orig_ok):

Dieses Feature beurteilt, ob der Fehler die Aussprache des Wortes beeinflusst.

- *True* bedeutet, die Aussprache der Schreibung des Kindes ist ähnlich oder gleich der Standardaussprache (z. B. *<ier> statt <ihr>).
- *coll* trifft zu, wenn die Schreibung einer umgangssprachlichen oder dialektalen Aussprache entspricht (z. B. *<schümft> für <schimpft>).
- *false* bedeutet, dass der Fehler die Aussprache deutlich verändert (z. B. *<Schle> für <Schule>).

Morphemkonstanz (morph_const):

Das Feature beurteilt, ob Wissen über Morphemkonstanz nötig ist, um das Wort korrekt schreiben zu können. Die folgende Erläuterung der möglichen Werte erfolgt am Beispiel von Konsonanten-verdopplungen.

- *neces*: Gewisse Konsonantenverdopplungen wie in <kommst> lassen sich nur mit Vererbung via Morphemkonstanz (<kommen>) erklären, d. h. Wissen über Morphemkonstanz ist nötig, um zur korrekten Schreibung zu gelangen.
- *redun*: Der Ursprung des Doppelkonsonanten in <kommend> lässt sich sowohl mit Morphem-konstanz als auch mit silbenstrukturellen Eigenschaften erklären, d. h. Wissen über Morphem-konstanz ist nützlich aber redundant.
- *na*: Im Fall von *<dan> statt <dann> hilft Morphemkonstanz nicht bei der Bestimmung der korrekten Schreibung, da der Doppelkonsonant sich nicht mit einer verwandten Wortform erklären lässt. Wissen über Morphemkonstanz ist deshalb nicht anwendbar.
- *hyp*: Fälle, in denen die Morphemkonstanz verletzt ist wie bei <Bus/Busse>, werden mit *hyp* markiert.

Für die Annotation der Fehler in diesem Projekt wurde das errors_baseRate-Skript (Stand 30.08.2017) aus dem Litkey-Projekt verwendet. Dieses baut auf den Annotationen von filetoxml.py (s. Abschnitt 4.8) auf und fügt die beschriebenen Informationen zu den erzeugten LearnerXML-Dateien hinzu. Zunächst werden dafür Transkription und orthographische Zielhypothese aligniert, um eindeutig festzuhalten, welche Buchstaben in der Schreibung des Kindes welchen Buchstaben des Zielwortes entsprechen. Fehler können dann exakt lokalisiert und den entsprechenden betroffenen Zeichen zugeordnet werden.

Da sich nicht nur das Annotationsschema, sondern auch das Skript noch in der Entwicklung befinden, traten bei der Annotation der Korpora teilweise Probleme auf, die eine rein automatische Annotation verhinderten. Die folgenden Abschnitte gehen für die einzelnen Subkorpora auf die spezifischen Schwierigkeiten ein und beschreiben, wie die Korpora in das finale LearnerXML-Format gebracht wurden.

4.9.1 Osnabrücker Bildergeschichtenkorpus

Die Annotation der Orthographie-Fehler im Osnabrücker Bildergeschichtenkorpus erfolgte wie bei den anderen Korpora automatisch mit errors_baseRate.py. Als Input dienten die von filetoxml.py produzierten LearnerXML-Dateien, in denen die Transkriptionen und orthographischen Zielwörter aligniert und mit Fehlern annotiert wurden.

Vier Texte mussten manuell mit Fehlern annotiert werden, da die automatische Annotation an folgenden Stellen abgestürzt ist:

- k014_s18: "Vusapfschtreifer Fußabstreifer"
- k020_s19: "veruchtek frühstückt"
- k029_s05: "bauhen bauen"
- k033_s06: "fer_sunden verschwunden"

Dabei wurde auch die für die Fehlerannotation vorausgesetzte Alignierung der Transkription mit den Zielwörtern manuell durchgeführt. Die Output-Dateien der Fehlerannotation wurden abschließend mit den weiteren in diesem Projekt vorhandenen Informationen (grammatische Zielhypothese, POS-Tags, Dependenzrelationen, etc. sowie Meta-Informationen s. Abschnitt 4.10) ergänzt, um das finale Zielformat zu erhalten.

4.9.2 H1 Children's Writing

Die Fehlerannotation im H1 Children's Writing Korpus erfolgte wie bei den anderen Korpora ebenfalls automatisch mit errors_baseRate.py. Als Input dienten die von filetoxml.py produzierten LearnerXML-Dateien, in denen die Transkriptionen und orthographischen Zielwörter aligniert und mit Fehlern annotiert wurden.

Sechs Texte mussten manuell mit Fehlern annotiert werden, da die automatische Annotation an folgenden Stellen abgestürzt ist oder wegen zu hohen Arbeitsspeicherverbrauchs (> 16 GB) nicht durchgeführt werden konnte:

- H1.KA.G2.2.week5.csv "Geburtztag_Parti Geburtstagsparty"
- H1.KB.G2.5.week5.csv "seh_Hunde Seehund"
- H1.KA.G2.22.longterm.csv "Tehos Theos"
- H1.KA.G3.12.week5.csv "Geburtstags_Party Geburtstagsparty"
- H1.KB.G2.12.week4.csv "sehepfert Seepferd"
- H1.KB.G3.7.week4.csv "Gegengesetzen entgegengesetzten"

Hierbei wurde auch die für die Fehlerannotation nötige Alignierung der Transkription mit der Zielhypothese manuell vorgenommen. Die Output-Dateien der Fehlerannotation wurden anschließend mit den weiteren in diesem Projekt vorhandenen Informationen (grammatische Zielhypothese, POS-Tags, Dependenzrelationen, etc. sowie Meta-Informationen s. Abschnitt 4.10) ergänzt, um das finale Zielformat zu erhalten.

4.9.3 Litkey-Korpus

Auch für das Litkey-Korpus erfolgte die Fehlerannotation automatisch mit errors_baserate.py. Input für das Programm waren die von filetoxml.py produzierten LearnerXML-Dateien, in denen die Transkriptionen und orthographischen Zielwörter aligniert und mit Fehlern annotiert wurden.

17 Texte mussten manuell mit Fehlern annotiert werden, da die automatische Annotation an folgenden Stellen abgestürzt ist oder aus mangelndem Arbeitsspeicher nicht durchgeführt werden konnte:

- 029-201112-II-Fundbuero_Katrin.csv "Fundbüro_geschäft Fundbürogeschäft"
- 058-201011-I-Frosch_Katrin.csv "spaziren-^gegangen spazierengegangen"
- 102-201112-I-Schule_Katrin.csv "gegenüber-^liegenden gegenüberliegenden"
- 181-201112-IV-Weg_Katrin.csv "freuhen freuen"
- 210-201112-IV-Weg_Doreen.csv "wra war"
- 219-201011-III-Seilbahn_Maurice.csv "gegenüber_liegenden gegenüberliegenden"
- 227-200910-I-Eis_Maurice.csv "sparihen spazieren"
- 270-201011-II-Jenga_Doreen.csv "bauhen bauen"
- 270-201011-IV-Weg_Doreen.csv "vor_vor_vor_vor_vor_gestern vorvorvorvorvorgestern"
- 279-201011-II-Jenga_Doreen.csv "Bauhen bauen"
- 284-201112-IV-Weg_Doreen.csv "Vermissten-^plakate Vermisstenplakate"
- 310-201011-III-Staubsauger_Doreen.csv "auslehen ausleeren"

- 328-201112-III-Seilbahn_Doreen.csv "bauhen bauen"
- 371-201112-III-Seilbahn_Doreen.csv "lieblingsteedybären Lieblingsteddybären"
- 458-201112-I-Schule_Katrin.csv "mathematick_auf_gaben Mathematikaufgaben"
- 489-201112-II-Fundbuero_Anna.csv "Nasenferschtopfung Nasenverstopfung"
- 497-201112-III-Seilbahn_Katrin.csv "Nachtbers_^venstebank Nachbarfensterbank"

Hierbei wurde auch die mit der Fehlerannotation verbundene Alignierung der Buchstaben manuell durchgeführt. Die Output-Dateien der Fehlerannotation wurden anschließend mit den weiteren in diesem Projekt vorhandenen Informationen (POS-Tags, Dependenzrelationen, etc. sowie Meta-Informationen s. Abschnitt 4.10) ergänzt, um das finale Zielformat zu erhalten.

