Eine Ressource durch Abwägung: Wie bestehende generische Softwarelösungen ‚über sich hinauswachsen' und ein Parallelkorpus ermöglichen

Martin Klotz und Thomas Krause

1 Motivation

Das Pro*Niedersachen-Projekt ‚Wiedererzählen im Norden. Digitale Analyse weltlicher Erzähltexte in niederdeutschen Inkunabeln' (WiN) sammelt und annotiert frühneuhochdeutsche (fnhd.) und mittelniederdeutsche (mnd.) Versionen von Erzähltexten mit dem Ziel der systematischen Analyse und Auswertung von Übersetzungs-/Übertragungsstrategien. Zu diesem Zweck werden beide Fassungen eines Textes hinsichtlich morphosyntaktischer und lexikalischer Merkmale annotiert. Zudem werden Wortkorrespondenzen zwischen beiden Textvarianten identifiziert, annotiert und bezüglich ihrer Übertragungsdiskrepanzen kategorisiert. Das Ziel dieser Annotationen ist die Erschließung syntaktischer, morphologischer und lexikalischer Muster im Prozess gegebener Textübertragung.

Um für das WiN-Projekt relevante Fragen beantworten zu können, müssen sowohl die fnhd. als auch die mnd. Version eines Textes intratextuell sowie intertextuell annotiert werden. Als intratextuelle Annotationen sind solche Annotationen zu verstehen, die sich nur auf Elemente (Wörter oder Spannen von Wörtern) einer Fassung eines Textes beziehen und innerhalb eines technischen Dokumentes annotiert werden können. Intertextuelle Annotationen hingegen zeichnen Beziehungen und deren Eigenschaften zwischen zwei Textversionen bzw. den Elementen dieser

DOI: https://doi.org/10.17875/gup2024-2620

Versionen aus und können nicht problemlos innerhalb eines technischen Dokumentes ausgewiesen werden. Auf intratextueller Ebene benötigt ein Korpus wie das des WiN-Projekts lexikalische, morphologische, morphosyntaktische, syntaktische und textstrukturelle Informationen, um Fragen aus linguistischer und literaturwissenschaftlicher Sicht gezielt und effizient untersuchen zu können. Um Erklärungsansätze für Phänomene, die an der Übertragungsschnittstelle zwischen zwei Versionen eines Textes zu beobachten sind, untersuchen zu können, bedarf es zudem auf intertextueller Ebene der Annotation von Wort- bzw. Phrasenkorrespondenzen zwischen beiden Fassungen sowie einer Kategorisierung vorliegender Diskrepanzen. Auf diese Weise lassen sich wiederkehrende Muster in der Textübertragung erfassen und quantitativ auswerten, was wiederum Rückschlüsse auf Übertragungsstrategien zulässt.

Für Ressourcenersteller- und -nutzer:innen soll der vorliegende Beitrag zeigen, wie wichtig eine frühzeitige Charakterisierung der Strukturierung der eigenen Daten ist, um in einer frühen Arbeitsphase die Anforderungen an die eigene Toolchain zu kennen. Hierbei müssen nicht in allen Fällen die bereits implementierten Fähigkeiten der Tools berücksichtigt werden, solange in deren Rahmen eine effiziente und zielführende Behelfslösung gelingt.

In Hinblick auf Toolersteller:innen möchte die folgende Darstellung für pragmatische Lösungen werben: Generische (und damit komplexe) Lösungen sind nur dann nachhaltig, wenn eine vielfältige Nachnutzung der eigenen Prozesse und Werkzeuge überhaupt erreichbar ist. Dagegen kann ein weniger generischer, sondern an den wohldefinierten Anforderungen orientierter Ansatz bei gleicher Transparenz schneller und damit effizienter zum Ziel führen. Bedarfe und Konzepte der Ressourcenersteller:innen müssen hierzu nur prototypisch mit bestehenden Tools abgebildet werden. Das so erlangte bessere Verständnis des eigentlichen Bedarfs aufseiten der Ressourcenersteller:innen erlaubt im Idealfall eine besser abgestimmte und fokussiertere Entwicklung generischer Tools in einem Folgeschritt. Gleichzeitig kann auf diesem Wege die Notwendigkeit eines solchen Schritts evaluiert werden.

Unter diesen Gesichtspunkten ergeben sich Perspektiven für Nachhaltigkeit und Forschungscommunity. Die Entwicklung von Tools und Skripten zur Erstellung von Korpora zählt nicht in jedem Fall zu den Zielen eines Forschungsprojekts, da die eigentlichen Forschungsdaten und deren Auswertung im Zentrum stehen sollten. Für das WiN-Projekt wurde ein Ansatz gewählt, bei dem bestehende Software durch begrenzte Neuentwicklungen ergänzt wird. Durch die Nachnutzung bestehender Software sollen Entwicklungsressourcen effizienter eingesetzt und mehr Forschung ermöglicht werden. Auch die Erweiterungen selbst sollen anderen Forscher:innen zur Verfügung gestellt werden, um die eine spätere Erweiterung des WiN-Korpus oder die Erstellung eines vergleichbar strukturierten Korpus zu ermöglichen. Die Software in diesem Projekt kommt damit der Forschungscommunity zugute, auch wenn sie selbst nicht für alle denkbaren Einsatzszenarien entwickelt worden, sondern als Nebenprodukt der Datenerstellung entstanden ist.

2 Vorarbeiten

2.1 Datengrundlage

Die Datenbasis des WiN-Korpus sind gedruckte Erzählungen, deren Versionen auf Mnd. und Fnhd. vorliegen. Es handelt sich bei den gepaarten Erzählungen zumeist um Übertragungen vom Fnhd. ins Mnd.[1] Alle in das Korpus aufgenommenen Texte stammen von 1470 bis 1510 (vgl. Tabelle 1).

