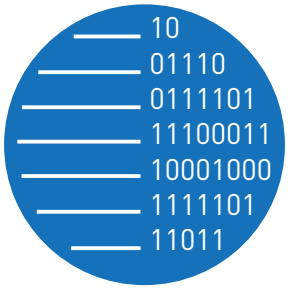

MAGAZIN FÜR DIGITALE EDITIONSWISSENSCHAFTEN

Herausgegeben vom Interdisziplinären Zentrum
für Editionswissenschaften
der Friedrich-Alexander-Universität Erlangen-Nürnberg

1 / 2015

FAU UNIVERSITY PRESS

FAU University Press
Magazin für digitale Editionswissenschaften
ISSN 2364-0855

Herausgeber:
Interdisziplinäres Zentrum für Editionswissenschaften
Prof. Dr. Boris Dreyer (Sprecher)
Universität Erlangen-Nürnberg
Department Geschichte
Alte Geschichte
Kochstr. 4, Postfach 8
D-91054 Erlangen

PSEUDO-MARKUP:
EINE ESELSBRÜCKE ZWISCHEN MANUELLER UND MASCHINELLER TEXTVERARBEITUNG

Florian Kragl

In jüngerer Zeit hat sich XML als das Mittel der Wahl durchgesetzt, wenn es darum geht, Texte gleich welcher Herkunft professionell elektronisch aufzubereiten. Sind die Texte erst einmal XML-konform gespeichert und mehr oder weniger intensiv mit Metadaten angereichert, erweist sich das Format tatsächlich als ein zuvor ungekannter Segen: Die Verbalisierung von Texteigenschaften auf allen erdenklichen Ebenen – Segmentierung, Lemmatisierung, Kommentierung, was immer – erlaubt ein rasches und präzises Navigieren durch das Korpus und dessen automatisierte Weiterverarbeitung, z. B. für linguistische oder stilistische Analysen. Wer sich allerdings je mit der Aufgabe konfrontiert sah, Textinformation in XML einzutragen, weiß, dass dieser Benutzersegen um einen Bearbeiterfluch erkauft ist. Was nämlich benutzerseitig von unschätzbarem Vorteil ist – die extensive ›Markierung‹ sämtlicher relevanter bzw. relevant erscheinender Texteigenschaften –, erweist sich bei der Dateneingabe als umständlich, zeitraubend und fehlerträchtig. Handelsübliche XML-Editoren mögen da noch eine gewisse Unterstützung bieten, weil sie einem wenigstens das Abtippen der verwinkelten Klammerstrukturen ersparen; eine rechte Hilfe sind sie aber auch nicht, gerade wenn ein Text komplex annotiert wird, weil dann aus dem Klammertippen eben ein Herumgeklicke wird, dessen Frustrationsgrad bestenfalls knapp unter dem der *plain text*-Eingabe liegt. Dieser Befund wäre im Übrigen auch projekthistorisch abzusichern: Es gibt kaum ein größeres XML-Korpus, das von Hand eingegeben worden ist; üblicherweise werden große Textmassen via Scan/OCR digitalisiert und dann erst sekundär, halbautomatisch oder von Hand, in XML weiterverarbeitet (die TextGrid-Korpora wären dafür im deutschsprachigen Raum der beste Beleg; `https://www.textgrid.de/`). Man entgeht damit der skizzierten Frustrationserfahrung; Preis dafür ist aber wiederum das Arbeiten mit ›sekundären‹ Texten, die – wenn es sich um Editionen handelt – eben *nicht* neu aus den Quellen gewonnen sind.