4.9.4 Grundschulwiki

Auch die Annotation der Orthographie-Fehler im Grundschulwiki wurde wie bei den anderen Korpora automatisch mit errors_baseRate.py durchgeführt. Als Input dienten die von filetoxml.py produzierten LearnerXML-Dateien, in denen die Transkriptionen und orthographischen Zielwörter aligniert und mit Fehlern annotiert wurden. Die Annotation funktionierte prinzipiell problemlos, sodass keine Fehler manuell annotiert werden mussten.

Lange Links konnte das Programm jedoch nicht verarbeiten, diese wurden nachträglich ergänzt:

- http://www.milkmoon.de/magazin/PippiLangstrumpfLied
- http://www.alv.se/index-ger.asp
- http://www.de/infonetz/erfindungen/pippilangstrumpf/-/id16014/nid16014/did34096/4w1vx8/index.html

Die Output-Dateien der Fehlerannotation wurden anschließend mit den weiteren in diesem Projekt vorhandenen Informationen (POS-Tags, Dependenzrelationen, etc. sowie Meta-Informationen s. Abschnitt 4.10) ergänzt, um das finale Zielformat zu erhalten.

4.9.5 Klexikon

Wie bei allen anderen Korpora wurde die Annotation der Orthographie-Fehler im Klexikon automatisch mit errors_baseRate.py durchgeführt. Als Input dienten die von filetoxml.py produzierten LearnerXML-Dateien, in denen die Transkriptionen und orthographischen Zielwörter aligniert und mit Fehlern annotiert wurden. An dieser Stelle waren jedoch keine Fehler anzunehmen, da es sich um Texte von geschulten Autoren für Kinder handelt. Diese Annahme bestätigte auch die Annotation der Fehler. Hier traten keine Probleme auf.

Die Output-Dateien der Fehlerannotation wurden anschließend mit den weiteren in diesem Projekt vorhandenen Informationen (grammatische Zielhypothese, POS-Tags, Dependenzrelationen, etc. sowie Meta-Informationen s. Abschnitt 4.10) ergänzt, um das finale Zielformat zu erhalten.

4.10 Metadaten

Bei der Datenakquise bzw. dem Auslesen der Transkription wurden den Subkorpora ebenfalls Meta-Informationen entnommen. Während diese in einigen Korpora einzeln für jedes Kind, jede Klasse und jede Schule vorlagen, mussten einige Informationen für andere Korpora entweder aus einer Metadatei oder aus dem Text bzw. aus der ursprünglichen XML-Datei extrahiert werden.

Angepasst an die verfügbaren Informationen wurden diese anschließend individuell für jeden Text in Form eines Meta-Headers in die LearnerXML-Datei eingefügt. Den Korpora gemeinsam sind dabei die folgenden Informationen:

- topic
- date

Während das Klexikon und das Grundschulwiki keine nützlichen Informationen zum Autor enthalten, bestehen die Header des Osnabrücker Bildergeschichtenkorpus, des H1 Children's Writing Korpus und des Litkey-Korpus aus zwei verschiedenen Informationskategorien: Sie bieten sowohl Informationen über den Text als auch Informationen über den jeweiligen Autor.

In den folgenden Abschnitten werden die individuellen Meta-Informationen beschrieben. Insbesondere werden die Extraktion der Daten, das entstandene Dateiformat zur Zwischenspeicherung der entsprechenden Informationen und die nicht einbezogenen aber verfügbaren Informationen thematisiert.

4.10.1 Osnabrücker Bildergeschichtenkorpus

Für jede der 36 Klassen aus dem Osnabrücker Bildergeschichtenkorpus wurde eine Textdatei *kXXX_meta.txt* angelegt, in der alle Meta-Informationen zu dieser Klasse gespeichert sind.

In der XML-Datei befindet sich die Meta-Information im Wurzelknoten <klasse> und dessen Tochterknoten <info>. <klasse> enthält die Klassen-ID, den Ort, die PLZ und die Klassenstufe. <info> kann das Datum und Angaben zu Hilfestellungen, Erläuterungen, nicht-deutschen Kindern und Besonderheiten enthalten. Diese Informationen wurden extrahiert und im Format Attribut\tWert in die Textdateien übernommen. Die Datumsangaben wurden manuell in ein einheitliches Format Tag(e)/Monat(e)/Jahr(e) überführt und dabei ggf. vervollständigt. Anschließend wurde manuell zu jedem Ort das Bundesland ermittelt und in den Dateien ergänzt.

Von den Meta-Informationen über die Klasse wurden folgende Informationen, wenn vorhanden, in die LearnerXML-Dateien übernommen:

- Datum
- Bundesland
- Klassenstufe
- Klassen-ID

Für jedes Kind jeder Klasse wurde zudem eine Textdatei *kXXX_sXX_meta.txt* angelegt, in der die Meta-Informationen zu diesem Kind gespeichert sind. In der XML-Datei befinden sich die Meta-Informationen als Attribute in den <person>-Knoten. Folgende Informationen können darin enthalten sein:

- ID
- Geschlecht
- Herkunft
- Muttersprache (codiert als Zahl)
- Förderung
- Anmerkungen

Der <person>-Knoten kann zudem einen Tochterknoten <anmerkungen> haben, in dem Anmerkungen von (vermutlich) Tobias Thelen enthalten sind. Die Informationen wurden extrahiert und im Format Attribut\tWert in die Textdateien übertragen. In die LearnerXML-Dateien wurde davon, wenn vorhanden, übernommen:

- ID (*sXX*)
- Geschlecht (*male, female, unknown*)
- Herkunft
- Muttersprache ("*German*", "*German (weak dialect)*", "*German (strong dialect)*", "*Bilingual (German and language of origin)*", "*Language of origin*", "*Language of origin (but good knowledge of German)*")
- Förderung ("*no additional tutoring*", "*German as second language*", "*Reading, orthography*", "*Speech therapy*", "*Within internal differentiation*", Freitext)
- Anmerkungen

Zu jedem Text existiert zudem die Meta-Information, auf welcher Grundlage (Bildergeschichte oder Erlebnisbericht) der Text basiert. Die Information ist in den XML-Dateien im <text>-Element als Wert des Attributs *bezug* enthalten. Sie wurde in die LearnerXML-Dateien übernommen und zusätzlich an den Dateinamen angehängt als _bg01 bis _bg04 oder _erlebnisbericht. Bildergeschichte 1 und 4 scheinen hierbei dieselbe Geschichte zu sein.

