# (Semi-)Automatic Normalization of Historical Texts using Distance Measures and the Norma tool

Marcel Bollmann

Department of Linguistics
Ruhr-University Bochum, Germany

Second Workshop on Annotation of Corpora
for Research in the Humanities

November 29, 2012, Lisbon, Portugal

# Motivation

The problem with historical data...

- High variance in spelling
- Difficult to annotate with tools aimed at modern data, e.g. POS taggers
- None or very little training data to (re-)train annotation tools

# Motivation

The problem with historical data...

- High variance in spelling
- Difficult to annotate with tools aimed at modern data, e.g. POS taggers
- None or very little training data to (re-)train annotation tools

A possible solution...

- Pre-processing data to “modernize” spelling

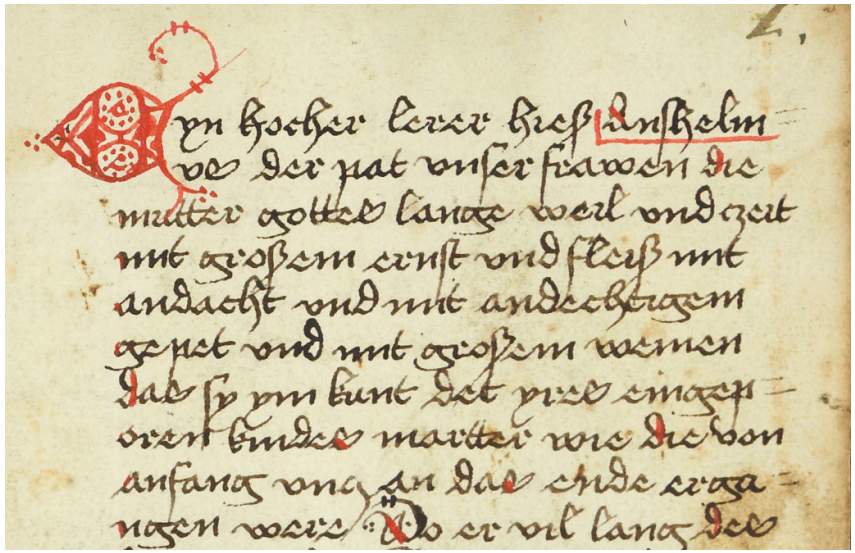
**Normalization** as the process of mapping historical spellings to its modern equivalents.

# Outline

# Corpora

## Anselm Corpus

- Collection of Early New High German (ENHG) texts
  "Interrogatio Sancti Anselmi de Passione Domini"
  (*Questions by Saint Anselm about the Lord's Passion*)
- More than
  50 manuscripts and
  prints (in German)
- $14^{th}$–$16^{th}$ centuries
- Various German
  dialects



Sample from an Anselm manuscript

# Corpora

## Anselm Corpus

### Goals

- Lemmatization, POS tagging
- Paragraph, sentence, and word alignment
- Digital edition

### Method

- Normalization of historical wordforms to modern ones
  - Allows the use of already-existing tools
  - Simplifies wordform queries in a resulting corpus

# Corpora

## Anselm Corpus

| | | | | | | |
|---|---|---|---|---|---|---|
| **ENHG$_1$** | do | meín | chind | híet | geezzen· | … |
| **ENHG$_2$** | da | myn | kínt | hatte | geſzen | … |
| **ENHG$_3$** | do | mín | kínt | hatt | geſſen | … |

---

| | | | | | | |
|---|---|---|---|---|---|---|
| **Norm** | da | mein | kind | hatte | gegessen | … |
| | *as* | *my* | *child* | *had* | *eaten* | |

# Corpora

## Luther Bible

- Bible translation by Martin Luther
  - 1545 version and a modernized equivalent
  - Freely available on the web:
    `http://www.sermon-online.de/`
- Extraction of 550,000 alignment pairs
  - Randomly split into development/training/evaluation corpus

→ **Large test corpus for normalization**

# Methods

Comparison of different normalization methods:

1. Wordlist mapping
2. Rule-based normalization
   - Character rewrite rules
3. Distance-based normalization
   - (Weighted) Levenshtein distance