All dies ist intrinsische Konsequenz der XML-Struktur und betrifft ein jedes textbezogene Projekt, das sich für dieses Format entscheidet. Es gibt allerdings Bereiche, wo die damit verbundenen arbeitspragmatischen Schwierigkeiten besonders virulent werden. Dies ist immer dann der Fall, wenn (1) die XML-Code-Struktur eine Komplexität erreicht, die weit über die schiere Linearität eines Prosatextes hinausgeht, und/oder wenn (2) kei-

ne Textquellen vorliegen, die automatisch eingelesen und dann als Basis für die weitere Arbeit teilautomatisch verwendet werden können. XML-Editionsprojekte, die sich Texten aus dem Zeitalter vor der Gutenberg-Galaxis widmen, können mit beidem aufwarten: Sie gewinnen ihre Texte, gerade weil sie sich oft von alten, in die Kritik gekommenen Editionspraktiken des langen 19. Jahrhunderts absetzen wollen, neu aus Handschriften, aus Quellen also, denen mit OCR zumindest aktuell noch nicht so recht beizukommen ist: die schlicht abgeschrieben werden müssen. Und sie können aber zugleich – anders als etwa ein Editionsprojekt zu Texten der Goethe-Zeit – nicht damit Halt machen, die aus den Quellen gewonnene Textschicht im Grunde so, wie sie ist, zu präsentieren. Die Textspeicherung in mittelalterlichen oder antiken Codices ist der uns gewohnten dermaßen fremd, dass die Texte für den allergrößten Teil auch das fachnahen Publikums unbrauchbar wären, würde man nicht unablässig in sie eingreifen, um sie modernen Leseaugen erträglich zu machen: Abkürzungen müssen aufgelöst, die alte, nicht-syntaktische Interpunktion durch eine moderne ersetzt, Zeilenfall und (bei gebundener Sprache) Versgrenzen abgebildet, Korruptes verbessert (das *proof reading* ist bei Handschriften weniger streng als beim modernen Buchdruck), z. T. auch phonetisch-phonologische Disambiguierungen und Normierungen vorgenommen werden. Egal ob es nun altnordische Saga-Literatur, mittelhochdeutscher Minnesang, die gotische Wulfila-Bibel ist, die Schritte sind immer dieselben, und auf Tagungen und Kongressen wird allenthalben darüber gestöhnt, dass von modernen Projektmitarbeitern erwartet wird, was vernünftiger Weise nur von einem Computer zu fordern wäre: fließend und fehlerfrei XML, meist in der Species der TEI, schreiben und lesen zu können. Erschwert wird die Lage davon, dass die betroffenen Fächer allesamt mehr oder weniger exotisch sind, zu klein jedenfalls, um jene Geldmengen freizusetzen, deren es bedürfte, um Tools zu entwickeln, die einem wenigstens einen Teil dieser frustrierenden Handarbeit abnähmen und die Mitarbeiter aus dem *purgatorium parenthesum* erlöste (um von Handschriften-OCR gar nicht erst zu träumen).

Es war genau dieses Problem, mit dem wir uns konfrontiert sahen, als wir zu dritt – Sonja Glauch und ich in Erlangen, Manuel Braun in Stuttgart – daran machten, eine Online-Edition der mittelhochdeutschen Lyrik (Leich, Minnesang, Sangspruch) zu konzipieren. Unsere Textbasis sind Handschriften überwiegend des spätesten 13. und des 14. Jahrhunderts; diese gilt es zu transkribieren; Abkürzungen müssen aufgelöst werden; die handschriftliche Interpunktion, die meist die Strophenform anzeigt (Reimpunkte), muss dokumentiert und im Editionsprozess durch eine neuzeitliche ersetzt, umgekehrt die Strophenform durch Zeilenfall abgebildet werden; offensichtliche Fehler gehören gebessert; und schließlich wollen wir – aus Traditionsgründen, aber auch aus solchen der Didaktik – für jene Texte, für die sich dies aus sprachhistorischen und stilistischen Gründen anbietet, ›normalisierte‹ Lesefassungen herstellen, die sich von der mitunter stark dialektal gefärbten Schreibsprache der Handschriften ein Stück weit entfernen und einen

Zeichen- und Phonemsatz verwenden, den das 19. Jahrhundert als ›mittelhochdeutsche Dichtersprache‹ erfunden hat.

