Semi-supervised Contextual Historical Text Normalization

Peter Makarov **Simon Clematide**
Institute of Computational Linguistics
University of Zurich, Switzerland
makarov@cl.uzh.ch simon.clematide@cl.uzh.ch

Abstract

Historical text normalization, the task of mapping historical word forms to their modern counterparts, has recently attracted a lot of interest (Bollmann, 2019; Tang et al., 2018; Lusetti et al., 2018; Bollmann et al., 2018; Robertson and Goldwater, 2018; Bollmann et al., 2017; Korchagina, 2017). Yet, virtually all approaches suffer from the two limitations: 1) They consider a fully supervised setup, often with impractically large manually normalized datasets; 2) Normalization happens on words in isolation. By utilizing a simple generative normalization model and obtaining powerful contextualization from the target-side language model, we train accurate models with unlabeled historical data. In realistic training scenarios, our approach often leads to reduction in manually normalized data at the same accuracy levels.

1 Introduction

Text normalization is the task of mapping texts written in some non-standard variety of language L (a dialect or an earlier diachronic form) to some standardized form, typically the official modern standard variety of L (Table 1). Examples include the normalization of informal English-language tweets (Han and Baldwin, 2011); quasi-phonetic transcriptions of dialectal Swiss German (Samardžić et al., 2015); and historical documents such as religious texts in 15th-century Icelandic (Bollmann et al., 2011; Pettersson et al., 2013b; Ljubešić et al., 2016, *inter alia*).

Text can to a large extent be normalized by replacing non-standard words with their standard counterparts. Because of this often-made assumption, this task is also known as “lexical” or “spelling normalization” (Han and Baldwin, 2011; Tang et al., 2018).

There has been a lot of interest in historical and dialectal text normalization over the past years. Earlier works attempt type-level normalization by way of search for standardized words (Pettersson et al., 2013a; Bollmann, 2012). More recently, the dominant approach casts the problem as probabilistic type-level character transduction. Most commonly, a fully-supervised machine translation system transduces words in isolation (Bollmann, 2019). The use of context is limited to employing a target-side language model for an improved, contextualized decoding (Ljubešić et al., 2016; Etxeberria et al., 2016; Jurish, 2010).

In this paper, we develop simple approaches to semi-supervised contextualized text normalization. On the example of historical text normalization, we show that one can reduce the amount of supervision by leveraging unlabeled historical text and utilizing context at training. Our methods build on familiar techniques for semi-supervised learning such as generative modeling and expectation–maximization and unify previous work (search, noisy channel, contextualized decoding, neural character-level transduction) in a simple setup.

We experimentally validate the strength of our models on a suite of historical datasets. In addition to the token-level supervision scenario, we show benefits of a more economic supervision by a word-type normalization dictionary.

2 Historical text normalization

Most normalization approaches attempt to learn a function from historical to modern word types without taking context into consideration. This is based on the observation that morpho-syntactic differences between the non-standard and standard varieties (e.g. in word order, grammatical case distinctions) are negligible and normalization ambi-

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 7284–7295
July 5 - 10, 2020. ©2020 Association for Computational Linguistics

wermuttſafft	in	die	ohren	getropfft	toͤdtet	die	wuͤrme	darinnen
wermutsaft	in	die	ohren	getropft	tötet	die	würmer	darin
wormwood juice	in	the	ears	dripped	kills	the	worms	inside

Table 1: Historical text normalization. An excerpt from the RIDGES corpus of 15th–17th century German scientific writing (Odebrecht et al., 2017). Top=Early New High German, middle=Modern Standard German.

guity is often not very high. Some earlier works cast text normalization as *search* over standardized word forms (Pettersson et al., 2013a; Bollmann, 2012). Hand-crafted rules or a string-distance metric (Levenshtein, 1966) with parameters estimated from the labeled data are used to retrieve best matching standard candidates.

Another line of work follows a principled probabilistic solution: a *noisy channel model* (Shannon, 1948), which consists of a channel $p(\mathbf{x} \mid \mathbf{y})$ and a language model $p(\mathbf{y})$ (Jurish, 2010; Pettersson et al., 2013b; Samardžić et al., 2015; Etxeberria et al., 2016; Ljubešić et al., 2016; Scherrer and Ljubešić, 2016). The channel model operates at the character level and takes the form of either a character alignment model (Brown et al., 1993) or a weighted finite-state transducer (WFST, Mohri, 1997). Channel model parameters are estimated from a manually normalized corpus. The language model is often trained on external target-side data. Some works perform normalization of words in context. Jurish (2010) and Etxeberria et al. (2016) decode sequences of historical word tokens by combining a character-level channel with a *word-level* language model $p(\mathbf{y}_{1:m})$. Scherrer and Ljubešić (2016) learn a character alignment model directly over untokenized segments of historical texts.

Numerous neural approaches to text normalization (Tang et al., 2018; Lusetti et al., 2018; Bollmann et al., 2018; Robertson and Goldwater, 2018; Bollmann et al., 2017; Korchagina, 2017) learn a *discriminative model* $p(\mathbf{y} \mid \mathbf{x})$—parameterized with some generic encoder-decoder neural network—that performs the traditional character-level transduction of isolated words. The models are trained in a supervised fashion on a lot of manually labeled data. For example, Tang et al. (2018) train on tens of thousands of labeled pairs, including for varieties that share more than 75% of their vocabularies. Except Lusetti et al. (2018), who use a target-side language model to rerank base model hypotheses in context, no other approach in this group uses context in any way.

