DFG-Projekt „Entwicklung der satzinternen Großschreibung im Deutschen“
Leitung:
Prof. Dr. Klaus-Michael Köpcke (Universität Münster)
Prof. Dr. Renata Szczepaniak (Universität Hamburg)
Mitarbeiter:
Fabian Barteld, Marc Schutzeichel
Annotationsteam (Hamburg):
Annemarie Bischoff, Lisa Dücker, Eleonore Schmitt, Annika Vieregge
Annotationsteam (Münster):
Julia Hübner, Johanna Legrum, Katja Politt, Nicholas Wieling
Hilfskraft, Technik:
Nicolai Pudimat

Fabian Barteld
Nicolai Pudimat
Renata Szczepaniak
Annika Vieregge

Manuelle Korpusannotation mit GATE

Einleitung

GATE [3] ist ein Framework zur Entwicklung von NLP-Anwendungen. Unter anderem bringt es mit dem *GATE Developer* aber auch eine IDE mit, die es erlaubt Annotationen anzulegen und zu bearbeiten. Die Annotationen werden als Graphen angelegt, die dem Formalismus *annotation graph* ähnlich sind [2]. Die Datenstruktur ist damit flexibel genug fast beliebige Annotationen abzubilden.
Im DFG-gefördeten Projekt „Entwicklung der satzinternen Großschreibung im Deutschen“ wurden frnhd. Hexenverhörprotokolle mit *PoS*, *Numerus*, *Belebtheit* und *Lemma* annotiert sowie in *syntaktische Einheiten* eingeteilt. Aufgrund von Besonderheiten der Textsorte und der Fragestellung wurde mit doppelter Tokenisierung gearbeitet, wobei beide Tokenisierungen mit Annotationen versehen wurden [1]. Aufgrund seiner Flexibilität bot sich *GATE* als Annotationstool an.

GATE erfüllt außerdem die Anforderungen, die Dipper et al. [5] an Annotationstools stellen:

- es bietet ein XML-basiertes Datenformat
- es wird aktiv entwickelt
- es benötigt keine Installation und ließ sich nach einer kurzen Einarbeitungszeit einfach benutzen
- als Java-Programm läuft es auf allen gängigen Plattformen
- es werden keine Lizenzgebühren fällig

Auf diesem Poster werden die Erfahrungen, die bei der Verwendung von *GATE* für die Annotation von frnhd. Hexenverhörprotokollen gemacht wurden und in diesem Kontext entwickelte Erweiterungen vorgestellt.

Annotieren

Bei den durchgeführten Annotationen lassen sich zwei Arten unterscheiden: *Unitizing* und *Coding*. Beim Unitizing wird der Text in Segmente unterteilt (hier: *Token* und *syntaktische Einheiten*). Beim Coding werden festgelegte Segmente mit Annotationen versehen (hier: *PoS*, *Numerus*, *Belebtheit* und *Lemma* als Annotation für Token). Beide Arten lassen sich mit dem *GATE Developer* durchführen, wobei sich die bei *GATE* enthaltenen Editor-Komponenten als schlecht geeignet für Coding-Aufgaben wie *PoS* herausstellten, was zur Entwicklung eines eigenen Plugin führte.

Unitizing
Der *Document Editor* bietet die Möglichkeit einen Text mit Annotationen zu versehen und vorhandene Annotationen zu bearbeiten. Annotationen sind in *GATE* gerichtete Kanten eines spezifischen Typs, die im Text verankert sind. Im Text werden die Annotationen als farbige Balken über dem Text angezeigt (vgl. Abb. 1). Mit der Maus lassen sich neue Spannen hinzufügen. Über den *Annotation Editor* lassen sich Annotation wieder entfernen bzw. der linke und rechte Rand nachträglich anpassen. Die Anzeige der Annotationstypen lässt sich einzeln aktivieren bzw. deaktivieren, so dass für die aktuelle Aufgabe irrelevante Annotationen ausgeblendet werden können.
Über die Integration von automatischen Annotationstools lässt sich der manuelle Aufwand reduzieren. So mussten für die Tokenisierung hauptsächlich sprachstufenspezifische Korrekturen vorgenommen werden, z.B. bei getrennt geschriebenen Komposita und Worttrennungen ohne Trennzeichen.