Dass wir unsere Daten in XML/TEI ablegen und auch anderen zur Verfügung stellen möchten, war von Anfang an klar; die konzeptuelle Aufgabe bestand darin, sie dorthin zu bringen. Wenn wir im Folgenden unseren Lösungsansatz beschreiben – dessen Idee wir im Kern Christian Aistleitner (Linz) verdanken –, so tun wir dies nicht in der Überzeugung, damit den Königsweg für das oben skizzierte Problem gefunden zu haben. Es geht uns vielmehr darum, eine Variante vorzustellen, mit der – wie wir gesehen haben – sich einigermaßen pragmatisch arbeiten lässt, und im selben Zuge zu einem Dialog über all diese kleinen Hürden anzuregen, über die nicht gerne offen gesprochen wird und mit deren Überwindung wir doch alle irgendwie auf je eigne Art kämpfen. Zu hoffen stünde, dass die Räder dann nicht immer neu erfunden werden müssten und Synergieeffekte auch jenseits der Antragsprosa tatsächlich effizient sein könnten. Da sich, soweit wir sehen, zumindest so gut wie alle Mittelalterphilologien, wenn sie sich in die Digitalität vorwagen, auf denselben Baustellen tummeln, möchte dies doch mehr sein als nur ein frommer Wunsch.

Unser erstes Prinzip ist noch keines der Dateneingabe, sondern der Datenstruktur: Wir projizieren sämtliche Textschichten, die sich aus der Art unseres Editionsprojekts ergeben, in die Linearität der Zeile. Wenn unsere Texte also beispielsweise sowohl als ›rohe‹ Transkription als auch als ›normalisierte‹ Edition vorliegen, so stehen hinter diesen beiden Ideen nicht separat abgelegte Zeichensequenzen, sondern beide werden aus ein und demselben Zeichenfluss errechnet. Dass dieser eklatant überdeterminiert ist und die Sequenzialität der Zeichenkette massiv darunter leidet, dass sie beständig und punktuell – für ein Wort, eine Silbe, häufig nur für einen Buchstaben – um zusätzliche Dimensionen erweitert wird, liegt auf der Hand; der Vorteil dieses Vorgehens aber scheint uns doch zu überwiegen, weil wir so nie Gefahr laufen, dass – etwa in Korrekturgängen – unsere Textschichten asynchron werden. Wie gesagt, dies betrifft noch nicht die Dateneingabe, sondern den Aufbau der Textdatensätze (die in unserem Projekt im Grunde Strophenbausteine sind). Und natürlich geschehen diese Vorstöße in die nächste Textschicht datentechnisch nur dort, wo die Textschichten tatsächlich auch differieren; wo ein und derselbe Buchstabe in allen Textschichten konstant ist, muss dies nicht eigens vermerkt werden.

Mit einem Beispiel: Handschriftliches *clage* wird bei uns als solches transkribiert, und es bleibt auch in der nicht ›normalisierten‹ Edition als *clage* stehen. In der ›normalisierten‹ Variante der Edition hingegen setzen wir *klage*, weil dort anlautend *c* zu *k* wird; es handelt sich um eine reine Schreibkonvention. In XML/TEI sähe dies folgendermaßen aus:

```
…
        <choice>
                <orig>
                        c
                </orig>
                <reg type=”n2”>
                        k
                </reg>
        </choice>
        lage
…
```

Wobei `<reg type=“n2“>` anzeigt, dass es sich hier um eine Normalisierung handelt (und nicht etwa um eine Textbesserung, also eine Konjektur). Man könnte auch sagen, dass wir unsere Datensätze auf diese Weise schlankestmöglich halten.

Unser zweites Prinzip beruht ebenfalls auf Effizienz, die nun aber nicht länger eine der Datenstruktur, sondern der Dateneingabe ist. Wieder aber versuchen wir, mit möglichst geringem Aufwand – heißt: mit möglichst wenigen Tastendrucken – maximale Informationsdichte zu erzeugen bzw. zu transportieren, indem mehrschichtige Informationen durch ein elaboriertes Kürzelsystem verdichtet werden. Dieses Kürzelsystem, das man als ein Sammelsurium von Tricks verstehen könnte, ließe sich am besten Pseudo-Markup nennen, und es besteht seinerseits aus einer Reihe von Komponenten, die teils auf die Zeichensatzdifferenz zwischen Handschrift und Tastatur, teils auf die Mehrdimensionalität unserer Textdatensätze reagiert:

