

Historische Textnormalisierung: Herausforderungen und Potentiale von Deep Learning

Ehrmanntraut, Anton

anton.ehrmanntraut@uni-wuerzburg.de
Julius-Maximilians-Universität Würzburg, Deutschland
ORCID: 0000-0001-6677-586X

Bracke, Yannic

yannic.bracke@bbaw.de
Berlin-Brandenburgische Akademie der Wissenschaften, Deutschland
ORCID: 0000-0002-7932-8230

Einleitung

DH-Forschende, die mit historischen Dokumenten arbeiten wollen, haben ein Problem: je älter die historischen Texte sind, desto mehr weichen sie in ihrer Schreibweise von der gegenwärtigen Standardsprache ab. Neben dieser diachronischen Veränderung kommt noch eine synchronische Variabilität hinzu; in historischen Korpora finden sich häufig mehrere Schreibvarianten. So beobachten wir in Dokumenten des Deutschen Textarchivs (DTA) zwischen 1830 und 1890 bspw. die Varianten *heirathen*, *heiraten*, *heurathen* und *heyrathen*.

Aus dieser orthographischen Abweichung ergeben sich nun für eine Nutzung digitalisierter historischer Texte Hindernisse: zum einen ist keine geeignete Volltextsuche mehr möglich, denn man müsste nun alle Schreibvarianten kennen und angeben. Zum anderen wird dabei die Weiterverarbeitung mithilfe von NLP-Tools erschwert: *off-the-shelf* NLP-Werkzeuge für POS-Tagging oder Named Entity Recognition wurden auf gegenwartssprachlichen Daten trainiert und liefern für die historischen Varianten schlechtere Ergebnisse (vgl. Scheible et al., 2011; Pettersson et al., 2013; Kogkitsidou und Gambette, 2020). Das limitiert insbesondere *Distant-Reading*-Ansätze auf historischen Korpora.

Eine Möglichkeit, diesen beiden Problemen zu begegnen, ist eine automatisierte historische *Textnormalisierung*. Vereinfacht gesprochen wird dabei das historische Dokument automatisch in die moderne Standardschreibung „übersetzt“, wodurch z. B. sämtliche o. g. Varianten von *heiraten* mit dem standardsprachlichen Äquivalent ersetzt werden. Das wird in vielen modernen Editionen älterer

Werke manuell praktiziert. Eine automatische Textnormalisierung für die deutsche Sprache ab ca. 1600 bietet das Tool *Cascaded Analysis Broker*¹ (CAB, Jurish, 2012) an, das u. a. für die Textnormalisierung der DTA-Korpora eingesetzt wird und als das *de-facto* Standard-Tool fürs Deutsche gelten kann. CAB wendet zur Normalisierung mittels endlicher Automaten bestimmte Ersetzungsregeln von Buchstaben an, die mit linguistischem Expertenwissen händisch entwickelt wurden. Damit baut CAB auf einer deutlich älteren Technologie auf als die gegenwärtig übliche NLP-Methodik des Machine Learning (ML) via Transformern. Während es also naheliegt, automatische Textnormalisierung aus einem ML-Ansatz heraus zu operationalisieren, ergibt sich hierbei eine neue Schwierigkeit: Textnormalisierung ist in vielen Fällen eine unscharfe Praxis und selten in Konventionen festgelegt.

Mit diesem Beitrag möchten wir folgendes leisten: Erstens wollen wir auf die Komplexität der Definition von *normalisiertem Text*, sowie auf den unterschiedlich engen Zusammenhang dieser Definition und der Methode für die *Textnormalisierung* hinweisen. Zweitens vergleichen wir in einer Fallstudie quantitativ die Güte von CAB mit gegenwärtigen Transformer-Architekturen zur Textnormalisierung. Es lassen sich zwar Verbesserungen erzielen, aber die Ergebnisse sind weiterhin ausbaufähig. Drittens ordnen wir abschließend die Ergebnisse der Fallstudie in das Vorhaben von *Text+* ein, der DH-Community ein Nachfolgesystem für CAB zur Verfügung zu stellen.