3 The role of context

If non-standard language exhibits normalization ambiguity, one would expect contextualization to reduce it. For example, historical German "deſz" in the RIDGES corpus (Odebrecht et al., 2017) normalizes to three modern word types: "das", "des" (various forms of the definite article), and "dessen" (relative pronoun *whose*). Knowing the context (e.g. whether the historical word occurs clause-initially or before a neuter noun) would help normalize "deſz" correctly. As suggested by Ljubešić et al. (2016), the accuracy of the oracle that normalizes words in isolation by always selecting their most frequent normalization upper-bounds the accuracy of non-contextual systems.

Many historical normalization corpora do not have high normalization ambiguity (Table 3). The upper bound on accuracy for non-contextual normalization is 97.0 on average (± 0.02) and is above 92.4 for every historical language that we study here, indicating that lexical normalization is a very reasonable strategy.

Even if context may sometimes not be necessary for adequately solving the task in a fully supervised manner, we would expect contextualization to lead to more accurate unsupervised and semi-supervised generative models.

4 Methods

4.1 Contextualized generative model

We start off with a generative model in the form of a noisy channel over sequences of words (Eq. 1). The channel model factorizes over non-standard words, and a non-standard word $\mathbf{x}_i$ depends only on the corresponding standardized word $\mathbf{y}_i$. The simple structure of our model follows from the lexical normalization assumption.

$$p_\theta(\overbrace{\mathbf{x}_{1:m}}^{\text{historical}}, \overbrace{\mathbf{y}_{1:m}}^{\text{modern}}) \approx \overbrace{p(\mathbf{y}_{1:m})}^{\text{language model}} \overbrace{\prod_{i=1}^{m} p_\theta(\mathbf{x}_i \mid \mathbf{y}_i)}^{\text{channel}} \tag{1}$$

Compared to a discriminative model, which would directly capture the mapping from non-standard word sequences $\mathbf{x}_{1:m}$ to standardized $\mathbf{y}_{1:m}$ without having to account for how non-standard data arise, this model offers some important advantages. First, it can be trained by maximizing marginal likelihood $p(\mathbf{x}_{1:m})$, which leads to semi-supervised learning. Second, we can use a language model estimated from arbitrary external text.

The only model parameters are the parameters θ of the channel model $p_\theta(\mathbf{x}_i \mid \mathbf{y}_i)$. The parameters of the language model $p(\mathbf{y}_{1:m})$ are held fixed.

4.2 Neural channel

The channel model $p(\mathbf{x}_i \mid \mathbf{y}_i)$ stochastically maps standardized words to non-standard words. Any type-level normalization model from §2 can be applied here (in the reverse direction from the normalization task).

For our experiments, we use the neural transducer of Makarov and Clematide (2018) as it has been shown to perform strongly on morphological character transduction tasks. Parameterized with a recurrent encoder and decoder, the model defines a conditional distribution over edits $p_\theta(\mathbf{x}, \mathbf{a} \mid \mathbf{y}) = \prod_{j=1}^{|\mathbf{a}|} p_\theta(a_j \mid a_{1:j-1}, \mathbf{y})$, where $\mathbf{y} = y_1, \ldots, y_{|\mathbf{y}|}$ is a standardized word as a character sequence and $\mathbf{a} = a_1 \ldots a_{|\mathbf{a}|}$ an edit action sequence. Using this model as a channel requires computing the marginal likelihood $p_\theta(\mathbf{x} \mid \mathbf{y})$, which is intractable due to the recurrent decoder. We approximate $p_\theta(\mathbf{x} \mid \mathbf{y})$ by $p_\theta(\mathbf{x}, \mathbf{a}^* \mid \mathbf{y})$, where $\mathbf{a}^*$ is a minimum cost edit action sequence from $\mathbf{y}$ to $\mathbf{x}$. This works well in practice as the network produces a highly peaked distribution with most probability mass placed on minimum cost edit action sequences.

4.3 Language model

We consider two language model factorizations, which lead to different learning approaches.

Neural HMM. If the language model is an n-gram language model

$$p(\mathbf{y}_{1:m}) \approx \prod_{i=1}^{m+1} p(\mathbf{y}_i \mid \mathbf{y}_{i-n+1:i-1}), \qquad (2)$$

the overall generative model has the form of an n-gram Hidden Markov Model (HMM) with transition probabilities given by the language model and emission probabilities by the channel. HMM has been proposed for this problem before but with different parameterizations (Jurish, 2010; Etxeberria et al., 2016). For simplicity, we use count-based language models in the experiments. Full neural parametarization can be achieved with n-gram feedforward neural language models (Bengio et al., 2003).

RNN LM-based model. Our second language model is a word-level recurrent neural language model (RRN-LM, Mikolov, 2012). It does not make any independence assumptions, which increases expressive power yet precludes exact inference in the generative model.