Erstens, wofür die Maschinentastatur keine Zeichen bereitstellt, verwenden wir Kurzschreibweisen. Beispiel könnte die *a*-*e*-Ligatur sein (*æ*), die in vielen Handschriften verwendet wird, um langes umgelautetes *a* darzustellen. Wir transkribieren sie mit `#ae`. Als Operation mag dies hart an der Banalität sein; es ermöglicht es uns aber, unsere gesamte Textarbeit in *plain text* durchzuführen, weil wir für die Dateneingabe kein Zeichen benötigen, das nicht Teil des Standard-ASCII-Satzes wäre.

Zweitens, wo immer es möglich ist, komprimieren wir mehrere Textschichten in einen Eingabecode. Dies tritt besonders häufig bei Abbreviaturen auf, die in den deutschen Handschriften des Mittelalters zwar nicht so vielfältig sind wie in den lateinischen, nichtsdestotrotz aber häufig auftreten und einen nicht unerheblichen Teil des Textbestandes ausmachen. Eine der häufigsten Abbreviaturen ist der Nasalstrich, der über Vokal steht und anzeigt, dass auf diesen ein Nasal, also *m* oder *n* folgt. Wir codieren ihn, indem wir den (in der Handschrift fehlenden) Nasal ausschreiben, vor diesen aber `#` setzen. Das heißt, dass ein *e* mit Nasalstrich sowohl als `e#m` als auch als `e#n` transkribiert werden kann, je nachdem, wessen Platz der Nasalstrich vertritt. Der Textprozessor, der aus unseren Daten dann etwa eine Transkriptions- oder eine Editionsfassung (oder aber XML/

TEI-konforme Daten) herstellt, weiß, dass in beiden Fällen in der ›rohen‹ Transkription ein Strich über dem *e* zu stehen hat, in der Edition aber *em* bzw. *en*. Es ist diese Art der Komprimierung, die am Wesentlichsten dazu beiträgt, dass unsere Texteingabe zügig und – wenn man die Codes erst einmal internalisiert hat – auch einigermaßen komfortabel geschehen kann. Im Übrigen muss man nicht ein Lexikon von selbsterdachten Codes auswendig lernen, um an unserem Projekt teilzuhaben: Die Liste umfasst aktuell nicht einmal eine A4-Seite. Das liegt nicht an der Einfalt unserer Texte, sondern daran, dass wir dieses Kompressionsverfahren nur bei häufigen Textphänomenen nutzen; alles andere wäre unökonomisch.

Drittens, wo Mehrschichtigkeit nicht über komprimierte Eingaben dieser Art bewältigt wird, arbeiten wir mit abgekürzter XML/TEI-Syntax. Während es dort eine ganze Reihe von Klammern braucht, um die schlichte Information abzuspeichern, dass handschriftliches *c* in der ›normalisierten‹ Edition als *k* darzustellen ist, setzen wir:

```
{c|k}lage
```

Die geschweiften Klammern definieren den Typ der Textoperation: den einer ›Normalisierung‹, der senkrechte Strich trennt die handschriftliche Lesung links von der hergestellten rechts. Würden wir in den Text eingreifen, um eine verderbte Stelle zu bessern, geschähe dies nach demselben Muster, nur dass der senkrechte Strich nun nicht von Klammern flankiert würde, sondern von Unterstrichen. Wesentlich dabei ist, dass diese Strukturen – wie bei XML/TEI – ineinander verschachtelt sein können, etwa:

```
_ic|ich {c|k}_lage
```

Hier wird handschriftliches *iclage* zuerst zu *ich clage* gebessert, auf nächsthöherer Ebene das *c* aus *clage* in der bekannten Weise zu *k* ›normalisiert‹. In XML/TEI stünde dafür:

```
…
        <choice>
                <sic>
                        ic
                </sic>
                <corr>
                        ich
                        <choice>
                                <orig>
                                        c
                                </orig>
                                <reg type=”n2”>
                                        k
                                </reg>
                        </choice>
                </corr>
        </choice>
        lage
…
```

Hier wird in beiden Fällen – im Pseudo-Markup und in XML/TEI – mit präzise denselben Strukturen gearbeitet; Verschlankung ist hier alleine Resultat einer Verkürzung der XML-Klammernstruktur. Dass diese arbeitsökonomisch durchaus wesentlich ist, ist selbstevident.