Herangehensweisen

Wie ein normalisierter Text genau aussehen soll, ist keine triviale Frage, denn die sprachlichen Unterschiede zwischen historischen und modernen Texten betreffen verschiedene Ebenen. So illustriert das folgende Beispiel von 1749 Abweichungen auf der Ebene der Schreibung (*wil* statt *will*), der Wortbildung (*fürstellen* statt *vorstellen*) und der Syntax (Pronomen im Nachfeld).

*Ich wil mir in meinem Sterben fürstellen dich, o JEsu!*²

Eine Reihe von Konventionen für die manuelle Normalisierung in verschiedenen sprachhistorischen Korpusprojekten (Durrell et al., 2012; Krasselt et al., 2015; Odebrecht et al., 2020) illustrieren die Komplexität der Aufgabe, die hier nur kurz angedeutet werden soll:³ Wie sollen ausgestorbene Lemmata normalisiert werden? Sollen morphosyntaktische Abweichungen normalisiert werden, und welche? Soll auch die Wortstellung der Normalisierung unterliegen (vgl. die Stellung von *dich* im angegebenen Beispiel)? Andere Fragen betreffen die Struktur der Korpusannotation: Soll jedem Token im Originaltext genau ein modernes Token zugeordnet werden oder sollen 1:n-, n:1-, n:m-Beziehungen zwischen beiden Ebenen möglich sein, um etwa Veränderungen in Getrennt- und Zusammenschreibung abzubilden (*Statt zu finden* vs. *stattzufinden*)?⁴ Die Antworten auf diese Fragen richten sich auch nach der vorgesehenen Nutzung der normalisierten Version.

Bei der technischen Umsetzung eines Normalisierers kann zwischen zwei Ansätzen unterschieden werden: einerseits Verfahren, die auf manuell erstellten Regeln basieren, und andererseits jene, die auf dem Lernen aus annotierten Daten basieren. Diese zwei unterschiedlichen Verfahren unterscheiden sich auch darin, wie eng die Definition von normalisiertem Text und die Methode zur automatischen Erzeugung von normalisiertem Text (die eigentliche *Textnormalisierung*) zusammenhängen.

Das System CAB verfolgt den ersten Ansatz. Zu CAB gehört keine natürlichsprachliche Beschreibung von normalisiertem Text, wie in den oben genannten Konventionen, aber es basiert auf einer Reihe formaler Regeln. Die Definition von normalisiertem Text und das Vorgehen zur automatischen Normalisierung fallen also zusammen, wobei die Regeln mit einer bestimmten Vorstellung vom Ziel der Normalisierung (d. h. von korrekt normalisiertem Text) entworfen wurden.

Beim zweiten Ansatz, dem Lernen aus Trainingsdaten wie im Fall der hier evaluierten ML-Modelle, sind die Definition von normalisiertem Text und der Prozess der Normalisierung vollständig getrennt zu sehen. Das Trainingskorpus zu erstellen, setzt eine Definition von normalisiertem Text voraus, etwa in Form der o. g. Konventionen, oder – im Fall von Textausgaben – linguistischer Intuition. Im Training selbst allerdings spielt diese Definition keine Rolle. Die Modelle lernen nur die statistischen Zusammenhänge zwischen historischem und normalisiertem Text.

Fallstudie: Normalisierung historischer Literatursprache des 19. Jahrhunderts

Um die aktuellen Möglichkeiten und Grenzen von automatisierter Textnormalisierung an einem konkreten Fall zu erörtern, fokussiert unsere Fallstudie sich auf die Normalisierung von historischer Literatursprache des 19. Jahrhunderts. Wir vergleichen hierbei quantitativ die methodischen Ansätze: auf der einen Seite das regelbasierte Normalisierungssystem CAB,⁵ auf der anderen Seite ML-Systeme, welche auf einem Parallelkorpus mit historischer und moderner Schreibweise trainiert sind. Für Letzteres trainieren wir zwei Transformer-Modelle, einmal auf Satz- und einmal auf Type-Ebene.

```
DTA      Der Officier mußte sich dazu setzen, man trank und ließ sich's wohl feyn.
TextGrid Der Offizier musste sich dazusetzen, man trank und ließ sich es wohl sein.
Input    Der Officier mußte sich dazu setzen , man trank und ließ sich's wohl seyn .
Output   Der Offizier musste sich dazu# setzen , man trank und ließ sich_es wohl sein .
```

Abbildung 1: (Fiktives) Beispiel einer Textnormalisierungs-Instanz der Fallstudie. „DTA“: Originalschreibweise einer Erstausgabe; „TextGrid“: normalisierte Form aus gegenwärtiger Edition; „Input“: tokenisierte und vorverarbeitete Eingabe für das Normalisierungssystem; „Output“: gewünschte Gold-Ausgabe. NB: „Input“ und „Output“ sind auf der Token-Ebene aligniert.