4.4 Expectation–maximization

Let U be a set of unlabeled non-standard sentences, $V_\mathbf{x}$ the set of non-standard word types in U, and $V_\mathbf{y}$ the vocabulary of the standardized variety. In the unsupervised case, we train by maximizing the marginal likelihood of U with respect to the channel parameters θ:

$$L_U(U, \theta) = \sum_{\mathbf{x}_{1:m} \in U} \log \sum_{\mathbf{y}_{1:m} \in V_\mathbf{y}^m} p_\theta(\mathbf{x}_{1:m},\ \mathbf{y}_{1:m}) \qquad (3)$$

For an n-gram neural HMM, this can be solved using generalized expectation–maximization (GEM, Neal and Hinton, 1998; Berg-Kirkpatrick et al., 2010). We compute the E-step with the forward–backward algorithm. In the M-step, given the posterior $p(\mathbf{y} \mid \mathbf{x})$ for each non-standard word type $\mathbf{x}$, we maximize the following objective with respect to θ with a variant of stochastic gradient ascent:

$$L_M(U, \theta) = \sum_{\mathbf{x} \in V_\mathbf{x}} \sum_{\mathbf{y} \in V_\mathbf{y}} p(\mathbf{y} \mid \mathbf{x}) \log p_\theta(\mathbf{x} \mid \mathbf{y}) \qquad (4)$$

GEM provably increases the marginal likelihood.

We train the RNN LM-based model with hard expectation–maximization (hard EM, Samdani et al., 2012). This is a simple alternative to approximate inference. Hard EM does not guarantee to increase the marginal likelihood, but often works in practice (Spitkovsky et al., 2010). The difference from GEM is the E-step. To compute it, we decode U with beam search. Let $B = \bigcup_{\mathbf{x}_{1:m} \in U} \{\mathbf{y}_{1:m} \in V_\mathbf{y}^m \mid \mathbf{y}_{1:m}$ is in the beam for $\mathbf{x}_{1:m}\}$. We set the posterior $p(\mathbf{Y} = \mathbf{y} \mid \mathbf{X} = \mathbf{x})$ to be proportional to the sum of the probabilities of sentence-wise normalizations from B where $\mathbf{x}$ gets normalized as $\mathbf{y}$.

Semi-supervised training. We linearly combine the maximum likelihood (MLE) of the set $S = \{(\mathbf{x}, \mathbf{y})_i\}_{i=1}^n$ of labeled normalization pairs with the marginal likelihood of U (Eq. 3):

$$L(S, U, \theta) = \sum_{(\mathbf{x},\mathbf{y}) \in S} \log p_\theta(\mathbf{x} \mid \mathbf{y}) + \lambda L_U(U, \theta) \quad (5)$$

$\lambda \geq 0$ controls how much information from U flows into the parameter update. The difference from the unsupervised case is that the M-step computes Eq. 4 scaled with λ and the MLE of S.

In practice, we initially pretrain the channel on the labeled data and then move to full semi-supervised training with some non-zero λ fixed for the rest of training.

4.5 Proposal of standardized candidates

Candidate set heuristic. Performing EM with the full modern vocabulary $V_\mathbf{y}$ as the set of possible normalization candidates is vastly impractical: The forward–backward algorithm runs in $O(m|V_\mathbf{y}|^n)$ time. In related tasks, this has lead to training heuristics such as iterative EM (Ravi and Knight, 2011). To keep this computation manageable, we propose generating a candidate set $C(\mathbf{x})$ of k modern words for each non-standard word $\mathbf{x}$. To this end, we use approximate nearest neighbor search with edit distance (Hulden, 2009). The algorithm efficiently searches through an FST, which encodes a part of the vocabulary, with the A^* search. We encode different word frequency bands of the vocabulary as separate FSTs, which we search in parallel. We rerank the candidates taking into account non-standard and standardized words' relative frequencies (see Appendix). Thus, all summations and maximizations over $V_\mathbf{y}$ are performed over the reduced set $C(\mathbf{x})$.

Our heuristic embodies normalization by search (§2) and could be replaced with a more informed search and reranking algorithm (Bollmann, 2012; Baron and Rayson, 2008).

Candidate generation with direct model. The candidate set heuristic is too restrictive. It is hard to achieve perfect coverage at manageable candidate set sizes (e.g. if $\mathbf{x}$ and target $\mathbf{y}$ have no characters in common as e.g. historical German "fy" ↦ "sie" (*they*)). Worse still, this approach completely fails if the target $\mathbf{y}$ does not appear in the corpus. This could be because the corpus is small (e.g. most Wikipedias); rich morphology or orthographic conventions lead to a vast number of word types (e.g. Hungarian); or the target word is not even attested in the standardized variety (e.g. "czuhant" ↦ *"zehant" (*immediately*) in the Anselm historical German corpus (Krasselt et al., 2015)). We, therefore, also consider candidate generation with a direct model $q_\phi(\mathbf{y} \mid \mathbf{x})$.

Algorithm 1 GEM training (§4.4)

Full version uses restarts and candidate pruning (see Appendix).