4.10.2 H1 Children's Writing

Aus der mit dem H1 Children's Writing Korpus veröffentlichten Metadaten-Tabelle wurden für jedes Kind folgende Informationen in eine csv-Datei übernommen:

- ID (z. B. H1.KA.G2.2)
- Klassenstufe (2, 3)
- Geschlecht (m, w, u)
- Alter (8-11)
- Muttersprache
- Zweitsprache
- Drittsprache
- Mehrsprachig (True oder leer)
- Einsprachig (True oder leer)
- Jahre in Deutschland

In die LearnerXML-Dateien wurde hiervon übernommen:

- ID
- Klassenstufe
- Geschlecht
- Alter
- Sprachen (in der Reihenfolge Muttersprache, Zweitsprache, Drittsprache)

Aus dem Dateinamen wurde zudem der Testzeitpunkt abgeleitet und in den Header der XML-Datei eingefügt, wobei folgende Abbildung verwendet wurde:

- "week3" : "t1"
- "week4" : "t2"
- "week5" : "t3"
- "week6" : "t4"
- "week7" : "t5"
- "week8" : "t6"
- "week9" : "t7"
- "week10" : "t8"
- "week11" : "t9"
- "longterm" : "t10"

Das Datum ist einheitlich 2015, das Bundesland Baden-Württemberg. Der letzte Teil des Dateinamens (d. h. "week3" bis "longterm") wurde als Thema bzw. Bezug des Textes übernommen.

4.10.3 Litkey-Korpus

Aus den zahlreichen zum Litkey-Korpus vorhandenen Metadaten wurden für jedes Kind folgende Informationen in eine csv-Datei und von dort in die LearnerXML-Dateien übernommen:

- ID (001 bis 693)
- Schule (1 bis 7)
- Klasse (GT.21 bis GT.24, GT.31 bis GT.34, VglmF1 bis VglmF4, VgloF11, VgloF12 und VgloF2)
- Geschlecht (m, w, u)
- Sprachen
- Förderung
- Alter je Testzeitpunkt

In einer weiteren Datei ist zu jedem Text die Klassenstufe gespeichert, in der der Text verfasst wurde. Diese Information wurde ebenfalls in die LearnerXML-Datei übernommen. Aus dem Dateinamen wurde der Testzeitpunkt abgeleitet und in den Header der LearnerXML-Datei eingefügt, wobei folgende Abbildung verwendet wurde:

- "200910-I" : "February/March 2010"
- "200910-II" : "June/July 2010"
- "201011-I" : "September/October 2010"
- "201011-II" : "November/December 2010"
- "201011-III" : "February/March 2011"
- "201011-IV" : "June/July 2011"
- "201112-I" : "September/October 2011"
- "201112-II" : "November/December 2011"
- "201112-III" : "February/March 2012"
- "201112-IV" : "June/July 2012"

Das Bundesland ist Nordrhein-Westfalen. Der im Dateinamen enthaltene Name der Bildergeschichte (Eis, Weg, Staubsauger, Fundbuero, Rucksack, Frosch, Seilbahn, Schule) ist das Thema des Textes.

4.10.4 Grundschulwiki

Jeder Text der exportierten XML-Datei verfügt über zugehörige Meta-Daten. Diese wurden während der Textverarbeitung akquiriert und, wenn nötig, entsprechend abgespeichert. Bei bisher nicht weiter verwendeten Daten wurde die Möglichkeit der Speicherung implementiert, jedoch nicht genutzt, sodass diese nicht in der Datei "meta" der Wikis abgespeichert wurden. Folgende Informationen lagen vor:

- Titel
- ID
- Revision-ID
- Parent-ID
- Datum
- Autor
    - Name
    - ID
- Format
- Bilddateien
- Kategorien
- Referenzen
    - Erscheinung im Text
    - Referenz zu
    - Index Referenz
- Verbesserungskommentar

In die Meta-Datei des Korpus übernommen und anschließend in die LearnerXML-Dateien eingefügt wurden:

- Titel
- ID
- Revision-ID
- Parent-ID
- Datum
- Kategorien
- Referenzen
    - Erscheinung im Text
    - Referenz zu
    - Index Referenz

Es wurde sich an dieser Stelle bewusst gegen die Verwendung der Autorinformationen entschieden, da diese aus frei wählbaren Nutzernamen und einer festgelegten ID bestehen. Ein Rückbezug zu den tatsächlichen Daten des Autors hinsichtlich Alter oder Wohnort ist nicht möglich. Format und Verbesserungskommentar wurden nicht genutzt, da hier eher eine interne Nutzung vorliegt und keine Aussagen über den Text selbst getroffen werden.

Die folgende Datei zeigt einen Ausschnitt der erstellten Meta-Datei, die bei der Erzeugung des Meta-Headers ausgelesen wird. In geschweiften Klammern befinden sich die Referenzen, in den eckigen Klammern die Kategorien:

```
Aschenputtel.txt        2055    6916    5240    2007-07-22T07:40:32Z    {}      []
Asterix und Obelix.txt  3718    14780   14776   2011-04-29T17:59:28Z    {}      []
Atwood, Roman.txt       5028    21191   21085   2017-04-15T09:38:47Z    {}      []
Audi.txt        2735    17921   17917   2014-03-14T11:28:23Z    {}      []
Australien.txt  3818    19512   15294   2016-01-27T11:32:44Z    {'Ozeanien':
('Ozeanien', 'tok6'), 'Wellensittich': ('Wellensittich', 'tok126'), 'Schnabeltier':
('Schnabeltier', 'tok129'), 'Kakadu': ('Kakadu', 'tok132'), 'Papagei': ('Papagei',
'tok135'), 'Bartagame': ('Bartagame', 'tok138'), 'Beutelmull': ('Beutelmull',
'tok141')}      []
Bad Godesberg.txt       5044    21185   21142   2017-04-15T09:31:01Z    {'Bonn':
('Bonn', 'tok4')}       []
```

Abbildung 25: Ausschnitt der Meta-Datei des Grundschulwikis

4.10.5 Klexikon

Ebenso wie bei der Erzeugung des Meta-Headers des Grundschulwikis verfügt das Klexikon über Informationen, die nicht im Meta-Header berücksichtigt wurden. Die Möglichkeit der Speicherung ist hier jedoch ebenfalls gegeben. Folgende Informationen lagen vor:

- Titel
- ID
- Revision-ID
- Parent-ID
- Datum
- Autor
    - Name
    - ID
- Format
- Bilddateien
- Kategorien
- Referenzen
    - Erscheinung im Text
    - Referenz zu
    - Index Referenz
- Verbesserungskommentar

In die Meta-Datei des Korpus übernommen und anschließend in die LearnerXML-Dateien eingefügt wurden:

- Titel
- ID
- Revision-ID
- Parent-ID
- Datum
- Kategorien
- Referenzen
    - Erscheinung im Text
    - Referenz zu
    - Index Referenz

Zwar bestanden die Autoren-Informationen beim Klexikon aus dem tatsächlichen Namen des Autors und einer ID, aus Gründen der Anonymität und dem fehlenden Rückschluss auf Alter und Wohnort, wurden diese jedoch genauso wie die Informationen zum Format sowie Kommentare, die Klexikon-intern benötigt werden, nicht genutzt.