Für das Modell auf Satz-Ebene führen wir ein *Fine-tuning* auf dem Encoder-Decoder-Modell ByT5-small (300M Parameter; Xue et al., 2021) durch, um Satz für Satz in gegenwärtige Orthographie zu „übersetzen“. ⁶ Encoder-Decoder-Modelle sind für solche „Übersetzungs-Tasks“ üblicherweise am besten geeignet. Das spezielle Modell ByT5 wurde auf dem multilingualen (nicht-parallelen) C4-Korpus zur Rekonstruktion von *span corruptions* unüberwacht vortrainiert. Eine Besonderheit ist, dass es keinen Tokenisierer gibt, sondern der Text Byte für Byte verarbeitet wird. Xue et al. (2021) zeigen, dass die gängige Subword-Tokenisierung von vortrainierten Modellen bei *noisy text* (z. B. Text mit *zUFälligEr GRoßSchHReiBunG*) zu Problemen führen kann und dass das Tokenisierer-freie ByT5 in diesem Setting besser abschneidet. Auch historischer Text weicht von der Standardschreibung ab und kann damit als *noisy* gelten. In ähnlichen Kontexten wurde ByT5 schon mit vielversprechenden Ergebnissen für die Normalisierung von Social-Media-Text und von dialektalen Daten eingesetzt (Samuel und Straka, 2021; Kuparinen et al., 2023).

Das zweite Modell ist ein Hybrid und besteht aus einem Encoder-Decoder-Modell und einem GPT-Modell. Die Encoder-Decoder-Komponente normalisiert kontext-unabhängig Type für Type. Dieses Modell ist ähnlich zu ByT5 aufgebaut, aber wesentlich kleiner, und wurde ohne Vortraining von Beginn an für den Task der Type-Normalisierung trainiert. Die generierten Normalisierungshypothesen pro Type werden anschließend noch von einem zweiten vortrainierten deutschsprachigen GPT2-Modell (ohne weiteres Fine-Tuning) entsprechend des Kontextes gewichtet (8M + 124M Parameter). ⁷ Vereinfacht gesprochen wird dabei beispielsweise für einen Eingabesatz „Die Gemälde an den Wänden des Pallastes beeindruckten alle“ zuerst vom ersten Encoder-Decoder-Modell für das fragwürdige Type *Pallastes* die Normalisierungshypothesen *Pallastes*, *Palistes* und *Palastes* generiert. Dann wird vom zweiten GPT-Modell evaluiert, welcher der drei Sätze „Die Gemälde an den Wänden des Pallastes/Palistes/Palastes beeindruckten alle“ am wahrscheinlichsten ist und als Ausgabe ausgewählt.

Als weitere Baseline trainieren wir noch den auf *statistical machine translation* basierenden Type-Normalisierer csmtiser⁸ (Ljubešić et al., 2016).

Als Datensatz zum Training bzw. für die Evaluation verwenden wir das Parallelkorpus *DTA EvalCorpus*⁹ (Jurish et al., 2013). Das Parallelkorpus aligniert auf Token-Ebene Erstausgaben des DTA mit gegenwärtigen Editionen (ab 1950) der Digitalen Bibliothek TextGrid. Diese Alignierung wurde in einem ersten Schritt automatisch berechnet, und anschließend auf Type- und Token-Ebene manuell verifiziert und ggf. korrigiert. Während dieses Parallelkorpus zwar manuell verifiziert wurde, muss die Heterogenität des Quellkorpus TextGrid problematisiert werden: so folgen die Editionen nicht „einem“ festen Normalisierungsstandard, sondern die Normalisierung geht je nach Edition und Vorstellungen der Editor:innen unterschiedlich weit. Diese wurden – anders als bei historischen Korpusprojekten – nicht in Form von Guidelines explizit gemacht. Hinzu

kommt noch, dass die Editionen wahlweise der neuen oder der alten Rechtschreibung von vor 1996 folgen.