Input: Unlabeled dataset U, labeled dataset S, development set, number of modern candidates k to generate, number of EM epochs K, mixture parameter λ combining the unsupervised and supervised objectives.

```
Compute $k$ candidates $C(\mathbf{x})$ for each non-standard word
  type $\mathbf{x} \in V_\mathbf{x}$ from $U$ (by either method in § 4.5).
Randomly initialize channel parameters $\theta^{(0)}$.
if labeled dataset $S \neq \emptyset$ then pretrain $\theta^{(0)}$ on $S$.
for epoch $t \leftarrow 1$ to $K$ do
    E-step:
    $Q \leftarrow \mathbf{0}^{k \times |V_\mathbf{x}|}$
    Compute channel scores $p_{\theta^{(t-1)}}(\mathbf{x} \mid \mathbf{y})$ for all $\mathbf{x} \in$
      $V_\mathbf{x}$ and $\mathbf{y} \in C(\mathbf{x})$. (Use uniform scores if $t = 1$ and
      $S = \emptyset$.)
    for non-standard word sequence $\mathbf{x}_{1:m} \in U$ do
        Run forward–backward or beam search (§4.4) to
          compute each word's posterior $p(\mathbf{Y}_i \mid \mathbf{x}_{1:m})$.
        for position $i \leftarrow 1$ to $m$ do
            $Q(\mathbf{y}, \mathbf{x}_i) \leftarrow Q(\mathbf{y}, \mathbf{x}_i) + p(\mathbf{Y}_i = \mathbf{y} \mid \mathbf{x}_{1:m})$
              for all $\mathbf{y} \in C(\mathbf{x}_i)$.
    Normalize: $p(\mathbf{y} \mid \mathbf{x}) \leftarrow Q(\mathbf{y}, \mathbf{x}) / \sum_\mathbf{y} Q(\mathbf{y}, \mathbf{x})$ for
      all $\mathbf{x} \in V_\mathbf{x}$ and $\mathbf{y} \in C(\mathbf{x})$.
    M-step:
    Start training from $\theta^{(t-1)}$ and use $p(\mathbf{y} \mid \mathbf{x})$ in unsu-
      pervised objective $L_M(U, \theta)$ (Eq. 4):
    $\theta^{(t)} \leftarrow \operatorname{argmax}_\theta \sum_{(\mathbf{x},\mathbf{y}) \in S} \log p_\theta(\mathbf{x} \mid \mathbf{y}) + \lambda L_M(U, \theta)$
return $\theta^{(t)}$ leading to best accuracy on development set
```

We bootstrap a direct model from a contextualized generative model. We fit it by minimizing the cross-entropy of the direct model relative to the posterior of the generative model $p(\mathbf{y} \mid \mathbf{x})$. For a semi-supervised generative model, this combines with the MLE of the labeled set S ($\kappa \geq 0$):

$$\begin{aligned} L(S, U, \phi) = & \sum_{(\mathbf{x},\mathbf{y}) \in S} \log q_\phi(\mathbf{y} \mid \mathbf{x}) \\ & + \kappa \sum_{\mathbf{x} \in V_\mathbf{x}} \sum_{\mathbf{y} \in V_\mathbf{y}} p(\mathbf{y} \mid \mathbf{x}) \log q_\phi(\mathbf{y} \mid \mathbf{x}), \end{aligned} \quad (6)$$

Any type-level normalization model from prior work could be used here. We choose the direct model to be a neural transducer, like the channel. It generates candidates using beam search.

4.6 Prediction and reranking

We consider two ways of sentence-wise decoding with our generative models. The first uses the maximum a posteriori (MAP) decision rule, which finds a normalization that maximizes the posterior $p(\mathbf{y}_{1:m} \mid \mathbf{x}_{1:m})$. Depending on the factorization of the language model, we solve this exactly (with the Viterbi algorithm) or approximately (with beam search).

The other approach is to learn a reranker model on the development set. The model rescores sentence hypotheses $\hat{\mathbf{y}}_{1:m}$ generated by the base model (with k-best Viterbi or beam search). It uses rich non-local features—the hypothesis' scores under a word- and character-level RNN language models—as well as length-normalized base model score, mean out-of-vocabulary rate and edit distance from $\mathbf{x}_{1:m}$ (see Appendix). We implement a PRO reranker (Hopkins and May, 2011) that uses hamming loss.

5 Experiments

For our experiments, we use eight datasets compiled by various researchers (Pettersson, 2016; Ljubešić et al., 2016; Bollmann, 2018) from historical corpora (Tables 2 and 3).[1]

Seven languages are Indo-European: Germanic (English, German, Icelandic, and Swedish), Romance (Portuguese and Spanish), and Slavic (Slovene). Additionally, we experiment with Hungarian, a Finno-Ugric language. From the Slovene dataset, we only use the collection of the older and more challenging texts in the Bohorič alphabet.

The data are of different genres (letters, religious and scientific writings). The earliest texts are in 14th-c. Middle English. In many datasets, the proportion of identity normalizations is substantial. The smallest word overlap is in the Hungarian data (18%), the largest is in English (75%).

All corpora are tokenized and aligned at the segment and token level. For some datasets, either segments do not coincide with grammatical sentences, or the data have no segment boundaries at all (e.g. Hungarian or Icelandic). In such cases, to make input amenable to training with context, we resort to sentence splitting on punctuation marks. We also split very long segments to ensure the maximum segment length of fifty words.

Algorithm 2 MAP decoding or reranking (§4.6)

Input: Non-standard word sequence $\mathbf{x}_{1:m}$, number of modern candidates c to generate, number of sentence hypotheses k to generate.

```
for $i \leftarrow 1$ to $m$ do
    Compute $c$ candidates $C(\mathbf{x}_i)$ (§4.5).
    Compute channel scores $p(\mathbf{x}_i \mid \mathbf{y})$ for all $\mathbf{y} \in C(\mathbf{x}_i)$.
if MAP decoding then
    Decode using Viterbi or beam search:
    $\mathbf{y}^*_{1:m} \leftarrow \operatorname*{argmax}_{\mathbf{y}_{1:m} \in V_{\mathbf{y}}^m} p(\mathbf{x}_{1:m}, \mathbf{y}_{1:m})$
else
    Produce $k$ hypotheses using $k$-best Viterbi or beam:
    $R(\mathbf{x}_{1:m}) \leftarrow \operatorname*{argmax}_{R \subseteq V_{\mathbf{y}}^m \text{ s.t. } |R|=k} \sum_{\mathbf{y}_{1:m} \in R} p(\mathbf{x}_{1:m}, \mathbf{y}_{1:m})$
    Rerank using a linear model with non-local features:
    $\mathbf{y}^*_{1:m} \leftarrow \operatorname*{argmax}_{\hat{\mathbf{y}}_{1:m} \in R(\mathbf{x}_{1:m})} \psi^\top g(\mathbf{x}_{1:m}, \hat{\mathbf{y}}_{1:m})$
return best hypothesis $\mathbf{y}^*_{1:m}$
```