# Methods

## Wordlist Mapping

- Word-to-word mappings
- Learned from an aligned corpus
- Chooses most frequent candidate wordform
- No knowledge about spelling variation

**Example**

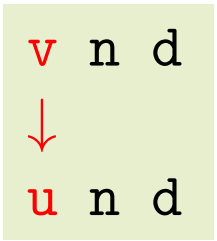
| | | | |
|---|---|---|---|
| do | $\rightarrow$ | da | 50 |
| meín | $\rightarrow$ | mein | 30 |
| myn | $\rightarrow$ | mein | 30 |
| mín | $\rightarrow$ | mein | 30 |
| | | $\vdots$ | |
| hatt | $\rightarrow$ | hatte | 50 |
| hatt | $\rightarrow$ | hat | 20 |
| hatt | $\rightarrow$ | hut | 1 |

# Methods

## Rule-Based Normalization

- "Context-aware" character rewrite rules

$$\texttt{v} \rightarrow \texttt{u} \;/\; \texttt{\#} \_ \texttt{n}$$




- Learned from aligned training corpus
  - **Levenshtein distance:** Minimum number of edit operations to transform string *a* into string *b*
  - **Modified algorithm:** Outputs the actual edit operations

# Methods

## Rule-Based Normalization

- Substitution rules

  $\texttt{v} \rightarrow \texttt{u} \;/\; \texttt{\#} \;\_\; \texttt{n}$

- Identity rules

  $\texttt{n} \rightarrow \texttt{n} \;/\; \texttt{e} \;\_\; \texttt{\#}$

- Insertion rules

  $\varepsilon \rightarrow \texttt{l} \;/\; \texttt{o} \;\_\; \texttt{l}$

- Deletion rules

  $\texttt{f} \rightarrow \varepsilon \;/\; \texttt{u} \;\_\; \texttt{f}$

- Additional lexicon lookup to prevent nonsense words

# Methods

## Rule-Based Normalization

- Substitution rules

  $\texttt{v} \rightarrow \texttt{u} \;/\; \texttt{\#} \;\_\; \texttt{n}$

- Identity rules

  $\texttt{n} \rightarrow \texttt{n} \;/\; \texttt{e} \;\_\; \texttt{\#}$

- Insertion rules

  $\varepsilon \rightarrow \texttt{l} \;/\; \texttt{o} \;\_\; \texttt{l}$

- Deletion rules

  $\texttt{f} \rightarrow \varepsilon \;/\; \texttt{u} \;\_\; \texttt{f}$

→ **Identity and non-identity rules intended to “compete”**

- Additional lexicon lookup to prevent nonsense words

# Methods

## Distance-Based Normalization

- **Levenshtein distance:** Count number of edit operations

$$\texttt{myn} \rightarrow \texttt{mein} \qquad \texttt{d} = 2$$

# Methods

## Distance-Based Normalization

- **Levenshtein distance:** Count number of edit operations

$$\texttt{myn} \rightarrow \texttt{mein} \qquad \texttt{d = 2}$$

- **Weighted Levenshtein distance**
  - Assigns weights to edit operations
    e.g., $d(\text{'y'}, \text{'ei'}) = 0.8$
  - Edit operations are directed/asymmetric
  - Edit operations may span multiple characters