Wir beschränken uns auf den nicht-lyrischen belletristischen Teil (85 Dokumente, 1784–1901) und erhalten nach Vorverarbeitung¹⁰ insgesamt 4 Millionen Tokens auf 180.000 Sätzen. Insbesondere bringen wir für die Type-basierten Modelle das Parallelkorpus in eine 1-zu-1-Alignierung, wie in Abb. 1. Durch die hierbei eingeführten Pseudo-Zeichen _ und # (siehe Abb. 1; vgl. Bawden et al., 2022) können wir in der Evaluation Token für Token mit der Gold-Variante vergleichen. Wir führen unseren Versuch auf einem 60-20-20 Train-dev-test-Split durch. Nähere Details zum Hybrid-Modell, zum Versuchsaufbau und weitere Auswertungen sind ferner von Ehrmanntraut (2024) in einem separaten Bericht beschrieben.¹¹

Tabelle 1 zeigt die Ergebnisse auf dem Test-Split in Wortgenauigkeit („WordAcc“) auf Token-Ebene, exklusive Interpunktion. Für eine verfeinerte Analyse geben wir neben der Gesamtgenauigkeit auch die Genauigkeit für Tokens an, welche die Modelle bereits im Training gesehen haben (invocab) bzw. welche die Modelle zum ersten Mal sehen (OOV). Nur letzteres Maß bildet die Generalisierungsleistung der Modelle ab (vgl. Robertson und Goldwater, 2018).

Tabelle 1: Wortgenauigkeit der Modelle auf dem Test-Split. „Invocab“ bzw. „OOV“ bezieht sich auf im Training gesehene bzw. nicht gesehene Tokens. Da CAB nicht trainiert ist, sind „invocab“ bzw. „OOV“ hier leer. „Trainings-Lexikon“ bezeichnet die Baseline, welche jedes Wort zur häufigsten beobachteten Normalisierung im Trainings-Datensatz normalisiert, und OOV-Wörter im Original beibehält. „Theoretisch bestmögliches Lexikon“ bezeichnet das bestmögliche Ergebnis, welches unter idealen (kontext-unabhängigen) Type-für-Type-Ersetzungen erreicht werden kann.

System / Baseline	Ebene	WordAcc	WordAcc (invocab)	WordAcc (OOV)
Original beibehalten	Type	96,513	97,015	83,912
Trainings-Lexikon	Type	98,881	99,477	83,912
Theoretisch bestmögliches Lexikon	Type	99,547	99,533	99,896
CAB	Type	98,072	—	—
csmtiser	Type	98,940	99,321	89,369
ByT5-Modell	Satz	99,156	99,431	92,237
Transformer-Hybrid-Modell	Type	99,194	99,493	91,701

Zunächst zu CAB: hier sehen wir, dass selbst ohne Trainingsressourcen das System eine starke Baseline darstellt. Allein durch die integrierten Regeln können gegenüber dem Beibehalten des Originals schon fast die Hälfte der Fehler eliminiert werden. Der Vergleich mit ML-Ansätzen zeigt, dass diese genauer arbeiten als CAB; so machen die Transformer-Modelle über alle Tokens des Test-Sets nur halb so viele Fehler wie CAB (0,844% Fehler bei ByT5 unter „WordAcc“ vs. 1,928% Fehler bei CAB), wobei dieser Vorteil durch ein Trainingskorpus und ressourcenintensives Training/Inferenz erkauft wird. Dennoch zeigt sich aus dieser Analyse, dass Textnormalisierung noch nicht als „gelöst“ gelten kann: selbst die Transformer-Modelle generieren immer noch ein fehlerhaftes Token pro 100 Tokens – unter den OOV-Tokens sogar ein Fehler pro 10 Tokens.

Eine mögliche Erklärung für den Performance-Unterschied zwischen CAB und den Transformern könnte lauten, dass CABs Regeln auf eine andere Vorstellung von normalisiertem Text ausgerichtet sind, die nicht vollständ-

dig mit der Vorstellung übereinstimmt, die in den modernen Editionen realisiert ist. Das ist zum Teil der Fall: so tilgt CAB im Gegensatz zum Parallelkorpus und den Transformer-Modellen keine Apostrophe vor Genitiv-s (*Hoppensack's* nicht zu *Hoppensacks*). An diesem Beispiel sehen wir, wie wichtig es ist, die unterschiedlichen Definitionen von normalisiertem Text mitzudenken. Dennoch machen sämtliche Apostroph-Schreibungen nur 0,344% der Tokens im Testset aus, und erklären daher nicht vollständig die Differenz zu den Transformer-Modellen. In einer Stichprobe zeigte sich, dass zahlreiche der verbleibenden abweichenden Normalisierungen von CAB tatsächlich nicht als eine alternative geeignete Normalisierungsform gewertet werden können.