Token alignment is largely one-to-one, with rare exceptions. Clitization and set phrases (e.g. German “muſtu” ↦ “musst du” (*you must*), “aller handt” ↦ “allerhand” (*every kind of*)) are common causes for many-to-one alignments, which our models fail to capture properly.

State-of-the-art. We compare our models to the strongest models for historical text normalization:

- the Norma tool (Bollmann, 2012), which implements search over standardized candidates; and
- the character-level statistical machine translation model (cSMT, Ljubešić et al., 2016), which uses the Moses toolkit (Koehn et al., 2007). This approach estimates a character n-gram language model on external data and fits a MERT reranker model (Och, 2003) on the development set.

According to Bollmann (2019), Norma performs best in the low-resource setting ($\leq$ 500 labeled tokens), and cSMT should be preferred in all other data conditions. Norma's strong performance in the low-resource scenario derives from the fact that searching for candidates can be fairly easy for some languages e.g. English. The reranker trained on the development set is key to cSMT's strength.

Realistic low-resource setting. Our contextualized models are particularly appealing when labeled data are limited to at most a couple of thousand annotated word pairs. This would be the most common application scenario in practice, and approaches requiring tens of thousands of training

[1] The datasets are featured in the large-scale study of Bollmann (2019), who conveniently provides most data in a unified format at https://github.com/coastalcph/histnorm/.

	period	train	dev	reference
de	1482–1652	41.9	9.7	Odebrecht et al. (2017)
en	1386–1698	147.8	16.3	Markus (1999)
es	15th–19th c.	97.3	11.7	Vaamonde (2017)
is	15th c.	49.6	6.1	Rögnvaldsson et al. (2012)
pt	15th–19th c.	222.5	26.7	Vaamonde (2017)
hu	1440–1541	134.0	16.7	Simon (2014)
sl	1750–1840s	50.0	5.8	Erjavec (2012)
sv	1527–1812	24.5	2.2	Fiebranz et al. (2011)

Table 2: Historical normalization datasets. Train and development set sizes in thousands of tokens.

samples would be ruled out as unrealistic. We, therefore, experiment with small labeled training set sizes n ranging from 500 to 5K. Additionally, we consider the unsupervised scenario ($n = 0$), which might be less relevant practically (even a small amount of labeled data might lead to substantial improvement) but allows us to demonstrate most directly the advantage of our approach.

To keep the experiments close to the real-life application scenario (Kann et al., 2019), we additionally cap the size of the development set at 2K tokens. Otherwise, we require that the development set have 500 tokens more than the labeled set S to ensure that we validate on not too small a number of word types (e.g. at 1K tokens, we get only about 600 word types on average).

Finally, the unlabeled set U comprises non-standard word sequences from all the remaining non-test data. Our sampled development sets are much smaller compared to the original development set from the official data split. Not to waste any data, we also include the historical part of the rest of the original development set into the unlabeled set U. The labeled training set S is sampled uniformly at random from U with targets.

Semi-supervised training with type-level normalization dictionary. Supervision by type-level dictionary (as opposed to token-level annotations) is a simple and effective way of reducing the amount of manually labeled data (Garrette et al., 2013). We simulate type-level normalization dictionary construction by selecting d most frequent non-standard word types from the original training set. We build a labeled set S by pairing them with the most frequent standard word types that they normalize to. We experiment on German and Slovene. We use a development set of 500 tokens.

Experimental setup. We use Wikipedia dumps for training language models and the candidate set heuristic. For the neural HMM, we fit count-based bigram language models using KenLM (Heafield et al., 2013). All RNN language models are parameterized with a Long Short-Term Memory cell (Hochreiter and Schmidhuber, 1997) and use dropout regularization (Zaremba et al., 2014). The HMMs use $C(\mathbf{x})$ of 150 candidates, the RNN LM-based models use 50 candidates.

	I	NC	H	Seg	C@50	C@150
de	43.8	95.6	.155	Y	91.0	94.2
en	74.9	98.0	.087	Y	94.7	96.3
es	72.9	97.2	.125	Y	94.3	95.8
hu	17.6	98.0	.075	N	78.1	81.2
is	46.7	92.4	.213	N	84.3	86.2
pt	65.3	97.4	.129	Y	92.3	94.7
sl	41.1	98.3	.057	N	90.7	92.1
sv	59.9	99.2	.026	Y	89.8	91.5
avg	52.8	97.0	.108		89.2	91.5

Table 3: Historical normalization datasets (cont.). I=proportion of identity normalizations, NC=accuracy of the non-contextual oracle that selects the most frequent normalization for each historical word, H=normalization entropy, Seg=whether the dataset is sentence-segmented (Y=yes, N=no), C@z=$C(\mathbf{x})$ coverage of the dataset at z standard candidates per historical word type $\mathbf{x}$. All statistics are computed on the official training sets.