Text
‚Dracula'
‚Juden von Sternberg'
‚Griseldis'
‚Graf im Pflug'
‚Vier Kaufleute'
‚Sieben weise Meister' (Auszüge)
‚Bruder Rausch'

Tabelle 1: Texte im Korpus

2.2 Probleme etablierter Ansätze und Tools

Intertextuelle Phänomene und tiefe linguistische Modellierung eines Textes schließen einander auf praktischer Seite nahezu aus. Datenformate, die dediziert sowohl intra- und intertextuelle Annotationen speichern bzw. deren Modellierung repräsentieren und für die auch graphische Editoren zur Verfügung stehen, gibt es bisher nach unserer Kenntnis nicht.

Die Verarbeitung linguistischer Daten zu Parallelkorpora wird oft erschwert durch das Verhältnis von Text, Dokument und Datei. Als Text sei hier eine zusammenhänge Folge linguistischer Einheiten verstanden, die aus einer Quelle von Primärdaten stammen. Ein Dokument wird in diesem Kontext definiert als abstrakter Container, der einen oder mehrere Texte, aber auch deren Annotationen enthält. Dateien wiederum sind vom Computer als zusammenhängende und gemeinsam mittels eines Pfades identifizierte Bereiche des Festplattenspeichers, die in einem bestimmten Format vorliegen (zum Beispiel XML, JSON etc.). Das WiN-Projekt strebt mit seinen intertextuellen Annotationen an, Beziehungen zwischen Texten darzustellen und damit durchsuch- und analysierbar zu machen. In ihrer ursprünglichen digitalisierten Form liegen diese Texte in verschiedenen Dokumenten vor, aber auch in verschiedenen Dateien. Die Abbildung von Dokument zu Datei ist hier zunächst 1:1, auch das Verhältnis von Dokument zu Text und folglich von

[1] Vgl. die Angaben zu den verwendeten Drucken bei Anabel Recker (in diesem Band) und Coniglio et al. (2019).

Datei zu Text. Aus diesem sehr etablierten Verhältnis haben sich über die Jahrzehnte computergestützter Annotation Probleme entwickelt, die eine intertextuelle Annotation erschweren und in vielen Fällen sogar unmöglich machen. Beziehungen zwischen Texten bzw. Dokumenten sind in einem solchen Verhältnis nicht modellierbar. Es gibt keine Tools und wenige Formate, die das befriedigend ermöglichen. PAULA XML (Zeldes et al. 2013), als positives Beispiel, repräsentiert alle Annotationen eines Textes anders als üblich in separaten Dateien. Die Gesamtheit aller Dateien bzgl. eines Textes ist in PAULA XML das abstrakte Dokument. Auch intertextuelle Beziehungen lassen sich in PAULA XML repräsentieren. Andere Formate und deren Tools, die dediziert zur Repräsentation intertextueller Beziehungen wie Textalignierung entwickelt wurden (zum Beispiel TMX, Savourel 2005), bieten sich zwar ebenso als Lösung an, unterstützen aber keine intratextuelle Annotation und damit keine tiefe linguistische Modellierung. Genau genommen handelt es sich hierbei zudem um eine (legitime) Behelfslösung, da mehrere Texte in einem Dokument untergebracht werden, die 1:1-Beziehung von Dokument zu Datei aber fortbesteht. Im Gegenzug unterstützen Tools wie EXMARaLDA (Schmidt 2012) und dessen Format EXMARaLDA-XML viele Annotationsebenen, sind aber für einzelne Texte konzipiert. Dies wird zum Problem für eine:n Annotator:in, sobald Tools, die auf solchen Formaten aufbauen, eingesetzt werden. Arbeitsabläufe und Verarbeitungsketten, die sowohl intertextuelle als auch intratextuelle Annotationen gemeinsam erfassen, stellen die Korpuspipeline als Konzept vor Herausforderungen bzw. erfordern Behelfslösungen, um innerhalb der Grenzen des Dokument-/Dateikonzepts zu bleiben und trotzdem parallele Texte gemeinsam zu annotieren.

Die beschriebenen Probleme setzen sich implizit auch im Datenmodell von Korpussuchmaschinen fort, wenn diese einzelne Textdateien direkt als Dokument abbilden, wie zum Beispiel AntConc (Anthony 2020) oder Tregex (Levy/Andrew 2006). ANNIS, ein Such- und Visualisierungstool für linguistische Daten (Krause/Zeldes 2016), organisiert ein Korpus dagegen analog zu PAULA XML in einzelnen Dokumenten, die mehrere Texte beinhalten können. Zwischen den Texten eines Dokumentes können Annotationen in Form von Kanten dargestellt werden.

2.3 Salt und andere graphbasierte Annotation

Salt ist ein Modell, das ursprünglich der Repräsentation linguistischer Daten dient. Als flüchtige Zwischenrepräsentation erlauben Salt-Graphen flexible Konversionen zwischen Dateiformaten in Pepper (Zipser/Romary 2010). Linguistische Daten können mittels Pepper von einem Format in ein anderes überführt werden, indem ein Import-Modul die Inhalte der Ausgangsdatei als Zwischenschritt in einem Salt-Graph repräsentiert, der anschließend über ein Export-Modul ins Zielformat über-

führt wird. Pro Format, das von Pepper unterstützt wird, steht idealerweise ein Import- und ein Export-Modul zur Verfügung, sodass flexible Konvertierung möglich ist.

Salt als graphbasiertes Framework kennt verschiedene Entitäten zur Modellierung linguistischer Konzepte. Dazu gehören Token, Spannen, Konstituenten, eine Timeline als verschiedene Knotenarten; darüber hinaus verschiedene Arten von typisierten Kanten. Diese Elemente können mit Name-Wert-Paaren (also zum Beispiel pos=NN mit ‚pos' als Namen und ‚NN' als Wert) annotiert werden. Eine Timeline ermöglicht es, zeitlich alignierte Texte (zum Beispiel Dialogaufnahmen, Sauer/Lüdeling 2016) sowie multiple Tokenisierungen abzubilden (zum Beispiel im RIDGES-Korpus, Odebrecht et al. 2017). Nicht-hierarchische Kanten (in Salt Pointing relations genannt) sind Kanten zwischen Token oder anderen Knoten, die auf eine zunächst nicht weiter spezifizierte Beziehung zwischen Quell- und Zielelement hinweisen. Sie können genutzt werden, um Dependenzsyntax oder aber die Alignierung von Token oder größeren Einheiten zu repräsentieren.