Abschließend wollen wir noch die zwei verschiedenen Transformer-Architekturen diskutieren: Auf Basis der Fallstudie lässt sich zwischen den Modellen kein Unterschied in der Genauigkeit feststellen. Ein Fine-Tuning von vortrainierten generativen LLMs wie ByT5 hat aber den entscheidenden Vorteil, dass es einfach umsetzbar und sehr flexibel ist, z. B. im Bezug auf Wortstellung, Tokenisierung, usw. Dagegen zeigt sich, dass auch kleinere Modelle wie das Hybrid-Modell genauso gut arbeiten können, dies ist aber im Design komplexer und abhängig von fester Wortstellung. Es würde sich daher anbieten, in zukünftiger Arbeit weitere Korpora bzw. Datensätze heranzuziehen, und insbesondere die Abhängigkeit vom Volumen des Trainingsmaterials auf die Güte der Modelle zu untersuchen. Die Evaluation sollte auch auf andere Textsorten und ältere Zeitstufen ausgeweitet werden, da diese schwieriger zu normalisieren sein dürften. Dafür bleibt auch die Gewinnung geeigneter Trainingsdaten eine Herausforderung.

Ausblick: Eine Infrastruktur für die Digital Humanities

Wie kann die Nutzung von Tools für die automatisierte historische Normalisierung in der Praxis der Digital Humanities aussehen? Aktuell besteht in erster Linie die Möglichkeit auf den CAB-Webservice zurückzugreifen.¹² Ein lokaler Betrieb für Nutzende ist nicht möglich, da die CAB-Instanz des DTA an zentraler Stelle urheberrechtlich geschützte Ressourcen einbindet.

Die hier vorgestellten Modelle bzw. deren Weiterentwicklungen stellen wir frei zur Verfügung. Für das Hybrid-Modell stehen der Code und die Modelle auf Github/Huggingface zur Nutzung bereit.¹³ Das ByT5-basierte Modell wurde als *Transnormer* innerhalb des Konsortiums *Text+* der Nationalen Forschungsdateninfrastruktur (NFDI) entwickelt mit dem Ziel, einen frei und flexibel nachnutzbaren Nachfolger für CAB bereitzustellen. Eine Weiterentwicklung des hier vorgestellten Modells ist auf Huggingface veröffentlicht.¹⁴ Weitere Veröffentlichungen für die Normalisierung historischer Texte ab dem 17. Jahrhundert werden folgen.

Lokal mit den Modellen zu arbeiten, bietet Nutzenden eine größere Flexibilität: Die Modelle können in eigene

Workflows eingebunden und auch nachtrainiert werden. Voraussetzung für Letzteres sind weitere Trainingsdaten in Form von Parallelkorpora aus historischem und gold-normalisiertem Text. Mit ausreichend vielen Trainingsdaten wäre ebenso ein vollständiges Training neuer Modelle nach den hier vorgestellten Architekturen möglich. In jedem Fall geht jedoch die Nutzung von Transformer-Modellen mit erhöhten Anforderungen an Hardware, höherem Ressourcenverbrauch und praktischen Hürden einher. Es bleibt zu diskutieren, wie die Voraussetzungen geschaffen werden können, dass gerade auch kleinere Projekte diese Herausforderungen bewältigen können, bzw. welche Infrastrukturen (z. B. Webservices) benötigt werden.