$$\texttt{myn} \rightarrow \texttt{mein} \qquad \texttt{d = 0.8}$$

# Methods

## Distance-Based Normalization

- Find lexicon entry with lowest distance to input string

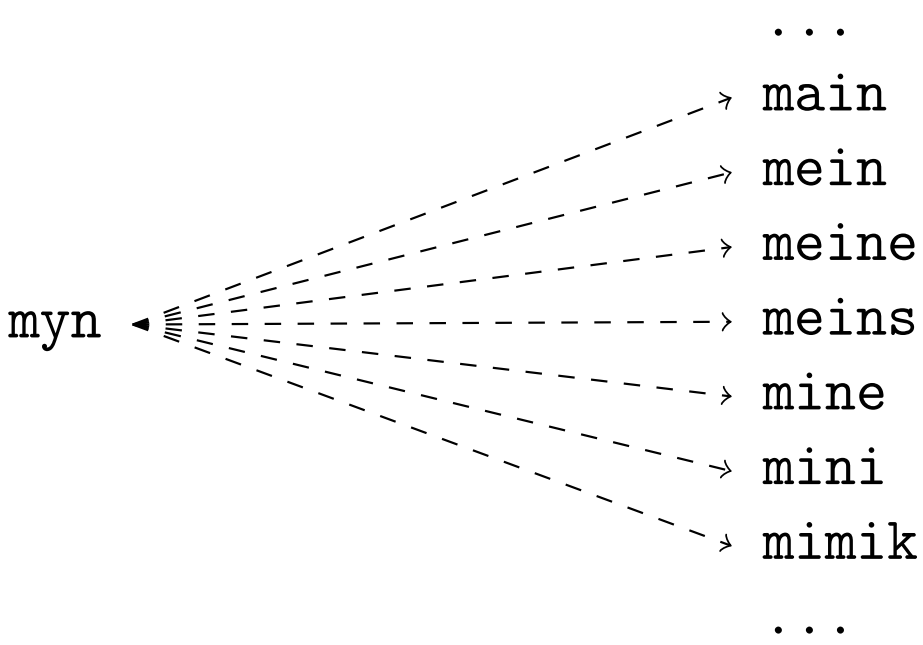
myn

```
...
main
mein
meine
meins
mine
mini
mimik
...
```

# Methods

## Distance-Based Normalization

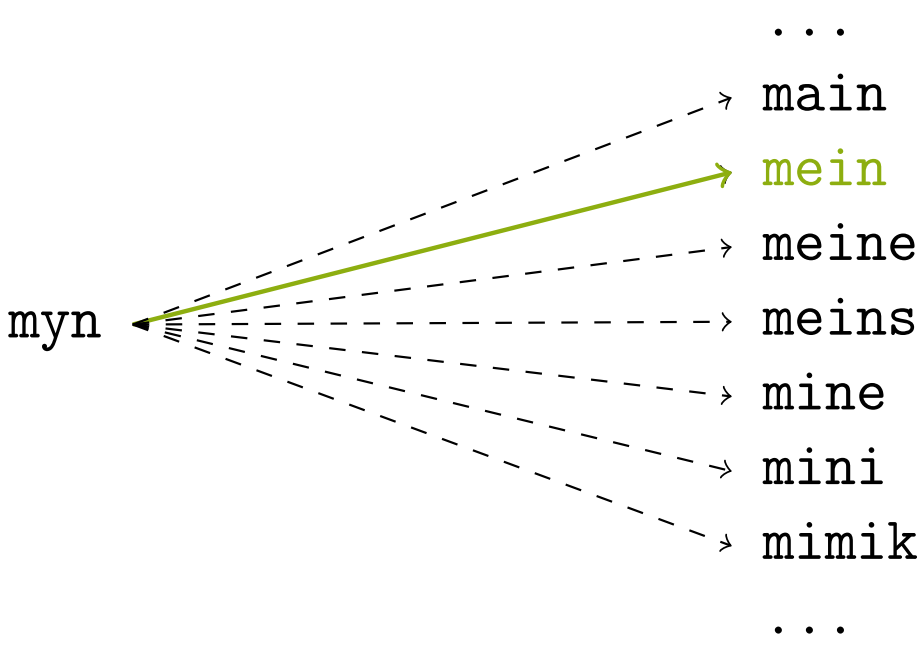
- Find lexicon entry with lowest distance to input string



# Methods

## Distance-Based Normalization

- Find lexicon entry with lowest distance to input string



# Methods

- Which normalization method is “best”?
- Does a combination of methods work better?

- Many other algorithms for normalization
  - Ernst-Gerlach & Fuhr (2006), Hauser & Schulz (2007): Information Retrieval (IR) on historical texts
  - Baron & Rayson (2009): focus on Early Modern English
  - Jurish (2010): evaluated as IR task

  → Results not easily comparable!

# Methods

- Which normalization method is “best”?
- Does a combination of methods work better?

- Many other algorithms for normalization
  - Ernst-Gerlach & Fuhr (2006), Hauser & Schulz (2007): Information Retrieval (IR) on historical texts
  - Baron & Rayson (2009): focus on Early Modern English
  - Jurish (2010): evaluated as IR task

  → Results not easily comparable!