Coding
Coding wird im Annotationsmodel von *GATE* über *Features* geregelt, die Annotationen zugeordnet sind. Ein Annotationstyp kann damit z.B. für Token stehen. Den Annotationen dieses Typs werden dann Features wie „pos“, „numerus“, „lemma“ und „belebtheit“ zugeordnet. Die Konsistenz von Annotationen kann über *Annotation Schemas* geschehen, die es erlauben, Features und dafür erlaubte Werte mithilfe von *XML-Schema* festzulegen.
GATE bietet von Haus aus einige Editoren für das Bearbeiten von Annotationen. Allerdings öffnen sich diese beim Überfahren einer Annotation mit dem Mauszeiger jeweils in einem eigenen Fenster. Dadurch ist es sehr aufwändig und fehleranfällig z.B. alle PoS-Features von Token-Annotationen in einem Text manuell zu überprüfen bzw. bearbeiten.
Für diese Art von Annotation geeigneter ist eine tabellarische Ansicht aller Annotationen eines Typs mit der Möglichkeit deren Features und Values zu bearbeiten, wie sie z.B. das Annotationstool *MAE* [7] bietet. Eine solche Funktionalität wurde zu *GATE* mit dem Plugin *Vertical Annotation Editor* hinzugefügt.

Plugin: Vertical Annotation Editor
Das Plugin *Vertical Annotation Editor* erlaubt das Anzeigen von Annotationen eines Typs in einer Tabelle und das Bearbeiten der zugehörigen Features. Die angezeigten Features und die möglichen Werte werden durch das geladene Annotationsschema bestimmt, wodurch sich der Editor flexibel an die durchgeführte Annotationsaufgabe anpassen lässt. Die Liste der Annotationen lässt sich filtern, so dass z.B. nur mit „pos=NN“ ausgezeichnete Token angezeigt werden. Sieht das Schema Werte aus einer Liste für die Annotation vor, wird dies per ComboBox mit Autovervollständigung realisiert. Dies ermöglicht die einfache Handhabung von Codingtasks mit einer großen Anzahl an Tags. Zusätzlich wird die Auswahl in der Tabelle mit der Position des Cursors im Text synchronisiert. Dies erlaubt das einfache Wechseln zwischen Tabelle und Text sowie die Anzeige von Kontext und weiterer Annotationen im *Annotation Stack*.