Fußnoten

1. <https://deustextarchiv.de/public/cab/>
2. Starck, Johann Friedrich: Tägliches Hand-Buch in guten und bösen Tagen. Frankfurt/Leipzig, 1749, S. 630. In: Deutsches Textarchiv, abgerufen am 28.11.2024.
3. Für einen Überblick zu verschiedenen Konventionen und eine ausführliche Diskussion zur Definition von Normalisierung, siehe Bollmann (2018), Kapitel 2.
4. In den genannten Konventionen bleibt die Wortreihenfolge unangetastet, womit auch die Ebenen aligniert werden können. 1:n/n:1-Beziehungen sind aber teils vorgesehen, etwa im RIDGES-Korpus (Odebrecht et al. 2020).
5. Um die Güte von CAB sinnvoll zu messen, haben wir das Ausnahmelexikon deaktiviert, denn dieses wurde teils aus dem DTA EvalCorpus erstellt. NB: CAB kann keine Wortverbindungen generieren.
6. Code: ; Das Fine-Tuning dauerte auf einer RTX2080ti etwa 24h.
7. Code: ; Type-Modell: ; GPT-Modell: . Das Training des Type-Modells dauerte auf einer RTX2080ti etwa 4h; das GPT-Modell wurde ohne Fine-Tuning übernommen.
8. <https://github.com/clarinsi/csmtiser>
9. <https://kaskade.dwds.de/~moocow/software/dtaec/>
10. Wir exkludieren insbesondere Sätze, welche im Parallelkorpus von Annotator:innen manuell als fremdsprachiges Material, als Druck-/OCR-Fehler, oder als „ungeeignet für die Aufnahme in das Trainingsmaterial (z. B. Tokenizer-Fehler, ausgestorbene Lemmata, zusammengesetzte Komposita)“ eingestuft wurde. Anschließend wenden wir eine regelbasierte Transliteration von Lang-S (/) auf Kurz-S (s) an, und bringen Umlaute in ihre gegenwärtige Form (æ zu ä usw.). NB: Für das ByT5-basierte Modell ist eine solche Transliteration nicht nötig, da es per Definition keine unbekannten Zeichen für das Modell gibt.
11. Code zur Reproduktion des Datensatzes und der Evaluation:
12. Die tatsächliche Performance von der aufrufbaren CAB-Instanz des DTA ist dabei i. d. R. besser als im hier vorgestellten Experiment, da das Ausnahmenlexikon einige Fehler des regelbasierten Ansatzes auffängt.

13. Code: ; Type-Modell: ; GPT-Modell: . Das Training des Type-Modells dauerte auf einer RTX2080ti etwa 4h; das GPT-Modell wurde ohne Fine-Tuning übernommen.
14. . Das veröffentlichte Modell hat die gleiche Architektur wie das hier beschriebene ByT5 Modell, wurde jedoch mit überarbeiteten und teils anderen Dokumenten aus dem DTA EvalCorpus trainiert. Der Datensatz ist auf Huggingface verfügbar: <https://huggingface.co/datasets/ybracke/dta-reviEvalCorpus-v1>.

Bibliographie

- Bawden, Rachel, Jonathan Poinhos, Eleni Kogktsidou, Philippe Gambette, Benoît Sagot, und Simon Gabay.** 2022. „Automatic Normalisation of Early Modern French“. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, hg. von Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, u. a., 3354–66. Marseille, Frankreich: European Language Resources Association. <https://aclanthology.org/2022.lrec-1.358>.
- Bollmann, Marcel.** 2018. *Normalization of historical texts with neural network models*. Dissertation, Ruhr-Universität Bochum. <https://doi.org/10.13154/294-6213>.
- Durrell, Martin, Paul Bennett, Silke Scheible, und Richard J. Whitt.** 2012. The GerManC Corpus. University of Manchester. <http://hdl.handle.net/20.500.14106/2544>, https://www.ids-mannheim.de/fileadmin/lexik/uvw/dateien/GerManC_Documentation.pdf (zugegriffen: 22.07.2024).
- Ehrmanntraut, Anton.** 2024. „Historical German Text Normalization Using Type- and Token-Based Language Modeling“. ArXiv Pre-print. <https://doi.org/10.48550/arXiv.2409.02841>.
- Jurish, Bryan.** 2012. *Finite-State Canonicalization Techniques for Historical German*. Dissertation, Universität Potsdam. <https://nbn-resolving.org/urn:nbn:de:kobv:517-opus-55789>.
- Jurish, Bryan, Marko Drotschmann, und Henriette Ast.** 2013. „Constructing a Canonicalized Corpus of Historical German by Text Alignment“. In *New Methods in Historical Corpora*, 221–34. Tübingen: Narr.
- Kogktsidou, Eleni, und Philippe Gambette.** 2020. „Normalisation of 16th and 17th century texts in French and geographical named entity recognition“. In *Proceedings of the 4th ACM SIGSPATIAL Workshop on Geospatial Humanities*, 28–34. New York, USA: Association for Computing Machinery. <https://doi.org/10.1145/3423337.3429437>.
- Krasselt, Julia, Marcel Bollmann, Stefanie Dipper, und Florian Petran.** 2015. *Guidelines für die Normalisierung historischer deutscher Texte / Guidelines for Normalizing Historical German Texts*. Bochumer Linguistische Arbeitsberichte 15. <https://www.linguistics.rub.de/>