Die Repräsentation linguistischer Daten als Graph (also als eine aus Knoten und Kanten bestehende Struktur) hat in der Linguistik eine lange Tradition und ist somit auch als technische Repräsentation naheliegend. Graphen sind sehr mächtige Strukturen, die eine Vielzahl von Anwendungsgebieten in der Linguistik haben. Neben der Syntax als vermutlich bekanntestes Anwendungsbeispiel lassen sich auch andere Phänomene wie Koreferenz, semantische Repräsentationen oder rhetorische Struktur (Mann/Thompson 1987) mittels Graphen effizient darstellen. Zudem erlauben Graphen eine ergänzende Erfassung sprachlicher Daten, die sonst nur durch sequenzielle Abfolgen und evtl. überlappende Annotationen beschrieben werden können. Tiefer liegende Beziehungen, wie die oben genannten Beispiele, sind ohne Graphen deutlich schwieriger abzubilden. Neben der reinen Beschreibung und Modellierung linguistischer Daten helfen Graphen zudem bei der Vermessung, d. h. der quantitativen Auswertung linguistischer Daten (Shadrova 2020).

Neben Salt existieren weitere technische Frameworks, die linguistische Daten mittels Graphen verarbeiten oder repräsentieren, zum Beispiel GraphANNIS (Krause 2019), GREW (Guillaume 2021), GraphAnno (Gast et al. 2015) und viele mehr.

2.4 Anforderungen

Aus Sicht der Ressourcenersteller:innen sowie Toolchainentwickler:innen ist für das WiN-Korpus eine möglichst kleine Menge an Annotationswerkzeugen erstrebenswert. Je weniger Tools in den Arbeitsprozess integriert werden müssen, desto geringer ist der Aufwand der Integration dieser in eine Pipeline. Gleichzeitig steigt die Robustheit der Toolchain, denn die Anzahl möglicher Fehlerquellen reduziert sich mit der Zahl der eingebundenen Werkzeuge und Dateiformate. Ein gemeinsames Werkzeug zum Erstellen von inter- und intratextuellen Annotationen hat zudem Vorteile für die Annotator:innen, die sich nicht auf

mehrere Arbeitsumgebungen einstellen müssen. Zudem kann ein idealerweise einheitliches Annotationswerkzeug für beide Annotationsarten auch einheitliche Modellierung, d. h. die Verwendung artgleicher Annotationsstrukturen in allen Annotationen, bedeuten. Ein Annotationswerkzeug in diesem Sinne unterstützt also die Annotation auf mehreren Ebenen.

Erforderlich ist für die Nutzer:innen der Ressource zudem eine gemeinsame Durchsuchbarkeit aller Annotationen, unabhängig davon, ob nur intratextuelle, intertextuelle oder beide Annotationsarten durchsucht werden sollen. Dies stellt sowohl Anforderungen an die Repräsentation der Daten (Kompatibilität mit generischen Suchtools) sowie die zugrundeliegende Korpusstruktur (Parallelkorpus statt Vergleichskorpora).

Alle Prozesse wie Annotation, Modellierung, Konvertierung und Korpussuche setzen nach Möglichkeit auf bestehende Softwarelösungen. Hierdurch verschiebt sich der Arbeitsschwerpunkt von Entwicklung und Wartung auf Integration und Anpassung existierender Tools, was zeitlich aufwendige Eigenentwicklungen vermeidet. Zugleich bedeutet dies eine nachhaltigere Nutzung von Software. Eigenentwicklungen – sollten sie notwendig sein – müssen zwingend unter ähnlicher Maßgabe und unter Berücksichtigung der FAIR-Prinzipien (Wilkinson et al. 2016) und der Verwendung freier Lizenzen öffentlich zugänglich gemacht werden. Diese Aspekte betreffen Ressourcennutzer:innen und -ersteller:innen, die Entwickler:innen der Toolchain, sowie letztlich die Forschungsgemeinschaft mit Fokus auf Softwarenachhaltigkeit. Bestehende Software hat in den meisten Fällen bereits Entwicklungsstadien und Anpassungen durchlaufen und sich so idealerweise den Bedürfnissen seiner Nutzer:innen angeglichen, was bei Eigenentwicklungen zwangsläufig nicht der Fall sein kann. Somit schont Nachnutzung bestehender Tools auch die eigenen finanziellen und zeitlichen Ressourcen.

Ähnlich der Korpuspipeline müssen spätestens im Anschluss an die Korpuserstellung alle Annotationsentscheidungen (anhand von Guidelines) sowie die entstehende Ressource (das Korpus) transparent und nachvollziehbar dokumentiert sein. Ressourcennutzer:innen benötigen Zugang zu Annotationsentscheidungen, um Ergebnisse der Korpussuche adäquat einordnen zu können bzw. die für sie interessanten Phänomene überhaupt zu finden. Außerdem ist eine langfristige Archivierung der Daten und Dokumentation in entsprechend dafür ausgelegten Archiven notwendig. Für ein nachhaltigeres Verständnis von Korpuserstellung ist es wichtig, auch die Archivierung und Veröffentlichung als Teil der Korpuspipeline zu verstehen. In dieser Fallstudie wird dieser Aspekt aber im Folgenden nicht weiter beleuchtet.

3 Datenmodellierung und Prozesse

3.1 Datenmodell

Im WiN-Korpus werden die fnhd. Texte als Ausgangstexte und die mnd. Texte als Übertragungen aufgefasst. Beide sollen als Parallelkorpus mit expliziten Alignierungskanten dargestellt werden. Dazu werden für jede Übertragung eigenständige Token erzeugt, die einem Text zugeordnet sind. Alle Annotationen, die mehrere Token umspannen können, werden als Knoten vom Typ ‚Spanne' in Salt dargestellt und ebenfalls einem Text zugeordnet. Die übertragenen Texte und der Ausgangstext sind Teil desselben Dokumentes. Ein Beispiel ist in Abbildung 1 gegeben.