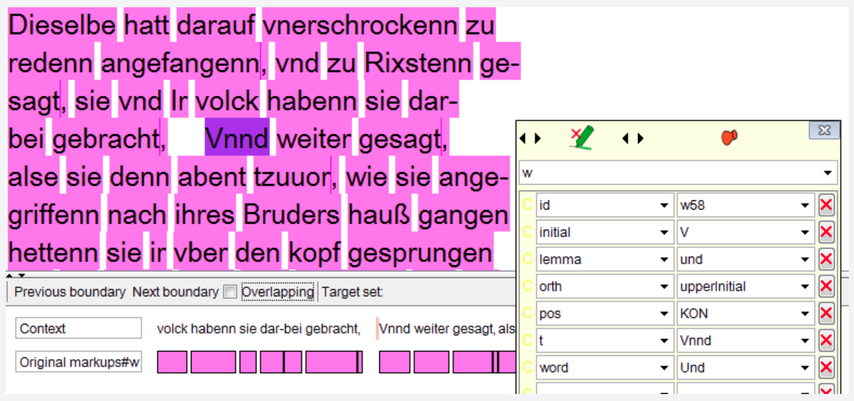


Abbildung 1: Document Editor mit angezeigtem Annotation Editor

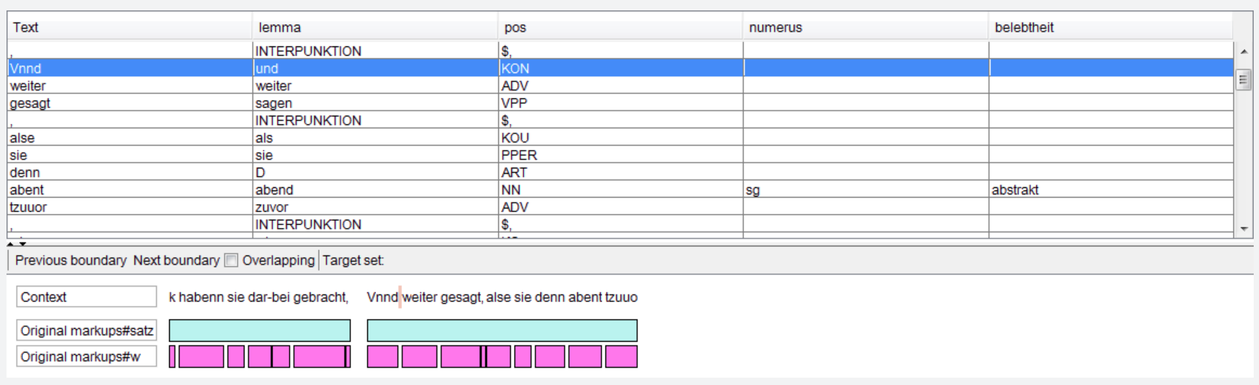
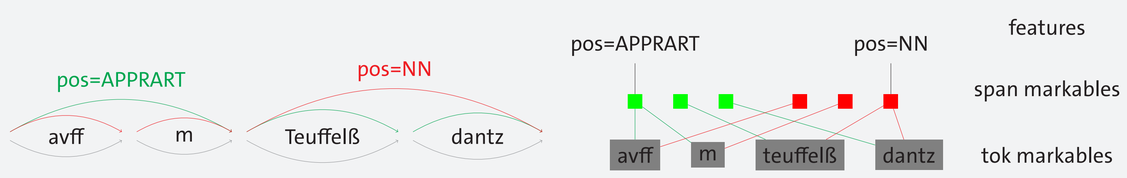


Abbildung 2: Vertical Annotation Editor mit angezeigtem Annotation Stack

Export der Daten

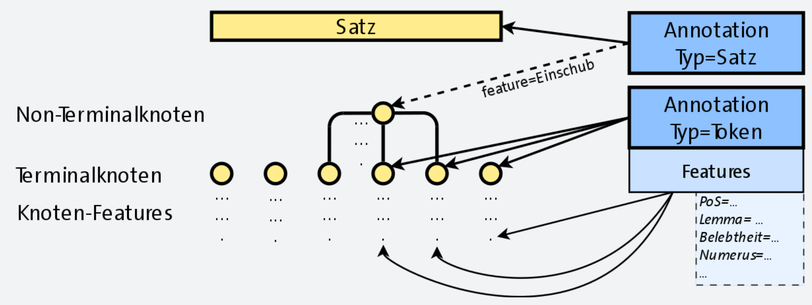
Für die Auswertung der Korpusdaten und evtl. die weitere Annotation mit spezialisierten Annotationstools ist die Möglichkeit des Exports in unterschiedliche Formate relevant. Für unser Projekt waren dies die Formate *PAULA* [4] (als Basis für den Import in *ANNIS* [8]) und *TIGER-XML* (als Basis für die Erstellung weiterer Annotationsebenen). Da *GATE* diese Export-Möglichkeiten nicht mitbringt, wurden im Projekt hierfür Plugins in Form von Groovy-Skripten geschaffen.

PAULA-Export
In *GATE* werden Dokumente als Text und einem dazugehörigen Graphen, der die Annotationen darstellt, modelliert. Die Kanten in dem Graph stehen für einzelne Annotationen, die an beliebigen Stellen im Text verankert werden. *GATE* hat daher keine Tokenisierung im Sinne einer minimalen Segmentierung des Textes. *PAULA* setzt eine solche Segmentierung allerdings voraus. Für den Export wird die Tokenisierung aus den vorhanden Annotationen erstellt, indem eine kleinste Segmentierung erstellt wird, aus der die Annotation zusammengesetzt werden können.



Die Struktur des Korpus und der Annotationen bleibt beim Export erhalten, d.h. *GATE*-Dokumente werden auch auf einzelne Dokument in *PAULA* abgebildet. Metadaten in Form von Features zu Dokumenten und zum Korpus werden ebenfalls genauso übernommen. Annotation Sets werden zu Namensräumen in *PAULA*.