Die folgende Datei zeigt einen Ausschnitt der erstellten Meta-Datei, die bei der Erzeugung des Headers ausgelesen wird. In geschweiften Klammern befinden sich die Referenzen, in den eckigen Klammern die entsprechenden Kategorien:

```
Zahn.txt        637     51249   49322   2017-03-27T21:51:56Z    {'Menschen': ('Mensch', 'tok11'),
'Tiere': ('Tier', 'tok14'), 'Kindern': ('Kind', 'tok131'), 'Wurzeln': ('Wurzel', 'tok191'), 'Zahnarzt':
('Zahnarzt', 'tok216'), 'Knochen': ('Knochen', 'tok283'), 'Wort': ('Wort', 'tok308'), 'Lateinischen':
('Latein', 'tok313'), 'Krankheit': ('Krankheit', 'tok322'), 'Bakterien': ('Bakterien', 'tok329'),
'Zucker': ('Zucker', 'tok335'), 'Zahnpasta': ('Zahnpasta', 'tok359'), 'Wirbeltiere': ('Wirbeltiere',
'tok371'), 'Reptilien': ('Reptilien', 'tok383'), 'Krokodile': ('Krokodil', 'tok385'), 'Amphibien':
('Amphibien', 'tok389'), 'Frösche': ('Frosch', 'tok393'), 'Vögel': ('Vögel', 'tok399'), 'Nagetiere':
('Nagetier', 'tok448'), 'Elefanten': ('Elefant', 'tok453'), 'Kopf': ('Kopf', 'tok473'), 'Haie': ('Hai',
'tok486'), 'Hai': ('Hai', 'tok515')}    ['Klexikon-Artikel', 'Körper und Gesundheit', 'Essen und
Trinken']
Rathaus.txt     639     49787   36001   2017-03-06T20:10:25Z    {'Haus': ('Haus', 'tok5'), 'Politiker':
('Politiker', 'tok10'), 'Stadt': ('Stadt', 'tok14'), 'Gemeinde': ('Gemeinde', 'tok16'), 'Politik':
('Politik', 'tok18'), 'Parlament': ('Parlament', 'tok39'), 'Bürgermeister': ('Bürgermeister', 'tok53'),
'Europa': ('Europa', 'tok127'), 'Mittelalter': ('Mittelalter', 'tok133'), 'Hamburg': ('Hamburg',
'tok211'), 'Musik': ('Musik', 'tok242')}        ['Klexikon-Artikel', 'Politik und Gesellschaft']
App.txt 693     50977   44531   2017-03-22T22:48:29Z    {'Computer': ('Computer', 'tok5'), 'englischen':
('Englische Sprache', 'tok13'), 'Wortes': ('Wort', 'tok14'), 'übersetzt': ('Übersetzung', 'tok21'),
'Handys': ('Handy', 'tok52'), 'Wort': ('Wort', 'tok85'), 'deutsche Sprache': ('deutsche Sprache',
'tok95'), 'Android': ('Android', 'tok130'), 'Kinder': ('Kind', 'tok198'), 'Euro': ('Euro', 'tok220'),
'Namen': ('Name', 'tok275'), 'Geld': ('Geld', 'tok303'), 'Nachrichten': ('Nachrichten', 'tok320'),
'Filme': ('Film', 'tok329'), 'Deutschland': ('Deutschland', 'tok336')}  ['Klexikon-Artikel', 'Sprache und
Kultur', 'Wissenschaft und Technik']
```

Abbildung 26: Ausschnitt der Meta-Datei des Klexikons

5. Anleitung zur Korpuserweiterung

Um eine Erweiterung des Korpus zu ermöglichen, wurden die entwickelten Algorithmen an die automatische Bearbeitung weiterer Daten angepasst. Da es bisher jedoch nicht möglich ist, automatisch und ohne manuelle Korrektur einen bisher unbekannten Text zu parsen, beschränkt sich diese Version auf die Datenakquise, die Erstellung der Transkription und der orthographischen Zielhypothese sowie die Übernahme in LearnerXML inklusive POS-Tagging.

Das Programm gliedert sich dabei in drei bzw. vier Teile. Im ersten Teil werden die Textdateien aus der exportierten XML-Datei erstellt. Der zweite Teil fokussiert sich auf die Normalisierung des Textes via BoNKid. Final liest der dritte Teil des Programms die vorhandenen Informationen im csv-Format ein, erzeugt die Annotation der Phoneme, Silben etc. sowie die Fehlerannotation und führt alles abschließend im LearnerXML-Format zusammen. Zusätzlich wird an dieser Stelle der Meta-Header eingefügt. Optional ermöglicht es der vierte Programmteil, die XML-Dateien strukturiert formatiert auszugeben.

Die folgenden Unterkapitel thematisieren die einzelnen Programmteile und zeigen anhand von Beispielen, wie die Anwendung erfolgt und welches Input- bzw. Outputformat verwendet wird. Bei der Ausführung der einzelnen Schritte wird vorausgesetzt, dass das aktuelle Verzeichnis der Ordner mit dem jeweiligen Programm ist.

Datenakquise

Um die Daten einer Wikimedia-Anwendung zu akquirieren, muss die Funktion "Seiten exportieren" verwendet werden, die sich zumeist unter den "Spezialseiten" des Wikis befindet. Die folgende Abbildung zeigt die entsprechende Seite des Grundschulwikis:

Abbildung 27: Ausschnitt der Seite "Seiten exportieren" des Grundschulwikis

Ist es notwendig, alle Texte des Korpus zu exportieren, kann unter "Seiten aus folgendem Namensraum" der Namensraum "Seiten" ausgewählt werden. Nach einem anschließenden Klick auf "Hinzufügen" werden alle Seiten des Wikis in dem gezeigten Textfeld abgebildet. Ein weiterer Klick auf "Seiten exportieren" exportiert den Inhalt der Seiten und fordert den Nutzer auf, die erstellte Datei herunterzuladen. Anschließend sollte die Datei zur einfacheren Weiterverarbeitung durch den Nutzer umbenannt werden.

Tokenisierung des Ursprungsformats und Erstellung der Meta-Datei – Part I

Nach der Datenakquise ist es notwendig, die XML-Datei einzulesen, die Meta-Informationen zu extrahieren, den Text zu tokenisieren und die Textdateien zu erstellen. Um die Verarbeitung der XML-Datei zu starten, ist der Aufruf des Befehls

```
>> python WikiToTxt.py Name.xml NeuerName
```

in der Kommandozeile nötig. Anstelle des Ausdrucks "Name.xml" sollte der Name der ausgelesenen XML-Datei eingefügt werden, der Ausdruck "NeuerName" steht für den Namen des neuen Korpus (z. B. "Grundschulwiki"), der insbesondere bei der Erstellung der Meta-Datei benötigt wird.

Sollte auf die Eingabe in der Kommandozeile verzichtet werden, fragt das Programm automatisch den Namen der auszulesenden Datei mit "XML-File:" und den Namen des neuen Korpus mit "Wiki:" ab. Die Nutzung außerhalb der Kommandozeile ist daher auch möglich.

Wurde das Programm gestartet, so werden die Texte sukzessiv abgearbeitet und in dem Ordner "Textfiles" abgelegt. Dabei erscheinen die Dateinamen der bereits verarbeiteten Textdateien auf der Kommandozeile.