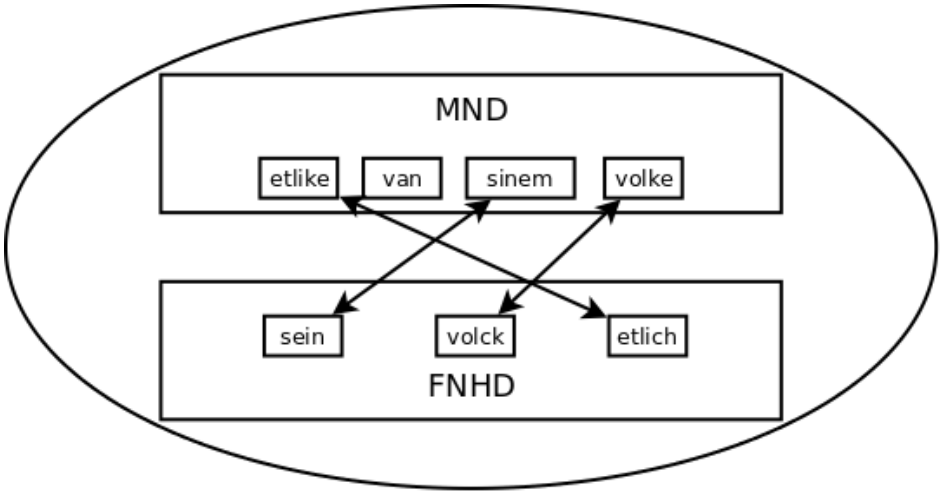

Abbildung 1: Beispielannotation eines Dokumentes mit den zwei Texten ‚FNHD' und ‚MND' sowie den Token und den Alignierungskanten zwischen den Token.

In diesem Datenmodell werden die Alignierungsinformationen direkt mit dargestellt, also zum Beispiel nicht in einer separaten Datei vermerkt. Diese Entscheidung zur Modellierung der Annotationen basiert auf dem Konzept von Salt, als Zwischenmodell alle Informationen eines Dokumentes komplett darzustellen. Damit benötigen die Exportmodule in Pepper, wie zum Beispiel das Modul für das ANNIS-Format, kein Wissen über die unterschiedlichen Möglichkeiten, solche Alignierungskanten in Dateien oder Formaten zu codieren. Das heißt aber auch, dass diese Alignierungskanten bereits vorher durch ein Import- oder Manipulator-Modul von Pepper in den Annotationsgraph eingefügt werden müssen.

3.2 Annotationsprozess

Die Alignierung des mnd. und fnhd. Textes erfolgt manuell, da das Ziel der Aufbereitung der WiN-Daten die automatische Extraktion bestimmter Übertragungsmuster aus einer sprachlichen Varietät in die andere ist. Dies erfordert eine akkurate Annotation sowie die Alignierung auf Wortebene. Anders als in den meisten Parallelkorpora ist für die Fragestellung des WiN-Projekts eine Alignierung für größere Einheiten nicht ausreichend, da sonst Variation auf kleinerer Ebene (Syntax, Lexik und Morphologie) nicht ausreichend erfasst werden kann. Die Wortalignierung wiederum lässt sich für die gegebenen Varietäten Fnhd. und Mnd. nicht mit ausreichender Akkuratheit automatisch annotieren, so dass eine manuelle Annotation allein aus diesem Grund erforderlich ist. Darüber hinaus ist die Klassifikation von Alignierungsbeziehungen, die ebenfalls stattfinden soll, zum Zeitpunkt der Annotation noch Forschungsgegenstand. Die Diskrepanzphänomene, die es durch verschiedene Kategorien gegeneinander abzugrenzen gilt, werden zum einen während des Annotationsprozesses exploriert. Zum anderen sind vorab definierte Kategorien unter Umständen nicht mit ausreichender Übereinstimmung annotierbar, d. h. wohldefiniert, was ebenfalls eine Anpassung einzelner Kategorien oder des Kategoriensets erforderlich machen kann. Dies schließt eine ad hoc automatische Annotation von Alignierungskanten ebenfalls aus.

Die manuelle Annotation der Daten erfolgt in EXMARaLDA, die Annotation der intratextuellen Merkmale wird mittels Spannen- und Tokenlabels vorgenommen, was dem üblichen Nutzungsszenario dieses Tools entspricht. Die Fnhd.- und Mnd.-Varianten werden beide in einem EXMARaLDA-Dokument abgebildet. Für beide Varietäten liegt der Tokentext diplomatisch transkribiert (dipl) und orthographisch normalisiert (norm) vor (Coniglio et al. 2019). Die normalisierten Ebenen beider Texte werden intratextuell bezüglich ihres Lemmas, ihrer Wortart, Satzspannenzugehörigkeit sowie textstruktureller Merkmale annotiert. Die intertextuelle Annotation, also die Alignierungsbeziehung zwischen den Token sowie die Klassifikation dieser Beziehung erfolgt mittels zweier Ebenen je Varietät: ‚align' und ‚align_tag'. Zunächst sind die Token des mnd. und des fnhd. Textes zeitlich nach ihrer Reihenfolge im Text repräsentiert, d. h. die Token laufen im Dokument parallel. Entsprechen das mnd. und das fnhd. Token zu einer Zeiteinheit auch tatsächlich einander im Sinne einer direkten Übertragung, so wird auf der Ebene ‚align' die ID ‚x' vergeben. Hier kann die Alignierung später also unmittelbar aus der Reihenfolge im Text abgeleitet werden. Sollte eine solche Entsprechung aber nicht gegeben sein, so kann dies über die Vergabe anderslautender IDs auf der jeweiligen ‚align'-Ebene für Mnd. und Fnhd. kenntlich gemacht und eine zweifelsfreie Zuordnung gewährleistet werden. Entsprechend ergibt sich für die Zuordnung von IDs zwischen Mnd. und Fnhd. – sowohl im Standardfall ‚x' als auch bei anderslautenden eindeutigen IDs – eine Kategorisierung auf der entsprechenden Ebene ‚align_tag', sollte im beschriebenen Übertragungsprozess eine Diskrepanz zwischen beiden Texten vorliegen (zum Beispiel syntaktische Neuordnung). Ein solches Tag wird nicht nur

zwangsweise individuell für eine ID, sondern auch für mehrere IDs gemeinsam vergeben, wenn bspw. eine Phrase in eine andere Phrase übertragen wurde und dieser Prozess in seiner Gesamtheit zu charakterisieren ist. Das ‚align_tag' ist dabei richtungsabhängig und für Mnd. und Fnhd. nicht zwangsweise identisch. Ein Annotationsbeispiel ist gegeben in Abbildung 2 .