forschung/arbeitsberichte/15.pdf
(zugegriffen: 22.07.2024).

Kuparinen, Olli, Aleksandra Miletić, und Yves Scherrer. 2023. „Dialect-to-Standard Normalization: A Large-Scale Multilingual Evaluation“. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, herausgegeben von Houda Bouamor, Juan Pino, und Kalika Bali, 13814–28. Singapore: Association for Computational Linguistics. <https://doi.org/10.18653/v1/2023.findings-emnlp.923>.

Ljubešić, Nikola, Katja Zupan, Darja Fišer, und Tomaž Erjavec. 2016. „Normalising Slovene data: historical texts vs. user-generated content“. In *Proceedings of the 13th Conference on Natural Language Processing, KONVENS 2016*, hg. von Stefanie Dipper, Friedrich Neubarth, und Heike Zinsmeister. Bochumer Linguistische Arbeitsberichte 16. Bochum. https://www.linguistics.rub.de/konvens16/pub/19_konvensproc.pdf (zugegriffen: 22.07.2024).

Odebrecht, Carolin, Laura Perlitz, Gohar Schnelle, und Catharina Fischer. 2020. *Annotationsrichtlinien zu Ridges Herbolgy Version 9.0*. Humboldt-Universität zu Berlin. https://www.linguistik.hu-berlin.de/de/institut/professuren/korpuslinguistik/forschung/ridges-projekt/download-files/pubs/ridgesv9_2020-03.pdf (zugegriffen: 22.07.2024).

Pettersson, Eva, Beáta Megyesi, und Jörg Tiedemann. 2013. „An SMT Approach to Automatic Annotation of Historical Text“. In *Proceedings of the Workshop on Computational Historical Linguistics at NODALIDA 2013*, hg. von Þórhallur Eypórsson, Lars Borin, Dag Haug, und Eiríkur Rögnvaldsson, 54–69. Oslo, Norwegen: Northern European Association for Language Technology. https://ep.liu.se/en/conference-article.aspx?series=ecp&issue=87&Article_No=5 (zugegriffen: 22.07.2024).

Robertson, Alexander, und Sharon Goldwater. 2018. „Evaluating Historical Text Normalization Systems: How Well Do They Generalize?“. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, hg. von Marilyn Walker, Heng Ji, und Amanda Stent, 720–25. New Orleans, Louisiana, USA: Association for Computational Linguistics. <https://doi.org/10.18653/v1/N18-2113>.

Samuel, David, und Milan Straka. 2021. „ÚFAL at MultiLexNorm 2021: Improving Multilingual Lexical Normalization by Fine-tuning ByT5“. In *Proceedings of the Seventh Workshop on Noisy User-generated Text (W-NUT 2021)*, herausgegeben von Wei Xu, Alan Ritter, Tim Baldwin, und Afshin Rahimi, 483–92. Online: Association for Computational Linguistics. <https://doi.org/10.18653/v1/2021.wnut-1.54>.

Scheible, Silke, Richard J. Whitt, Martin Durrell, und Paul Bennett. 2011. „A Gold Standard Corpus of Early Modern German“. In *Proceedings of the 5th Linguistic Annotation Workshop*, hg. von Nancy Ide, Adam Meyers, Sameer Pradhan, und Katrin Tomanek, 124–28. Portland,

Oregon, USA: Association for Computational Linguistics. <https://aclanthology.org/W11-0415>.

Xue, Linting, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, und Colin Raffel. 2021. „ByT5: Towards a token-free future with pre-trained byte-to-byte models“. ArXiv Pre-print. <https://doi.org/10.48550/arXiv.2105.13626>.