Zu Beginn der Verarbeitung wird das Korpus auf Meta-Informationen überprüft und anschließend bereinigt. Nicht mehr benötigte Meta-Ausdrücke wie z. B. Bildunterschriften, Kommentare, Referenzen wie "Hier geht es zur Hauptseite!" und weitere nicht dem Text zugehörige Informationen werden gelöscht. Sollten diese Informationen im späteren Verlauf des Projektes benötigt werden, ist die Abspeicherung ohne Probleme möglich, das Format und der Inhalt der Textdateien verändern sich dabei nicht. Die bereits gespeicherten Meta-Informationen werden anschließend in eine Datei geschrieben. Die Ausgabe ist dabei in folgendem Format:

Artemia salina.txt	2612	14310	7911	2010-12-05T12:47:49Z	{'Salzkrebschen': ('Salzkrebs', 'tok11'), 'Salinenkrebschen.': ('Salinenkrebs', 'tok13')}

Die Meta-Datei beinhaltet zuerst den Namen der entsprechenden Datei. Es folgen die ID des Textes, die Revision-ID, die Parent-ID und die Zeitdaten. Anschließend werden ausgelesene Referenzen und Kategorien abgebildet.

Das Programm beinhaltet eine optimierte Variante der Tokenisierung. Das zuvor für Klexikon und Grundschulwiki verwendete *tokenize*-Modul aus dem NLTK zeigte bei der Analyse der Daten keine

ausreichende Qualität für eine automatische Weiterverarbeitung. Token wie "5." und "St." wurden ebenso aufgetrennt wie Zahlen (z. B. "360.000"). Aus diesem Grund wird nun auf die Funktionen dieses Moduls verzichtet. Stattdessen trennt ein Algorithmus nun die Token an den Leerzeichen und überprüft anschließend, ob sich ein Satzzeichen in dem entsprechenden Token befindet. Ist dies der Fall, so wird anhand der vorhandenen Vokale und der Länge des Tokens entschieden, ob es sich um eine Abkürzung handeln könnte. Besteht die Annahme, dass es sich um eine Abkürzung handelt, wird der Punkt nicht abgetrennt. Analog wird dies bei Zahlen, auf die ein Punkt folgt, durchgeführt. Befindet sich ein Bindestrich innerhalb eines Wortes oder ein Doppelpunkt oder Komma zwischen zwei Zahlen, werden diese nicht weiter betrachtet. Alle weiteren Satzzeichen, die sich innerhalb oder am Ende eines Tokens befinden, werden abgetrennt.

Bei der Analyse der Daten zeigt diese Tokenisierung ein deutlich besseres Ergebnis als die zuvor verwendete Funktion *word_tokenize*. Beide Tools schaffen es jedoch nicht, Abkürzungen an einer Satzgrenze zu berücksichtigen. Während *sent_tokenize* hier i. d. R. eine Satzgrenze annimmt, wird bei der neuen Tokenisierung die Einstufung als Abkürzung bevorzugt. Ein Abgleich mit einem ausreichend großen Korpus (mit korrekter oder korrigierter Rechtschreibung) könnte dieses Problem in vielen Fällen vermeiden: Ist das folgende Token großgeschrieben, aber auch in kleingeschriebener Form im Korpus vorhanden, wird es sich eher um eine Satzgrenze handeln.

Die folgenden Beispiele zeigen links die Tokenisierung mit *word_tokenize* und *sent_tokenize* aus dem NLTK und rechts die optimierte Version der Tokenisierung:

Text "Adeliepinguin"	
Der	Der
Adeliepinguin	Adeliepinguin
heist	heist
so	so
weil	weil
der	der
Entdcker	Entdcker
Dumont	Dumont
d	d
'	'
Urille	Urille
ihn	ihn
1840	1840
nach	nach
seiner	seiner
Frau	Frau
benannte.Ein	benannte
	.
	Ein
Adeliepinguin	Adeliepinguin
in	in

der	der
Küste	Küste
der	der
Arktischen	Arktischen
Inseln.In	Inseln
	.
	In
einem	einem
Kieselstein	Kieselstein
Nest	Nest
geboren.Er	geboren
	.
	Er
Wiert	Wiert
ein	ein
mal	mal
am	am
Tag	Tag
gefüttert.Er	gefüttert
	.
	Er
wiegt	wiegt
100g	100g
mit	mit
schwarzem	schwarzem
Kopf	Kopf
.	.

Text "Achkarren"	
Die	Die
Rebfläche	Rebfläche
ist	ist
insgesamt	insgesamt
ca	ca.
.	
180	180
Hektar	Hektar
groß	groß
.	.

Nach der Durchführung der Tokenisierung werden die Texte jeweils in eine Textdatei eingefügt. Diese ist genauso wie der benötigte Input utf-8-codiert. Der Dateiname entspricht dem Titel des Wiki-Artikels. Die Textdateien enthalten schließlich ein Token pro Zeile und sind zur weiteren Verarbeitung bereit.

Normalisierung via BoNKid – Part II

Für die Bestimmung der orthographischen Zielhypothese wird das Tool BoNKid verwendet. Dieses nimmt als Input den tokenisierten Text mit einem Token pro Zeile. Als Datei-Format werden ".csv" und ".txt" akzeptiert. Zum Starten des Programms ist in der Kommandozeile

```
>> python normalize.py
```

einzugeben. Das Programm erfragt daraufhin zunächst die Pfade der Input- und Output-Dateien. Außerdem wird der Normalisierungsmodus abgefragt, wie im folgenden Screenshot zu sehen:

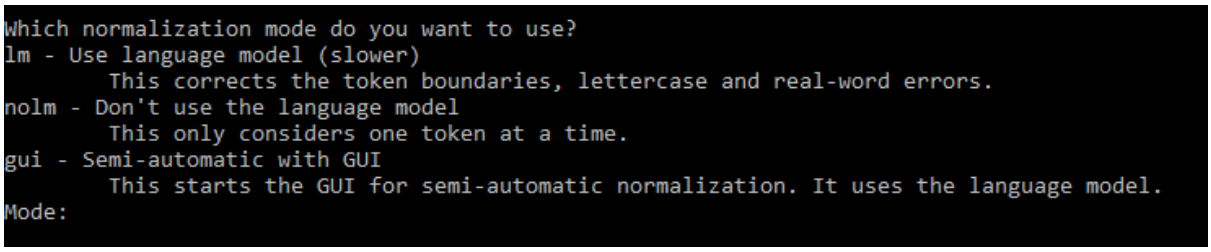

Abbildung 28: Abfrage des Normalisierungsmodus durch BoNKid

Durch die Eingabe von "lm" wird die erweiterte Normalisierung gestartet. Hierbei werden ein Sprachmodell und weitere Heuristiken zur Korrektur der Wortgrenzen und Groß- und Kleinschreibung sowie zur Erkennung von Real-Word-Errors genutzt. Dies ist etwas langsamer, als wenn "nolm" gewählt und die Standard-Normalisierung durchgeführt wird, die tokenbasiert vorgeht und keine kontextabhängigen Fehler korrigiert. Mit "gui" kann eine semi-automatische Normalisierung mittels einer graphischen Benutzeroberfläche gestartet werden.

Nach der Auswahl des Normalisierungsmodus startet die Korrektur und es werden sukzessive die Dateinamen der abgearbeiteten Texte in der Kommandozeile angezeigt. Bei den automatischen Normalisierungsmodi wird jeweils der erste Korrekturvorschlag des Systems als orthographische Zielhypothese übernommen.