Fnhd [sentence_chunk]			SU																
Fnhd [lemma]				er	haben		sein 2	volk 2	etlich 1	nacket	lassen				eingraben				bi
Fnhd [pos]	D			PPER	VAFIN		DPOSA	NA	DIS	ADJD	VVINF				VVINF				A
Fnhd [dipl]		.		Er	hat		ſein	volck	etlich	nacket	laſen				eĩ	grab	/	en	pil
Fnhd [fnhd]		.		**Er**	**hat**		**sein**	**volck**	**etlich**	**nacket**	**lasen**				**eingraben**				**pi**
Fnhd [align_tag]			NC			NC	SYN		SYN			NC			MO				
Fnhd [align]			x	x		19a	18	18a	19	x		x			x				x
Mnd [align]			x	x		19	19a	18	18a	x		x			x				x
Mnd [align_tag]			ET			SYN	ET	SYN				ET			MO				
Mnd [mnd]		.	**Item**	**he**	**hefft**	**etlike**	**van**	**sineme**	**volke**	**naket**	**laten**	**in**	**de**	**erde**	**grauen**				**be**
Mnd [dipl]		¶	Item	he	hefft	etlike	van	ſyneme	volke	naket	laten	in	de	erde	grauen				be
Mnd [pos]	D		FM	PPER	VAFIN	DIS	APPR	DPOSA	NA	ADJD	VVINF	APPR	DDART	NA	VVINF				A
Mnd [lemma]			item	hê	hebben	etlik	van	sîn	volk	naket	lâten	in	dê	erde	graven				be
Mnd [sentence_chunk]			SU																

Abbildung 2: Annotation in EXMARaLDA

3.3 Konvertierung

Die annotierten EXMARaLDA-Dateien werden als einzelne Dokumente mittels Pepper und des zugehörigen Import-Moduls für das EXMARaLDA-Format in das Salt-Datenmodell überführt. Hierbei werden die Textebenen als Token importiert und alle vergebenen Annotationen – einschließlich der ‚align'- und ‚align_tag'-Ebenen – als Label auf Spannen über ein oder mehrere dieser Token repräsentiert. Das Ergebnis dieses Schritts ist ein Saltgraph, von dem ein Ausschnitt in Abbildung 3 beispielhaft gegeben wird.

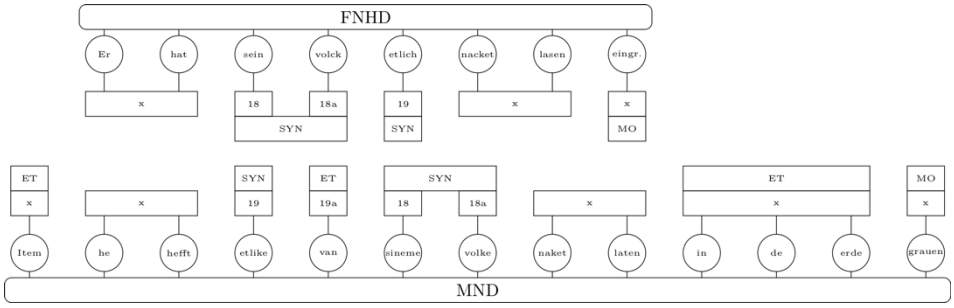

Abbildung 3: Saltgraph der annotierten Daten nach dem Import mittels des EXMARaLDA-Importers – die beiden Texte sind nur durch ihren Dokumentcontainer vereint. Die Subgraphen der beiden Texte inklusive ihrer Annotationen sind nicht zusammenhängend. Intratextuelle Annotationen sind zur besseren Übersicht nicht dargestellt.

Die tatsächliche Erstellung der Alignierung erfolgt erst nach Abschluss des Imports über ein zu diesem Zweck entwickeltes Alignierungsmodul (Aligner) für das Konvertierungsframework Pepper[2]. Dieser sogenannte Manipulator verarbeitet abgestimmt auf den hier beschriebenen Annotationsworkflow die bereits als Salt-Graph importierten Daten auf Dokumentebene. Im Verlauf der Manipulation des Graphen werden die vergebenen Annotations-IDs der beiden ‚align'-Ebenen ausgelesen und Token mit gleicher ID bzw. an gleicher Stelle in der Timeline (Standardfall) mittels zweier nicht-hierarchischer Kanten verknüpft (je eine für jede Richtung der Alignierung). Darüber hinaus wird an dieser Kante ein Label mit dem entsprechenden ‚align_tag' erstellt. Der in Abbildung 3 gezeigte Saltgraph wird durch den Manipulator in den in Abbildung 4 gezeigten umgewandelt.

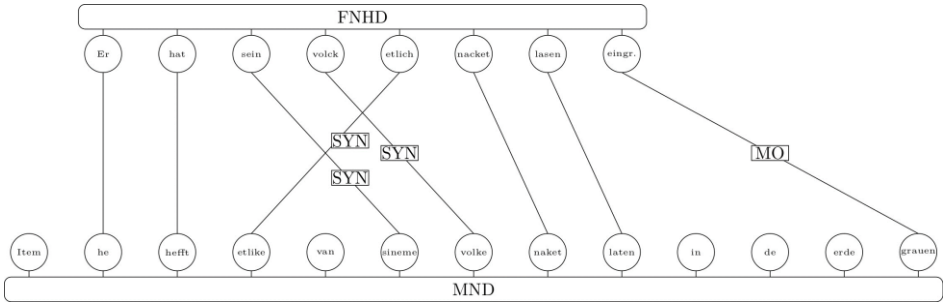

Abbildung 4: Saltgraph der annotierten Daten nach Manipulation durch den Aligner – der Graph innerhalb des Dokumentes ist nun zusammenhängend.