We train for 15 iterations of EM setting $\lambda = \kappa = 0.8$ throughout. We optimize the neural channel with mini-batched AdaDelta (Zeiler, 2012).

We set the beam size of the RNN LM-based models to four for both final decoding and the E-step. For reranking, the base HMMs output 150 k-best sentence hypotheses and the RNN LM-based models output the beam. The reranker models are trained with the perceptron algorithm.

The direct models are trained with AdaDelta. We decode them with beam search and rerank the beam with a PRO reranker using the channel and direct model scores and relative frequency as features. We use the top two reranked candidates as the new candidate set. We refer the reader to the Appendix for further details on training.

We train Norma and cSMT on our data splits using the training settings suggested by the authors.

6 Discussion

The semi-supervised contextualized approach results in consistent improvements for most languages and labeled data sizes (Tables 4 and 5).

	de	en	es	hu	is	pt	sl	sv	avg
identity	44.36	75.29	73.40	17.53	47.62	65.19	40.74	58.59	52.84
best supervised	88.22	95.24	95.02	91.70	87.31	95.18	93.30	91.13	92.14
					$n = 0$				
neural HMM	77.49	87.94	88.75	68.25	77.44	82.63	80.84	75.82	79.89
+rerank	81.02	89.92	89.29	68.66	77.09	**84.92**	83.52	77.19	81.45
+direct+rerank	79.28	89.21	**89.71**	**70.54**	78.91	83.81	84.67	79.88	**82.00**
RNN LM-based	80.70	90.47	87.04	57.86	73.20	82.95	81.01	78.30	78.94
+rerank	80.80	90.18	86.69	57.91	73.75	83.32	82.55	78.16	79.17
+direct+rerank	**81.15**	**91.06**	88.16	63.75	**79.87**	84.27	**85.08**	**81.61**	81.87
					$n = 500$				
Norma (Bollmann, 2012)	74.00	84.24	86.41	62.57	76.56	81.62	78.04	77.77	77.65
cSMT (Ljubešić et al., 2016)	76.28	85.17	88.88	68.31	79.00	83.00	83.67	81.66	80.75
neural HMM	80.34	88.97	90.34	69.00	78.38	86.88	85.23	80.08	82.40
+rerank	82.89	90.81	91.00	69.88	78.68	**88.53**	85.75	81.62	83.64
+direct+rerank	82.55	90.48	**91.55**	**71.04**	80.90	87.75	86.68	**84.55**	**84.44**
RNN LM-based	82.33	91.12	89.78	62.50	75.93	85.84	83.42	80.88	81.48
+rerank	82.42	91.19	90.14	62.21	75.93	86.01	83.85	81.08	81.60
+direct+rerank	**83.23**	**91.66**	90.18	67.70	**81.63**	87.26	**87.54**	84.11	84.16
					$n = 1,000$				
Norma (Bollmann, 2012)	75.52	85.27	87.94	64.84	77.49	83.56	79.16	79.35	79.14
cSMT (Ljubešić et al., 2016)	78.91	86.89	90.44	70.35	80.32	85.14	85.53	84.82	82.85
neural HMM	80.91	89.51	90.82	70.55	79.86	87.52	85.11	81.95	83.28
+rerank	83.34	91.18	**92.28**	70.95	79.92	**89.06**	85.81	83.19	84.47
+direct+rerank	82.68	91.18	92.15	**73.17**	**82.38**	88.98	87.32	85.78	**85.45**
RNN LM-based	82.84	91.35	90.45	65.02	77.80	87.03	83.53	82.77	82.60
+rerank	83.14	91.23	90.49	64.73	78.68	87.41	84.77	83.33	82.97
+direct+rerank	**83.39**	**91.76**	91.22	69.13	81.22	87.65	**87.60**	**85.86**	84.73

Table 4: Test set results for unsupervised and semi-supervised (500 and 1,000 tokens) settings. Best results within each category are highlighted in bold and, where applicable, are statistically significant compared to cSMT ($p < 0.05$, McNemar's test). Best supervised results are quoted from Bollmann (2019).

Compared to cSMT, an average error reduction ranges from 19% ($n = 500$) to almost 3% ($n =$ 5K) or 8% excluding Hungarian, the language on which the models perform worst. Reranking provides an important boost (almost 5% error reduction compared to the base model, and almost 8% for neural HMMs), and bootstrapping direct model candidates results in even better performance (almost 14% error reduction).

Unsupervised case. Remarkably, with no labeled training data (and only a 500-token labeled development set), the best configuration achieves 88.4% of the top scores reported for fully supervised models (Table 2 of Bollmann (2019)). It outperforms the Norma baseline trained on $n =$ 1K labeled samples, reducing its error by almost 4%.

Effects of unlabeled dataset size. We typically see strong performance for languages where the unlabeled dataset U is large ($\approx$ official training and development sets together, Table 2). This includes English, that shows little ambiguity (Table 3) and so would be expected to profit less from contextualization.

Effects of the modern corpus and preprocessing. The size and coverage of the Wikipedia dump (Table 3) for Icelandic and particularly Hungarian degrade the models' performance and are likely the key reason why cSMT outperforms all contextual models for Hungarian as the labeled dataset increases ($n =$ 2.5K and $n =$ 5K), despite the large amount of unlabeled Hungarian text. The RNN LM-based models are hit worst due to the poorest coverage. The lack of original segment boundaries (Table 3, Icelandic is only partially segmented) further exacerbates performance.