Im GUI-Modus erscheint für jeden Text zunächst das folgende Fenster:

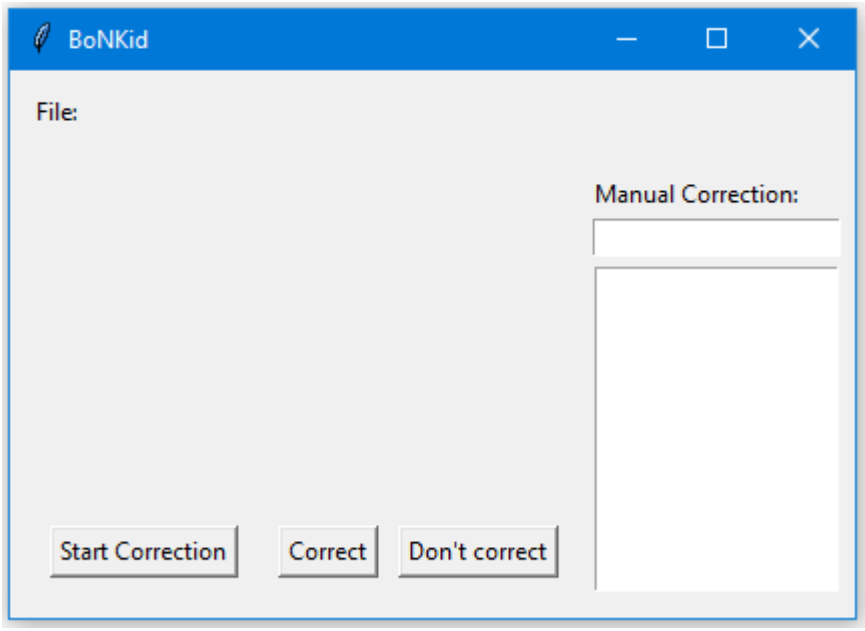

Abbildung 29: Startfenster der GUI von BoNKid

Durch einen Klick auf "Start Correction" wird die Korrektur des Textes begonnen und die GUI springt zur nächsten Stelle im Text, an der das System mehr als einen Korrekturvorschlag für ein Wort hat. Dies kann zum Beispiel wie folgt aussehen:

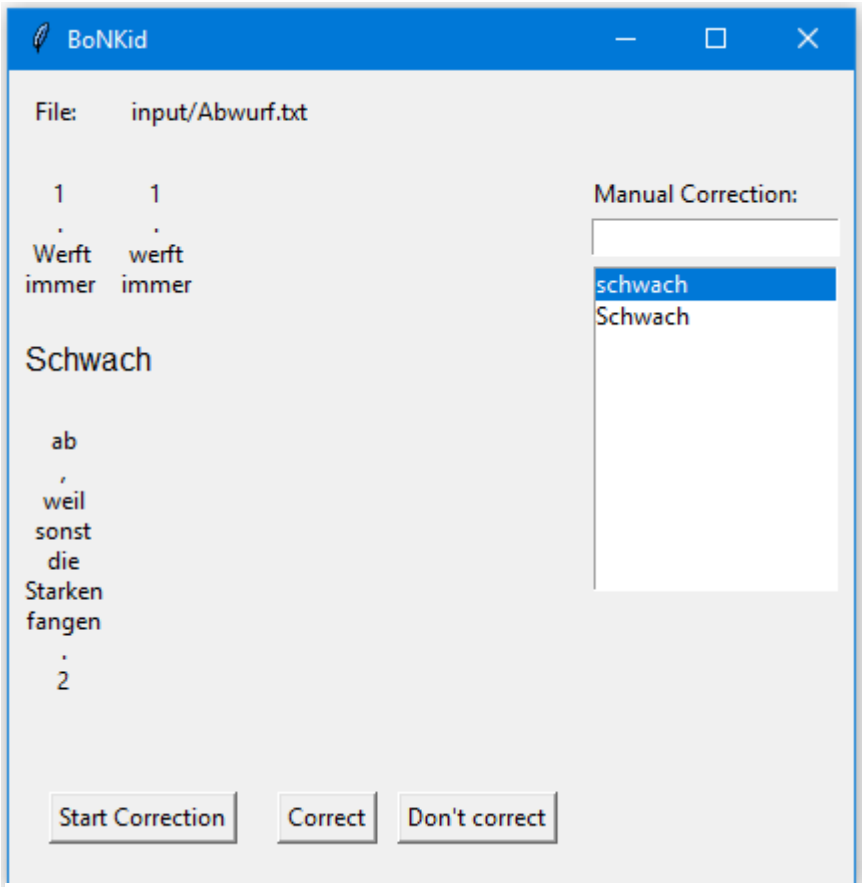

Abbildung 30: Korrektur mit der GUI von BoNKid

Oben im Fenster wird der Dateiname des Textes angezeigt, sodass die Datei im Anschluss an die Korrektur bei Bedarf manuell geöffnet und bearbeitet werden kann. Auf der linken Seite erscheinen die Transkription sowie die orthographische Zielhypothese, soweit diese bereits existiert. Das aktuelle Wort, für das die Zielhypothese bestimmt werden soll, ist größer dargestellt. Auf der rechten Seite werden in einer Liste die Korrekturvorschläge des Systems angezeigt. Darüber hinaus gibt es die Möglichkeit, eine eigene Korrektur in das Textfeld darüber einzugeben. Durch Drücken des Knopfes "Correct" wird die ausgewählte oder eingegebene Korrektur als Zielwort übernommen. Wird stattdessen "Don’t correct" gedrückt, wird die Transkription unverändert als Zielhypothese eingetragen.

Der Output des Programms ist eine dreispaltige csv-Datei zu jedem Text, die in der ersten Spalte die Token-ID tok1 bis tokN enthält, in der zweiten Spalte die Transkription und in der dritten Spalte die orthographische Zielhypothese.

tok1	1.	1.
tok2	Werft	werft
tok3	immer	immer
tok4	Schwach	schwach
tok5	ab	ab
tok6	,	,
tok7	weil	weil
tok8	sonst	sonst
tok9	die	die
tok10	Starken	Starken
tok11	fangen	fangen
tok12	.	.

csvToLearnerXML – Part III

Mit diesem Teil des Programms werden die Texte aus den vorherigen Schritten mit weiteren Informationen annotiert und in das Zielformat gebracht. Zum Starten des Programms ist

```
>> python csvToLearnerXML.py
```

einzugeben. Daraufhin fragt das Programm zunächst den Speicherort der Input-Dateien ab. Diese müssen im csv-Format vorliegen und mindestens drei Spalten mit der Token-ID, der Transkription und einer Zielhypothese enthalten. Handelt es sich bei dem einzulesenden Korpus um Texte, die keine Fehler enthalten und deshalb nicht mit dem vorherigen Programmschritt bearbeitet wurden, kann die korrekte Schreibung sowohl als Transkription als auch als Zielhypothese verwendet werden. Außerdem muss eine Token-ID tok1 bis tokN eingefügt werden.

Als Nächstes erfragt das Programm den Speicherort der originalen XML-Dateien. Hiermit sind die von filetoxml.py produzierten LearnerXML-Dateien gemeint. Wenn solche Dateien bereits erzeugt

wurden, kann der entsprechende Pfad angegeben werden, damit die zeitaufwendige Erstellung nicht wiederholt werden muss. Andernfalls werden die Dateien an dem eingegebenen Pfad neu erstellt.