Nach Abschluss dieses Manipulationsschritts wird jedes Dokument mittels des ANNIS-Exporters in das ANNIS-Format umgewandelt, um die Daten in das Suchtool importieren zu können. Das ebenfalls starre Dokumentkonzept in ANNIS, das keine Annotation über dessen Grenzen erlaubt, kann umgangen werden, indem die Fnhd.- und Mnd.-Fassungen jeweils als Text innerhalb desselben Dokumentes repräsentiert sind. Eine Visualisierung des bisher gezeigten Beispiels in ANNIS ist in Abbildung 5 gegeben.

[2] Eine Pepper-Version, die das Modul enthält, ist unter https://doi.org/10.5281/zenodo.6375088 verfügbar. Der Quelltext des Alignierungsmoduls ist veröffentlicht unter https://github.com/korpling/pepper-Modules-AlignmentModule.

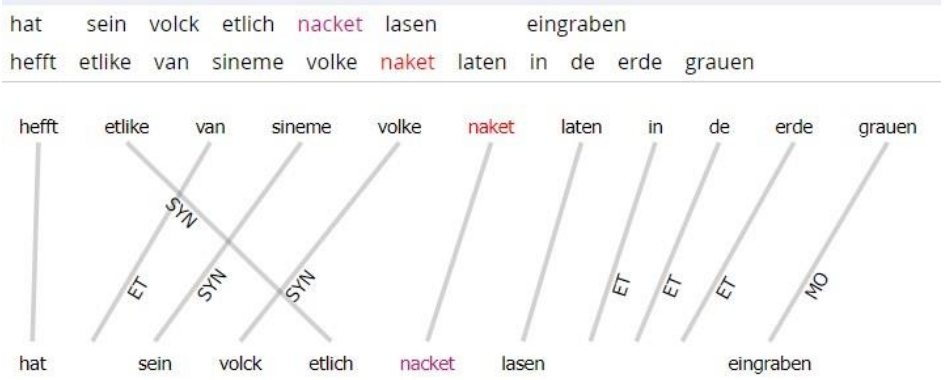

Abbildung 5: Visualisierung der Korpusdaten und der Alignierungsannotation in ANNIS

Dieses Vorgehen erfüllt die Anforderungen der Ressourcenersteller:innen, vereinfacht aber auch wesentlich die Arbeit der Toolchainentwickler:innen und ist im Interesse der Forschungscommunity zu nachhaltiger Softwarenutzung: Pepper-Module sind flexibel anpassbar und erweiterbar, da das Konvertierungsframework, in das sie eingebettet sind, bereits voll entwickelt ist und nicht weiter modifiziert werden muss. Lediglich die Funktionalität der Transformation eines Salt-Graphen in einen anderen muss für einen Manipulator implementiert werden. Eine Erweiterung der Funktionalität ist zu jedem Zeitpunkt möglich. Über einen Mechanismus zur Konfiguration des Laufzeitverhaltens von Modulen lässt sich eine generische Anwendbarkeit sicherstellen, soweit diese gewünscht ist.

4 Fazit

Die vorliegende Fallstudie illustriert, dass Softwarenachhaltigkeit nicht zwingend generische Lösungen erfordert, zumindest wenn es um die tatsächliche Nachnutzung bestehender, ausreichend generischer Entwicklungen geht (in diesem Fall EXMARaLDA, ANNIS und Pepper). Gerade die in nachhaltigen Entwicklungsprozessen eher vermiedene Zweckentfremdung bestehender Konzepte kann, wie hier gezeigt, in der Nachnutzung auch mehr Nachhaltigkeit bedeuten. Gleichzeitig ist das ressourcensparende Vorgehen bei der Erarbeitung der präsentierten Pipeline ein Beispiel für effiziente Entwicklung bzw. kann Impulsgeber für die Verteilung auch beschränkter Entwicklungsressourcen sein. Damit knüpfen die Prozesse an die für Toolchainentwickler:innen kritische Bedarfsevaluierung an und stellen deren Bedeutung heraus. Gleichzeitig wird aber auch deutlich, dass auf generische Tools allgemein nicht verzichtet werden kann, sondern dass diese für eine Forschungslandschaft unabdingbar sind unter der Maßgabe, dass sie eben vielfältige und hochfrequente Nachnutzung, wie hier vorgestellt, ermöglichen. Der Schlüssel liegt somit

im Bedienen möglichst generischer Bedürfnisse, solange dabei keine Redundanz in der Landschaft bestehender Lösungen entsteht. Für Ressourcenersteller:innen ergeben sich so neue Perspektiven (siehe zum Beispiel die Untersuchungen von Chiara de Bastiani in diesem Sammelband), besonders wenn die Mittel für Eigenentwicklungen beschränkt sind. Gerade generische Tools haben der Definition nach oft deutlichen Spielraum, von dem ressourcenarme Projekte profitieren können. Es lohnt sich, diesen Spielraum nach Evaluierung des eigenen Bedarfs zu erkunden bzw. die eigene Problemstellung im Rahmen der Möglichkeiten bestehender Lösungen zu verstehen, sofern dies die eigene Arbeit nicht an kritischen Stellen einschränkt. Ein wichtiges Mittel, um Lösungen, wie die hier präsentierte, einer breiteren Gruppe an Forscher:innen zu ermöglichen, sind neben der Veröffentlichung auf einer Plattform wie GitHub auch die nachhaltige Speicherung der Software[3] und die Verknüpfung dieser Veröffentlichung mit dem publizierten Korpus. Wenn eine bestehende Software erweitert wird, bietet diese idealerweise auch eine Auflistung bekannter Erweiterungen, in der man die eigene Lösung registrieren kann.