Remarkably, the overall approach works despite language models and candidate sets using out-of-domain standardized data. Leveraging in-domain data such as collections of literary works from the time period of the source historical text could lead to even better performance (Berg-Kirkpatrick

	de	en	es	hu	is	pt	sl	sv	avg	avg\hu
identity	44.36	75.29	73.40	17.53	47.62	65.19	40.74	58.59	52.84	57.87
best supervised	88.22	95.24	95.02	91.70	87.31	95.18	93.30	91.13	92.14	92.19
$n = 2.5$K										
cSMT (Ljubešić et al., 2016)	82.08	88.66	91.47	**75.85**	82.06	87.82	**88.32**	**87.28**	85.56	86.95
neural HMM	82.16	89.72	92.07	70.73	81.00	88.54	85.56	82.38	84.02	85.92
+rerank	84.64	91.58	92.96	71.52	80.77	89.99	86.48	84.44	85.30	87.27
+direct+rerank	**84.85**	91.48	**93.26**	74.34	81.86	**90.02**	88.04	86.80	**86.33**	88.04
RNN LM-based	83.55	91.79	92.11	67.70	79.19	88.38	85.01	84.06	83.97	86.30
+rerank	83.87	91.84	92.56	67.94	79.64	88.36	85.68	84.31	84.27	86.61
+direct+rerank	84.83	**92.16**	92.57	72.45	82.91	89.54	†88.12	86.22	86.10	**88.05**
direct model (from RNN LM)	83.59	91.39	91.79	72.57	**83.40**	89.75	87.92	†86.94	85.92	87.83
$n = 5$K										
cSMT (Ljubešić et al., 2016)	83.50	90.02	92.52	**79.16**	83.09	89.83	**89.40**	†88.51	87.04	88.16
neural HMM	83.15	89.51	92.74	71.92	80.88	89.38	86.52	83.75	84.73	86.56
+rerank	85.19	91.37	93.85	72.72	81.05	**90.99**	87.52	85.28	86.00	87.89
+direct+rerank	85.76	91.90	**94.24**	75.74	83.42	90.79	†89.18	88.29	**87.41**	**89.08**
RNN LM-based	84.50	92.11	92.64	69.03	79.79	89.07	85.51	85.01	84.71	86.95
+rerank	85.04	92.15	92.36	69.34	79.97	89.48	86.13	85.49	85.00	87.23
+direct+rerank	**85.82**	**92.41**	92.32	74.39	83.77	89.86	88.93	87.71	86.90	88.69
direct model (from RNN LM)	85.03	91.84	92.54	75.86	**83.97**	90.21	88.48	**88.56**	87.06	88.66

Table 5: Test set results for semi-supervised setting (2,500 and 5,000 tokens). Best results within each category are in bold. The differences between cSMT and the best of the proposed models are statistically significant ($p < 0.05$, McNemar's test) unless marked with †. Best supervised results are quoted from Bollmann (2019).

et al., 2013).

Candidate generation with direct model. Generating candidates with the direct model leads to large gains for languages with poor coverage (Icelandic and Hungarian RNN LM-based models see an average error reduction of over 20% and 14% respectively). At larger labeled dataset sizes (Table 5), bootstrapping a direct model and reranking its output without context becomes an effective strategy (Icelandic, Portuguese).

Normalization ambiguity. We would expect languages with higher normalization ambiguity to profit from contextualization (Ljubešić et al., 2016). German, Portuguese, and Spanish gain even in the most competitive semi-supervised 5K condition, consistent with the amount of ambiguity they exhibit (Table 3). Losses and modest gains are observed for languages with the lowest ambiguity rates (Slovene, Swedish).

We look at the accuracies on unambiguous and ambiguous normalizations (Figure 1). The contextual model consistently outperforms cSMT on ambiguous tokens, often by a wide margin and even when cSMT is better overall (Slovene). An extreme case is German at $n = 5$K, where the two approaches perform similarly on unambiguous tokens, yet cSMT suffers considerably on ambiguous ones (38% error reduction by the neural HMM). German ranks second by normalization ambiguity (Table 3).

Type-level normalization dictionary. We observe gains equivalent to using a token-level dataset of at least double the dictionary size (Table 6). Slovene profits a lot from dictionary supervision, with 1K-type model performing close to the 5K-token model.

n	de	sl
250	82.08	85.43
1K	83.29	86.33
	RNN LM-based	neural HMM

Table 6: Test set results for supervision by type-level normalization dictionary.

Shortcomings of the approach. The general problem of our approach, as well as most approaches that we build on, is reliance on gold tokenization. Overall, we have faced minor issues with tokenization (one notable example is Swedish where 0.6% of the target-side test data are words with a colon for which we fail to retrieve candidates from Wikipedia). Tokenization remains a challenge for normalization of unpreprocessed non-standard data (Berg-Kirkpatrick et al., 2013).