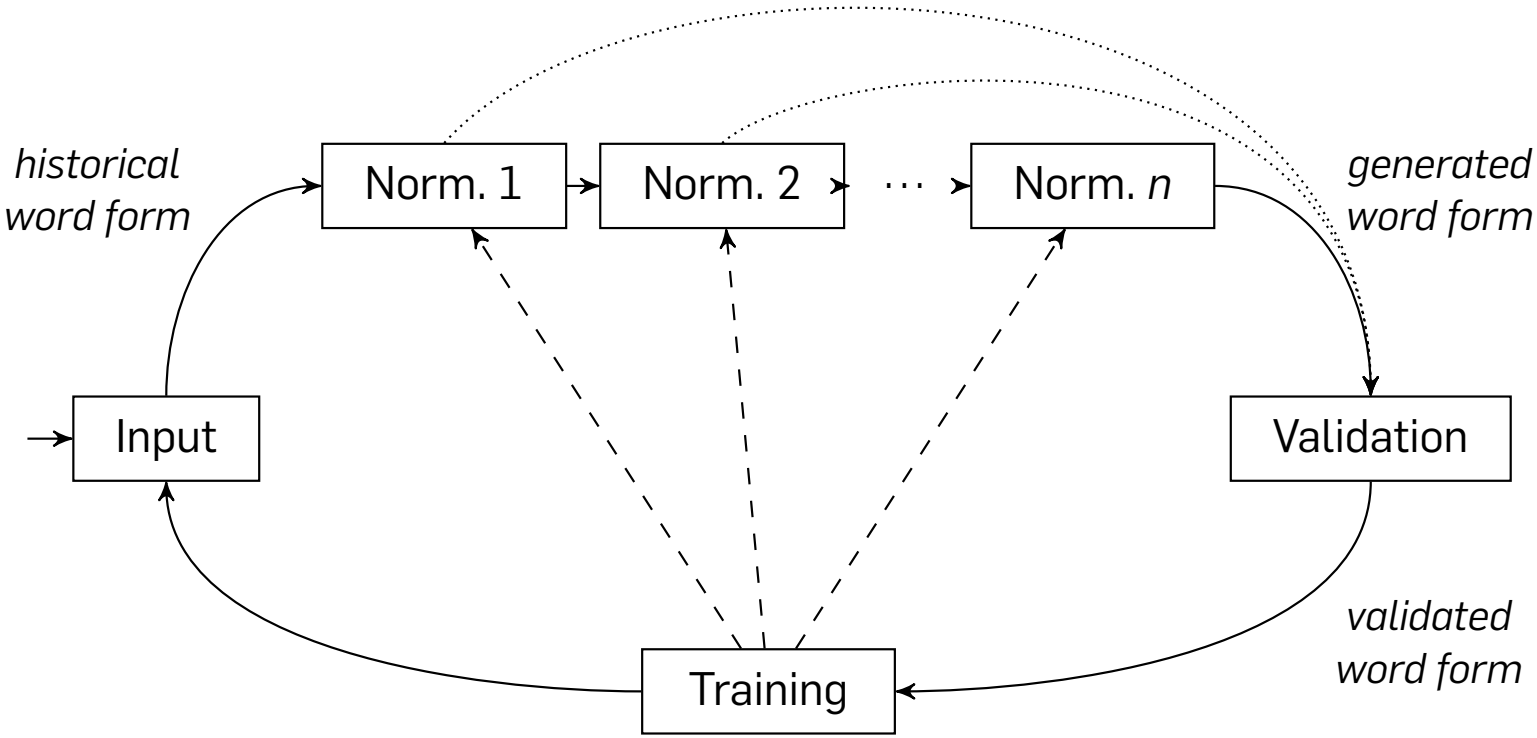
# Norma Tool

## Overview

- *Norma*: an interactive normalization tool
- Key features
  - Automatic and semi-automatic modes
  - Easy extensibility (with regard to normalization algorithms)
  - Support for dynamically trainable normalization methods
- Current limitations
  - No token context considered
  - Command-line interface only