5 Ausblick

5.1 Weiterverwendung des Manipulators

Das beschriebene Alignierungs-Modul wandelt Alignierungsannotationen innerhalb eines Dokumentes zwischen verschiedenen Texten um. Im Kontext der kritisierten starren Verzahnung der Konzepte Dokument, Datei und Text wäre ein logischer nächster Entwicklungsschritt die Alignierung zweier separater Dokumente bzw. Dateien zu einem Ausgabedokument. Für eine solche Anwendung gibt es bereits konkreten Bedarf: Das RUEG-Korpus (Wiese et al. 2020) vereint verschiedene Arten von Annotationen, die den Einsatz separater Annotationstools erfordern. Die resultierenden Dokumente können in diesem Fall nur per Alignierung der Basistexte zu einem gemeinsamen Dokument vereinigt werden. Zu diesem Zwecke existiert bereits ein Prototyp, der die Token zweier Dokumente in einem Dokument kombiniert und ihrer Reihenfolge nach miteinander aligniert. Eine Erweiterung auf annotationsbasierte Alignierung, wie hier vorgestellt, befindet sich in der Umsetzung.

5.2 Einsatz erweiterbarer Annotationstools

Der hier vorgestellte Ansatz hat sich bei der Erstellung des WiN-Korpus bewährt und die Verknüpfung von Token-, Spannen- und Alignierungsannotation ermöglicht. Wenn weitere Arten von Annotationen wie zum Beispiel Dependenzen oder

[3] Dafür bieten sich mehrere Plattformen wie zum Beispiel Zenodo (https://zenodo.org/), OSF (https://osf.io/) oder Software Heritage (https://www.softwareheritage.org/) an, die teilweise auch persistente IDs für Publikationen ermöglichen.

Konstituentenbäume zum Korpus hinzugefügt werden sollten, wäre dies im Moment durch die Beschränkungen von EXMARaLDA auf Spannenannotationen nicht möglich. Man könnte analog zu den Alignierungskanten Wege finden, diese in EXMARaLDA darzustellen, aber mit zunehmender Anzahl an Annotationskonzepten wird auch die Komplexität für die Annotator:innen höher. Annotationstools wie Hexatomic (Druskat/Krause 2021), die die Erstellung verschiedener Arten von Annotationen in Mehrebenenkorpora erlauben, sind eine mögliche Lösung. Aber auch bei solchen flexibleren Annotationstools wird es den Bedarf geben, Bestandsdaten zu konvertieren oder neue Arten von Annotationen umzusetzen. Hexatomic erlaubt zwar durch eine Plugin-Architektur die Erweiterung um neue graphische Annotationseditoren, der Aufwand zur technischen Umsetzung ist aber gerade im Vergleich zu der hier vorgestellten Lösung recht groß und lohnt sich unter Umständen erst bei etablierten Annotationskonzepten.

5.3 Konvertierung über Pepper in Python

Die maschinelle Verarbeitung und Aufbereitung linguistischer Daten gestaltet sich oft effizienter und flexibler in Python als in Java. Dies hat mehrere Ursachen: Zum einen ist es nach unserer Erfahrung eher unüblich, Skripte für die Verarbeitung eines einzelnen Korpus in Java zu schreiben. Python ist Forscher:innen oft zugänglicher, wie die dominante Verwendung in natürlicher Sprachverarbeitung sowie die vielen Frameworks in diesem Bereich zeigen. Der Fokus der Entwicklung von Python liegt anders als bei Java auf Niedrigschwelligkeit für unerfahrene Programmierer:innen und Lesbarkeit des Codes (Python Software Foundation 2021). Bisher können Pepper-Module nur in Java implementiert werden: Eine Schnittstelle zur Erstellung von Pepper-Modulen in Python wäre aber aufgrund der genannten Vorteile wünschenswert. Damit würde auch eine breitere Nachnutzung von Pepper und seinen bereits bestehenden Modulen für weniger erfahrene Nutzer:innen ermöglicht. Eine Einbindung von Pepper in bestehende Python-Skripte könnte zudem die Effizienz und Robustheit in der Korpusaufbereitung erhöhen, da auf eine Schnittstelle zwischen Python und Java in der eigenen Verarbeitungspipeline verzichtet werden kann.

5.4 Einbindung automatischer Alignierung

Statt bei der Alignierung auf manuell erstellte Labels oder auf die Tokenreihenfolge zurückzugreifen, ließen sich auch im Manipulationsprozess des Alignierungs-Moduls die Zugehörigkeiten zwischen Token verschiedener Texte automatisch detektieren. Entsprechende Modelle könnten beispielsweise statistisch oder regelbasiert funktionieren und über ein einfaches Interface in das Alignierungs-Modul eingebunden werden. Die im vorherigen Abschnitt beschriebene Python-Schnittstelle würde einen solchen Prozess erheblich vereinfachen, da zumindest der statistischen Modellierung in Python kaum Grenzen gesetzt sind und viele frei zugängliche Frameworks zur Verfügung stehen (Abadi et al. 2016, Kramer 2016).

Förderung

Die Forschungsergebnisse dieser Veröffentlichung wurden gefördert durch die Deutsche Forschungsgemeinschaft (DFG) – SFB 1412, 416591334 sowie FOR 2537, 313607803, GZ LU 856/16-1.

Literaturverzeichnis

Abadi, Martín/Barham, Paul/Chen, Jianmin/Chen, Zhifeng/Davis, Andy/Dean, Jeffrey/Devin, Matthieu/Ghemawat, Sanjay/Irving, Geoffrey/Isard, Michael/Kudlur, Manjunath/Levenberg, Josh/Monga, Rajat/Moore, Sherry/Murray, Derek G./Steiner, Benoit/Tucker, Paul/Vasudevan, Vijay/Warden, Pete/Wicke, Martin/Yu, Yuan/Zheng, Xiaoqiang: TensorFlow: A System for Large-Scale Machine Learning. In: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). Savannah 2016, S. 265-283. https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi.