Abschließend erfragt das Programm den Speicherort für die Ergebnisdateien sowie den Namen des Korpus, um den passenden Meta-Header erzeugen zu können. Implementiert sind Module für "obigeko", "h1", "litkey", "grundschulwiki" und "klexikon", die zudem den Pfad der Metadaten-Dateien abfragen. Wenn keines der verfügbaren Korpora ausgewählt wird, enthalten die Ergebnisdateien keinen Meta-Header.

Nachdem alle Pfade ermittelt sind, liest das Programm die csv-Dateien ein. Enthält eine Datei weniger als 14 Spalten – bei Dateien, die in Part II erzeugt wurden, bspw. nur drei – werden die restlichen leeren Spalten ergänzt, sodass das Programm wie mit dem in Abschnitt 3 beschriebenen csv-Format ausgeführt werden kann.

Für jeden Text wird dann filetoxml.py sowie errors_baseRate.py gestartet, um die Annotation mit Phonemen, Silben, etc. und Fehlern durchzuführen. Sind keine POS-Tags in den Input-Daten vorhanden, werden diese von filetoxml.py automatisch unter Verwendung des Stanford-POS-Taggers mit dem Kinder-Modell ergänzt.

Der Output der Skripte im Standard-LearnerXML-Format wird anschließend mit den zusätzlichen Informationen aus der csv-Input-Datei ergänzt, wenn derartige Informationen vorhanden sind. Zudem werden der Meta-Header und das Root-Token <text> eingefügt und so die finale erweiterte LearnerXML-Datei zu jedem Text erzeugt. Der Dateiname der bereits verarbeiteten Dateien wird fortlaufend in der Kommandozeile ausgegeben, ebenso etwaige Fehlermeldungen.

Prettify – Part IV

Durch die nachträgliche Zugabe des Meta-Headers ist die XML-Datei nicht mehr strukturiert formatiert, d. h. die Einrückungen verschiedener Knoten fehlen, sodass die Dateien unübersichtlicher als nötig sind. Mit der Linux-Funktion *xmllint* kann dieses Problem behoben werden. Um die Verarbeitung eines ganzen Ordners zu ermöglichen, wurde die Anwendung des Tools in einen Algorithmus eingebaut, der via Kommandozeile aufgerufen werden kann. Genutzt wird dazu der Befehl:

```
>> python pretty.py path corpus
```

Während mit *path* auf den Pfad referiert wird, an dem sich der Ordner mit den LearnerXML-Dateien des Korpus befindet, muss unter dem Argument *corpus* der Name des Korpus angegeben werden (z. B. "litkey").

Sollte die Nutzung der Kommandozeile nicht favorisiert werden oder keine Argumente angegeben werden, fragt das Programm automatisch nach dem benötigten Pfad ("Path:") und dem entsprechenden Korpus ("Corpus:").

Nach dem Aufruf befinden sich die modifizierten Dateien in dem Ordner "pretty". Sie tragen den gleichen Namen wie die Ursprungsdatei und sind, falls Unterordner vorhanden, auch unter den gleichnamigen Ordnern wiederzufinden (z.B. "pretty/development").

Auf einem Windows-PC lässt sich dieser Programmteil leider nicht ausführen. Ohne die Anwendung sähe der Anfang einer LearnerXML-Datei aus dem Litkey-Korpus mit ungleichmäßigen Einrückungen der <text>-, <meta>- und <dependencies>-Abschnitte des XML-Baums beispielsweise wie in Abb. 31 aus. Eine Anwendung des vierten Programmteils führt zu der strukturierten Ausgabe in Abb. 32.

```
<?xml version="1.0" ?><text id="001-200910-II-Weg_Anna"><meta subcorpus="Litkey">
    <text>
        <topic>Weg</topic>
        <date>June/July 2010</date>
        <federal_state>Nordrhein-Westfalen</federal_state>
        <grade>3</grade>
    </text>
    <author>
        <id>001</id>
        <gender>male</gender>
        <age>10</age>
        <languages>Türkisch, Deutsch</languages>
        <school>2</school>
        <class>GT.33</class>
        <additional_tutoring>Muttersprachlicher Unterricht Klasse 1 - 4, Deutsch-Förderunterricht Klasse 3 - 4</additional_tutoring>
    </author>
</meta><tokens>
    <token beginSentStanford="true" exist_orig="false" foreign_target="false" id="tok1" orig="ver" stts="ART" target="Der">
        <characters_orig>
            <char_o id="o1">v</char_o>
            <char_o id="o2">e</char_o>
            <char_o id="o3">r</char_o>
        </characters_orig>
        <characters_target>
            <char_t id="t1">D</char_t>
            <char_t id="t2">e</char_t>
            <char_t id="t3">r</char_t>
        </characters_target>
        <characters_aligned>
            <char_a id="a1" o_range="o1" t_range="t1"/>
            <char_a id="a2" o_range="o2" t_range="t2"/>
            <char_a id="a3" o_range="o3" t_range="t3"/>
        </characters_aligned>
        <phonemes_target>
            <phon_t id="p1" t_range="t1">d</phon_t>
            <phon_t id="p2" t_range="t2">e:</phon_t>
            <phon_t id="p3" t_range="t3">6</phon_t>
        </phonemes_target>
        <graphemes_target>
            <gra id="g1" range="t1"/>
            <gra id="g2" range="t2"/>
            <gra id="g3" range="t3"/>
        </graphemes_target>
        <syllables_target>
            <syll id="s1" plausible_orig="true" range="t1..t3" type="stress"/>
        </syllables_target>
        <morphemes_target>
            <mor id="m1" range="t1..t1" type="ART"/>
            <mor id="m2" range="t2..t3" type="INFL"/>
        </morphemes_target>
        <errors>
            <err cat="PGIII:repl_C" morph_const="na" phon_orig_ok="false" range="a1"/>
            <err cat="SN:low_up" morph_const="na" phon_orig_ok="true" range="a1"/>
        </errors>
    <dependencies><dep_dipper head="tok3" rel="det"/><dep_tueba head="tok3" rel="DET"/><dep_ud head="tok3" rel="det"/></dependencies></token>
```

Abbildung 31: LearnerXML-Datei eines Textes aus dem Litkey-Korpus vor Anwendung der *prettify*-Funktion. Unformatierte Stellen sind markiert.

```
<?xml version="1.0" encoding="utf-8"?>
<text id="001-200910-II-Weg_Anna">
  <meta subcorpus="Litkey">
    <text>
      <topic>Weg</topic>
      <date>June/July 2010</date>
      <federal_state>Nordrhein-Westfalen</federal_state>
      <grade>3</grade>
    </text>
    <author>
      <id>001</id>
      <gender>male</gender>
      <age>10</age>
      <languages>Türkisch, Deutsch</languages>
      <school>2</school>
      <class>GT.33</class>
      <additional_tutoring>Muttersprachlicher Unterricht Klasse 1 - 4, Deutsch-Förderunterricht Klasse 3 - 4</additional_tutoring>
    </author>
  </meta>
  <tokens>
    <token beginSentStanford="true" exist_orig="false" foreign_target="false" id="tok1" orig="ver" stts="ART" target="Der">
      <characters_orig>
        <char_o id="o1">v</char_o>
        <char_o id="o2">e</char_o>
        <char_o id="o3">r</char_o>
      </characters_orig>
      <characters_target>
        <char_t id="t1">D</char_t>
        <char_t id="t2">e</char_t>
        <char_t id="t3">r</char_t>
      </characters_target>
      <characters_aligned>
        <char_a id="a1" o_range="o1" t_range="t1"/>
        <char_a id="a2" o_range="o2" t_range="t2"/>
        <char_a id="a3" o_range="o3" t_range="t3"/>
      </characters_aligned>
      <phonemes_target>
        <phon_t id="p1" t_range="t1">d</phon_t>
        <phon_t id="p2" t_range="t2">e:</phon_t>
        <phon_t id="p3" t_range="t3">6</phon_t>
      </phonemes_target>
      <graphemes_target>
        <gra id="g1" range="t1"/>
        <gra id="g2" range="t2"/>
        <gra id="g3" range="t3"/>
      </graphemes_target>
      <syllables_target>
        <syll id="s1" plausible_orig="true" range="t1..t3" type="stress"/>
      </syllables_target>
      <morphemes_target>
        <mor id="m1" range="t1..t1" type="ART"/>
        <mor id="m2" range="t2..t3" type="INFL"/>
      </morphemes_target>
      <errors>
        <err cat="PGIII:repl_C" morph_const="na" phon_orig_ok="false" range="a1"/>
        <err cat="SN:low_up" morph_const="na" phon_orig_ok="true" range="a1"/>
      </errors>
      <dependencies>
        <dep_dipper head="tok3" rel="det"/>
        <dep_tueba head="tok3" rel="DET"/>
        <dep_ud head="tok3" rel="det"/>
      </dependencies>
    </token>
```