Wichtig ist, dass das Pseudo-Markup und XML/TEI vor- und rückwärtskompatibel sind, jede Strophe also, die bereits in XML/TEI abgespeichert ist, für die Bearbeitung wieder in das Pseudo-Markup zurückverwandelt werden kann. Allerdings werden hier dann doch auch Grenzen sichtbar: In dem Moment, wo die XML-Daten mit weiteren Metainformationen – etwa solchen zur metrischen Gestalt oder einer vollständigen Lemmatisierung des Wortmaterials – angereichert wären, würde der Rücktransfer nur noch mit einem erhöhten Programmieraufwand gelingen und bliebe wohl selbst dann erheblich fehleranfällig. Was oben beschrieben ist, ist also tatsächlich primär ein Mittel der Textdatenerfassung; ob und inwieweit es später auch für Korrekturgänge taugt, wird erst noch zu prüfen sein. Seinen primären Zweck erfüllt es allerdings inzwischen schon seit etwa zwei Jahren gut, und dass wir und unsere Kooperationspartner in diesem Zeitraum und zum größeren Teil sogar ohne Drittmittelfinanzierung (diese läuft erst seit einem knappen Jahr) bereits über 2000 Strophen auf diese Weise transkribieren konnten, spricht dafür, dass so nicht nur Zeit, sondern auch Geld gespart wird.

Klar ist auch, dass dieses System keinen Universalschlüssel für Editionen ›alter‹ (vorgutenbergscher) Texte bereitstellt. Es ist zugeschnitten auf eine ganz bestimmte textuelle Situation, hier auf die deutsche Lyrik des 12. und 13. Jahrhunderts, Texte in Strophenhäppchen, in einer ganz bestimmten Sprache, überliefert mit ganz bestimmten Zeichensätzen, in sehr charakteristischen Überlieferungszusammenhängen. Im intensiven Austausch mit Kollegen benachbarter Disziplinen ist uns aber bewusst geworden, dass es sehr leicht anpassbar wäre zumindest an die Anforderungen anderer mittelalterlicher Überlieferungszusammenhänge. Heißt: Eine altnordische Saga wird andere Codes benötigen, andere Formen der Textgliederung (Prosa), ein anderes Normierungs- und Normalisierungssystem nutzen. Im Prinzip aber muss auch deren Edition auf ein mehrschichtiges Konzept – von der ›rohen‹ Transkription bis hin zum normierten oder ›normalisierten‹ Editionstext – bauen, tut dies in aller Regel auch, und darum könnte auch sie dasselbe Gerüstsystem nutzen, auch wenn die Oberflächen je nach den konkreten Gegebenheiten zu gestalten wären. Ob auch der digitalen Edition antiker Texte damit geholfen wäre, müsste man sehen; neuzeitliche Überlieferungen scheinen anders zu funktionieren, entweder stärker drucklastig, was einen Gutteil der hier beschriebenen Operationen obsolet macht, oder aber ganz strikte auf den Autograph und seine Bearbeitungsstufen konzentriert, was ebenfalls ein mehrschichtiges Editionssystem, aber eines von einer ganz anderen Art einfordert. Universal ist also der Schlüssel schon darum nicht, weil er auf handschriftliche, vielleicht noch enger: weil er auf mittelalterliche Textüberlieferung hin geschliffen ist. Wenn es mit seiner Hilfe aber gelänge, diese leichter auf- bzw. neu und

datentechnisch ambitioniert zu verschlüsseln, wäre immerhin für diesen ›mittleren‹ Bereich etwas gewonnen.