Anthony, Laurence: AntConc (Version 3.5.9). 2020. https://www.laurenceanthony.net/software.

Coniglio, Marco/De Bastiani, Chiara/Schaffert, Jan Christian/Recker, Anabel/Sahm, Heike: Transkriptions- und Annotationshandbuch für das Pro*Niedersachsen-Projekt ‚WiN – Wiedererzählen im Norden. Digitale Analyse weltlicher Erzählungen in niederdeutschen Inkunabeldrucken'. Version 1.0. DARIAH-DE. 2019. https://doi.org/10.20375/0000-000C-35E3-8.

Druskat, Stephan/Krause, Thomas: Hexatomic. 2021. https://hexatomic.github.io/.

Gast, Volker/Bierkandt, Lennart/Rzymski, Christoph: Creating and retrieving tense and aspect annotation with GraphAnno, a lightweight tool for multilevel annotation. In: Proceedings of the 11th Joint ACL-ISO Workshop on Interoperable Semantic Annotation (ISA-11). London 2015. https://aclanthology.org/W15-0203.

Guillaume, Bruno: Graph Matching and Graph Rewriting: GREW tools for corpus exploration, maintenance and conversion. In: EACL 2021 – 16th conference of the European Chapter of the Association for Computational Linguistics. Kiew 2021. https://hal.inria.fr/hal-03177701.

Kramer, Oliver: Scikit-Learn. In: ders.: Machine Learning for Evolution Strategies. Cham 2016 (Studies in Big Data 20), S. 45-53. https://doi.org/10.1007/978-3-319-33383-0_5.

Krause, Thomas: ANNIS: A graph-based query system for deeply annotated text corpora. 2019. https://dx.doi.org/10.18452/19659.

Krause, Thomas/Zeldes, Amir: ANNIS3: A new architecture for generic corpus query and visualization. In: Digital Scholarship in the Humanities 31,1 (2016), S. 118-139.

Levy, Roger/Andrew, Galen: Tregex and Tsurgeon: tools for querying and manipulating tree data structures. In: Proceedings of the fifth international conference on Language Resources and Evaluation (LREC). Genua 2006, S. 2231-2234. http://www.lrec-conf.org/proceedings/lrec2006/.

Mann, William C./Thompson, Sandra A.: Rhetorical structure theory: A theory of text organization. 1987.

Odebrecht, Carolin/Belz, Malte/Zeldes, Amir/Lüdeling, Anke/Krause, Thomas: RIDGES Herbology: designing a diachronic multi-layer corpus. In: Language Resources and Evaluation 51,3 (2017), S. 695-725.

Python Software Foundation: The Python Tutorial. 2021. https://docs.python.org/3.9/tutorial/.

Sauer, Simon/Lüdeling, Anke: Flexible Multi-Layer Spoken Dialogue Corpora. In: International Journal of Corpus Linguistics 21,3 (2016), S. 419-438.

Savourel, Yves: TMX 1.4b Specification. 2005. https://www.gala-global.org/tmx-14b.

Schmidt, Thomas: EXMARaLDA and the FOLK tools – two toolsets for transcribing and annotating spoken language. In: Proceedings of the 8th international conference on Language Resources and Evaluation (LREC). Istanbul 2012. http://www.lrec-conf.org/proceedings/lrec2012/pdf/529_Paper.pdf.

Shadrova, Anna: Measuring coselectional constraint in learner corpora: A graph-based approach. 2020. https://dx.doi.org/10.18452/21606.

Wiese, Heike/Alexiadou, Artemis/Allen, Shanley/Bunk, Oliver/Gagarina, Natalia/Iefremenko, Kateryna/Jahns, Esther/Klotz, Martin/Krause, Thomas/Labrenz, Annika/Lüdeling, Anke/Martynova, Maria/Neuhaus, Katrin/Pashkova, Tatiana/Rizou, Vicky/Rosemarie, Tracy/Schroeder, Christoph/Szucsich, Luka/Tsehaye, Wintai/Zerbian, Sabine/Zuban, Yulia: RUEG Corpus. 2020. https://doi.org/10.5281/zenodo.3765218.

Wilkinson, Mark D./Dumontier, Michel/Aalbersberg, Ijsbrand Jan/Appleton, Gabrielle/Axton, Myles/Baak, Arie/Blomberg, Niklas/Boiten, Jan-Willem/Bonino da Silva Santos, Luiz/Bourne, Philip E./Bouwman, Jildau/Brookes, Anthony J./Clark, Tim/Crosas, Mercè/Dillo, Ingrid/Dumon, Olivier/Edmunds, Scott/Evelo, Chris T./Finkers, Richard/Gonzalez-Beltran, Alejandra/Gray, Alasdair J.G./Groth, Paul/Goble, Carole/Grethe, Jeffrey S./Heringa, Jaap/'t Hoen, Peter A. C./Hooft, Rob/Kuhn, Tobias/Kok, Ruben/Kok, Joost/Lusher, Scott J./Martone, Maryann E./Mons, Albert/Packer, Abel L./Persson, Bengt/Rocca-Serra, Philippe/Roos, Marco/van Schaik, Rene/Sansone, Susanna-Assunta/Schultes, Erik/Sengstag, Thierry/Slater, Ted/Strawn, George/Swertz, Morris A./Thompson, Mark/van der Lei, Johan/van Mulligen, Erik/Velterop, Jan/Waagmeester, Andra/Wittenburg, Peter/Wolstencroft, Katherine/Zhao, Jun/Mons, Barend: The FAIR Guiding Principles for scientific data management and stewardship. In: Scientific Data 3,1 (2016). https://doi.org/10.1038/sdata.2016.18.

Zeldes, Amir/Zipser, Florian/Neumann, Arne: PAULA XML Documentation. 2013. https://hal.inria.fr/hal-00783716.

Zipser, Florian/Romary, Laurent: A model oriented approach to the mapping of annotation formats using standards. In: Workshop on Language Resource and Language Technology Standards, LREC 2010. http://hal.archives-ouvertes.fr/inria-00527799/.