TIGER-Export
Im *TIGER*-Format werden Texte als Abfolge von Sätzen dargestellt, die wiederum aus Token bestehen. Für den Export sind daher zwei Annotationstypen notwendig, die diese Segmentierungen darstellen. Das Plugin für den Export benötigt zwei entsprechende Annotationstypen. Annotationen vom Typ „Satz“ werden dann auf Sätze im *TIGER*-Modell abgebildet. Alle Annotationen vom Typ „Token“, die sich zwischen den Kanten einer als Satz annotierten Textspanne befinden, werden als *Terminalknoten* dieses Satzes exportiert, wobei die Features erhalten bleiben.



Eine besondere Behandlung erfahren Sätze mit dem Feature „Einschub“. Diese werden mit dem vorhergenden und nachfolgenden Satz kombiniert und mittels Non-Terminalknoten als eingeschobener Satz markiert.

Fazit

GATE bietet ein flexibles Annotationsformat, das beliebige Kombinationen von Unitizing- und Coding-Annotationen abbilden kann. *GATE Developer* bietet eine gut zu bedienende Oberfläche für Unitizing-Aufgaben. Mit dem vorgestellten Plugin *Vertical Annotation Editor* lassen sich zudem auch umfangreiche Coding-Aufgaben einfach bearbeiten.
Durch die Nutzung von *GATE Developer* als Annotationstool lassen sich Tools zur automatischen Annotation einfach in den Workflow einbinden, so z.B. die von Scheible et al. erstellte Pipeline für frnhd. Text [6]. Da auch die Integration von *Groovy*-Skripten möglich ist, lassen sich einfach eigene Automatisierungen erstellen.

Für die Erstellung von Annotationstypen, die *GATE* nicht (gut) unterstützt, z.B. Syntaxbäume, und zur späteren Auswertung, ist es hilfreich die Daten in andere Formate überführen zu können. Im Projekt wurden zwei Exportmöglichkeiten für GATE entwickelt: Ein Exporter für das *PAULA*-Format, wodurch sich u.a. die Analyse des Korpus in *ANNIS* erschließt und ein Exporter ins Format *TIGER-XML*, was die Anreicherung der Texte mit syntaktischen Annotationen ermöglicht.

Der *Vertical Annotation Editor* und der Exporter für *PAULA XML* finden sich unter `http://github.com/fab-bar`.

Literatur

[1] Fabian Barteld, Renata Szczepaniak und Heike Zinsmeister. 2014.
The definition of tokens in relation to words and annotation tasks.
Proceedings of the Thirteenth International Workshop on Treebanks and Linguistic Theories (TLT 13), 250–258, Tübingen, Germany.

[2] Steven Bird und Mark Liberman. 2001.
A formal framework for linguistic annotation.
Speech and Communication 33, 1-2, 23–60.

[3] Hamish Cunningham et al. 2011.
Text Processing with GATE (Version 6).
University of Sheffield. Department of Computer Science.

[4] Stefanie Dipper, Michael Götze, Uwe Küssner und Manfred Stede. 2005.
Representing and querying standoff XML.
G. Rehm, A. Witt und L. Lemnitzer (Hrsg.): Datenstrukturen für linguistische Ressourcen und ihre Anwendungen. Proceedings of the Biennial GLDV Conference 2007, 337–346. Tübingen.

[5] Stefanie Dipper, Michael Götze und Manfred Stede. 2004.
Simple annotation tools for complex annotation tasks: an evaluation.
Proceedings of the LREC Workshop on XML-based richly annotated corpora, 54–62, Lisbon, Portugal.

[6] Silke Scheible, Richard J. Whitt, Martin Durrell und Paul Bennett. 2012.
GATEtoGerManC: A GATE-based annotation pipeline for historical German.
Proceedings of the Eighth International Language Resources and Evaluation Conference (LREC 2012), 3611–3617, Istanbul, Turkey.

[7] Amber Stubbs. 2011.
MAE and MAI: Lightweight annotation and adjudication tools.
Proceedings of the 5th Linguistic Annotation Workshop (LAW V), 129–133, Portland, Oregon.

[8] Amir Zeldes, Julia Ritz, Anke Lüdeling und Christian Chiarcos. 2009.
ANNIS: A search tool for multi-layer annotated corpora.
Proceedings of Corpus Linguistics 2009, Liverpool, UK.