# Norma Tool

## Description



# Norma Tool

## Example

```
> chind
```

1. Generate normalization candidate

# Norma Tool

## Example

```
> chind
```

1. Generate normalization candidate
   - Wordlist substitution: no mapping found

# Norma Tool

## Example

```
> chind
  kind (0.62)
? _
```

1. Generate normalization candidate
   - Wordlist substitution: no mapping found
   - Rule-based normalizer: generates *kind* with probability score 0.62

# Norma Tool

## Example

```
> chind
  kind (0.62)
?
```

1. Generate normalization candidate
   - Wordlist substitution: no mapping found
   - Rule-based normalizer: generates *kind* with probability score 0.62
2. User confirms *kind* as correct

# Norma Tool

## Example

```
> chind
  kind (0.62)
?
```

1. Generate normalization candidate
   - Wordlist substitution: no mapping found
   - Rule-based normalizer: generates *kind* with probability score 0.62
2. User confirms *kind* as correct
3. Retrain normalizer modules

# Norma Tool

## Example

```
> chind
  kind (0.62)
?
```

1. Generate normalization candidate
   - Wordlist substitution: no mapping found
   - Rule-based normalizer: generates *kind* with probability score 0.62
2. User confirms *kind* as correct
3. Retrain normalizer modules
   - Wordlist substitution: learns mapping *chind* $\rightarrow$ *kind*

# Norma Tool

## Example

```
> chind
  kind (0.62)
?
```

1. Generate normalization candidate
   - Wordlist substitution: no mapping found
   - Rule-based normalizer: generates *kind* with probability score 0.62
2. User confirms *kind* as correct
3. Retrain normalizer modules
   - Wordlist substitution: learns mapping *chind* $\rightarrow$ *kind*
   - Rule-based normalizer: adds rules to its ruleset

# Norma Tool

## Comparison to VARD 2

- VARD 2 normalization tool
  `http://www.comp.lancs.ac.uk/~barona/vard2/`
- Graphical user interface
- Combination of normalization methods
- Disadvantages
  - Designed for Early Modern English
  - Not completely customizable

  → Problematic especially for phonetic matching

# Evaluation

## Procedure

- Compare different normalization methods
  1. On their own
  2. In combination
- Luther bible
  - Training part: 218,504 tokens
  - Evaluation part: 109,258 tokens
- Anselm text
  - Training part: 500 tokens
  - Evaluation part: 4,020 tokens

# Evaluation

## Procedure

- Wordlist mapping & Rule-based normalization
  - → Automatically trained
- Weighted Levenshtein distance (MultiWLD)
  - → Weights defined manually
    - Inspection of first 500 tokens as basis
- Methods were trained separately for both texts

# Evaluation

## Results

| | Luther | | Anselm | |
|---|---|---|---|---|
| Baseline | 71,163 | (65.13%) | 1,512 | (37.61%) |
| *Mapper* | **101,170** | **(92.60%)** | 2,448 | (60.90%) |
| *Rule-based* | 98,767 | (90.40%) | 2,530 | (62.94%) |
| *MultiWLD* | 96,510 | (88.33%) | **2,730** | **(67.91%)** |
| *Levenshtein* | 87,619 | (80.19%) | 2,161 | (53.76%) |
| *Mapper* → *Rules* → *WLD* | **102,193** | **(93.53%)** | 2,947 | (73.31%) |
| *Mapper* → *Rules* → *Leven* | 102,160 | (93.50%) | 2,793 | (69.48%) |
| *Mapper* → *WLD* | 102,131 | (93.48%) | **2,960** | **(73.63%)** |
| *Mapper* → *Leven* | 101,867 | (93.24%) | 2,705 | (67.29%) |
| *VARD 2* | 99,632 | (91.19%) | 2,789 | (69.38%) |

# Conclusion

- Interactive normalization tool *Norma*
  - Flexible, easily extensible
  - Support for trainable normalization algorithms
- Different normalization methods
  - Rule-based normalization: better with larger training corpora
  - Weighted Levenshtein distance: better with smaller training corpora
  - Simple wordlist mapping is always helpful
- Normalization profits even from small amounts of training data

# Thank you for listening!