Abbildung 32: LearnerXML-Datei des Textes aus dem Litkey-Korpus nach Anwendung der *prettify*-Funktion. Die veränderten Stellen sind markiert.

6. Literaturverzeichnis

Berkling, K. (2016). Corpus for Children's Writing with Enhanced Output for Specific Spelling Purposes (2nd and 3rd Grade). In: *Proceedings of LREC*, 3002-3006. Abgerufen von http://www.lrecconf.org/proceedings/lrec2016/pdf/148 _Paper.pdf

Brants, S., Dipper, S., Eisenberg, P., Hansen, S., König, E., Lezius, W., Rohrer, C., Smith, G., & Uszkoreit, H. (2004). TIGER: Linguistic Interpretation of a German Corpus. *Journal of Language and Computation, 2,* 597-620.

Chen, D., & Manning, C. D. (2014). A Fast and Accurate Dependency Parser using Neural Networks. In: *Proceedings of EMNLP 2014.* Abgerufen von https://cs.stanford.edu/~danqi/papers/emnlp2014.pdf

Dipper, S. (2017, März 10). *Dependenzannotation (Seminar Korpuslinguistik WS16/17)*. Ruhr-Universität, Bochum.

Eisenberg, P. (2006). *Grundriss der deutschen Grammatik Band 1: Das Wort* (3. Aufl.). Stuttgart: J.B. Metzler.

Fay, J. (2010). *Die Entwicklung der Rechtschreibkompetenz beim Textschreiben: Eine empirische Untersuchung in Klasse 1 bis 4*. Frankfurt a. M.: Peter Lang.

Foth, K. A. (2006). *Eine umfassende Constraint-Dependenz-Grammatik des Deutschen*. Abgerufen von http://edoc.sub.uni-hamburg.de/informatik/volltexte/2014/204/pdf/foth_eine_umfassende_.pdf

Frieg, H. (2014). *Sprachförderung im Regelunterricht der Grundschule: Eine Evaluation der Generativen Textproduktion* (Doktorarbeit). Abgerufen von http://www-brs.ub.ruhrunibochum.de/netahtml/HSS/Diss/FriegHendrike/diss.pdf

Laarmann-Quante, R. (2015). *Automatic Analysis of Orthographic Properties of German Words* (Masterarbeit). Abgerufen von https://www.linguistics.rub.de/~laarmann-quante/pub/MAThesis_Laarmann-Quante.pdf

Laarmann-Quante, R., Knichel, L., Dipper, S., & Betken, C. (2016). Annotating spelling errors in German texts produced by primary school children. In: *Proceedings of the ACL Linguistic Annotation Workshop (LAW X),* 32-42. Abgerufen von https://www.linguistics.ruhr-uni-bochum.de/litkey/pubfiles/laarmann_etal_2016_law.pdf

Laarmann-Quante, R., Ortmann, K., Ehlert, A., Betken, C., Dipper, S., & Knichel, L. (2017). *Guidelines for the Manual Transcription and Orthographic Normalization of Handwritten German Texts Produced by Primary School Children*. Bochumer Linguistische Arbeitsberichte (BLA), Vol. 20. Abgerufen von https://www.linguistics.ruhr-unibochum.de/bla/020laarmannquante_etal2017.pdf

Reichel, U. D. (2012). PermA and Balloon: Tools for string alignment and text processing. In: *Proceedings of Interspeech*. Abgerufen von https://epub.ub.uni-muenchen.de/18042/1/ReichelIS2012.pdf

Reichel, U. D., & Kisler, T. (2014). Language-independent grapheme-phoneme conversion and word stress assignment as a web service. In: R. Hoffmann (Hrsg.), *Elektronische Sprachverarbeitung: Studientexte zur Sprachkommunikation 71* (S. 42–49). Dresden: TUDpress.

Schiller, A., Teufel, S., Stöckert, C., & Thielen, C. (1999). *Guidelines für das Tagging deutscher Textcorpora mit STTS (Kleines und großes Tagset)*. Abgerufen von http://www.ims.unistuttgart.de/forschung/ressourcen/lexika/TagSets/stts-1999.pdf

Schmid, H. (1995). Improvements in Part-ofSpeech Tagging with an Application to German. In: *Proceedings of the ACL SIGDAT-Workshop*, 47-50.

Schroeder, S., Würzner, K.-M., Heister, J., Geyken, A., & Kliegl, R. (2015). childLex: A lexical database of German read by children. In: *Behavior Research Methods*(47), 1085-1094.

Schroff, C. (2000). *Lea, Lars und Dodo: Bilderbox*. Schaffhausen, CH/Braunschweig, D: SCHUBI Lernmedien.

Schulte, M., & van Dijk, Z. (2015). *Projekt Freies Kinderlexikon. Konzept für eine Kindgerechte Wiki-Enzyklopädie. („Wikipedia für Kinder“ / „KlexikonProjekt“)*. Abgerufen von https://upload.wikimedia.org/wikipedia/commons/8/84/Konzept_Wikipedia_für_Kinder_-_Projekt_Freies_Kinderlexikon.pdf

Telljohann, H., Hinrichs, E., & Kübler, S. (2004). The TüBa-D/Z Treebank: Annotating German with a Context-Free Backbone. In: *Proceedings of LREC*, 2229–2235. Abgerufen von http://cl.indiana.edu/~skuebler/papers/tuebad.pdf

Thelen, T. (2000). *Osnabrücker Bildergeschichtenkorpus. Version 1.0.0.* Abgerufen von http://tobiasthelen.de/uploads/Wissenschaft/osnabruecker_bildergeschichtenkorpus_1_0_0.pdf

Thelen, T. (2010). *Automatische Analyse orthographischer Leistungen von Schreibanfängern* (Dissertation). Abgerufen von https://repositorium.uni-osnabrueck.de/handle/urn:nbn:de:gbv:700-201006096307

Toutanova, K., Klein, D., Manning, C., & Singer, Y. (2003). Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In: *Proceedings of HLT-NAACL 2003*, 252-259. Abgerufen von http://nlp.stanford.edu/~manning/papers/tagging.pdf