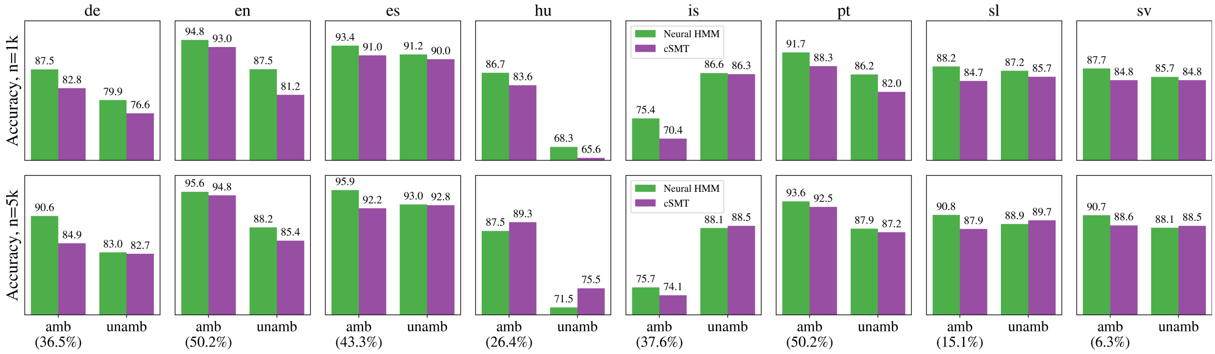

Figure 1: Test set performance breakdown by unambiguous and ambiguous tokens in $n = 1\text{K}$ (top) and $n = 5\text{K}$ (bottom) semi-supervised conditions. Comparisons are between neural HMM (+direct+rerank, **green**) and cSMT (Ljubešić et al., 2016, **violet**). Ambiguity (=whether a historical word normalizes into more than one standard word type) is computed on the official training data.

7 Future work

Clearly, one can simultaneously use both methods of candidate generation (§4.5). We leave it for future work to verify whether this leads to an improved performance.

Computing the posterior $p(\mathbf{y} \mid \mathbf{x})$ in both generative models is hard, which is why we are forced to reduce the number of admissible candidates $\mathbf{y}$ and, in the case of the RNN LM-based model, approximate the posterior with maximization. This problem can be addressed in a principled way by using variational inference (Jordan et al., 1999), a framework for approximate inference that deals with intractable distributions. We leave it for future work to validate its effectiveness for this problem.

As noted earlier, it is a simplification to assume that non-standard text is tokenized. Being able to normalize across token boundaries (by merging multiple non-standard tokens or splitting one into multiple standardized ones) is crucial for tackling real-world text normalization tasks and related problems such as optical character recognition error correction. An appealing direction for future work would be developing a joint model for text tokenization and normalization. One family of latent-variable models that would be suitable for this task are segmental recurrent neural networks (SRNNs, Kong et al., 2016). SRNNs explicitly model input segmentation and have been successfully applied to online handwriting recognition, Chinese word segmentation, joint word segmentation and part-of-speech tagging (Kong et al., 2016; Kawakami et al., 2019).

8 Conclusion

This paper proposes semi-supervised contextual normalization of non-standard text. We focus on historical data, which has gained attention in the digital humanities community over the past years. We develop simple contextualized generative neural models that we train with expectation–maximization. By leveraging unlabeled data and accessing context at training time, we train accurate models with fewer manually normalized training samples. No labeled training data are necessary to achieve 88.4% of the best published performance that uses full training sets. Strong gains are observed for most of the considered languages across realistic low-resource settings (up to 5K labeled training tokens). The techniques developed here readily apply to other types of normalization data (e.g. informal, dialectal). We will make our implementation publicly available.[2]

Acknowledgments

We thank the reviewers for helpful comments and Eva Pettersson and Manfred Markus for their help with the English dataset. This work has been supported by the Swiss National Science Foundation under grant CR-SII5_173719.

References

Alistair Baron and Paul Rayson. 2008. VARD2: A tool for dealing with spelling variation in historical corpora. In *Postgraduate conference in corpus linguistics*.

[2] https://github.com/ZurichNLP/acl2020-historical-text-normalization

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*.

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *NAACL-HLT*.

Taylor Berg-Kirkpatrick, Greg Durrett, and Dan Klein. 2013. Unsupervised transcription of historical documents. In *ACL*.

Marcel Bollmann. 2012. (semi-)automatic normalization of historical texts using distance measures and the Norma tool. In *Second Workshop on Annotation of Corpora for Research in theHumanities (ACRH-2)*.

Marcel Bollmann. 2018. *Normalization of historical texts with neural network models*. Sprachwissenschaftliches Institut, Ruhr-Universität.

Marcel Bollmann. 2019. A large-scale comparison of historical text normalization systems. In *NAACL-HLT*.

Marcel Bollmann, Joachim Bingel, and Anders Søgaard. 2017. Learning attention for historical text normalization by learning to pronounce. In *ACL*.

Marcel Bollmann, Florian Petran, and Stefanie Dipper. 2011. Rule-based normalization of historical texts. In *Proceedings of the Workshop on Language Technologies for Digital Humanities and Cultural Heritage (LaTeCH)*.

Marcel Bollmann, Anders Søgaard, and Joachim Bingel. 2018. Multi-task learning for historical text normalization: Size matters. In *Proceedings of the Workshop on Deep Learning Approaches for Low-Resource NLP*.

Peter F. Brown, Stephen A. Della-Pietra, Vincent J. Della-Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation. *Computational Linguistics*.

Tomaz Erjavec. 2012. The goo300k corpus of historical Slovene. In *LREC*.

Izaskun Etxeberria, Iñaki Alegria, Larraitz Uria, and Mans Hulden. 2016. Evaluating the noisy channel model for the normalization of historical texts: Basque, Spanish and Slovene. In *LREC*.

Rosemarie Fiebranz, Erik Lindberg, Jonas Lindström, and Maria Ågren. 2011. Making verbs count: the research project 'Gender and Work' and its methodology. *Scandinavian Economic History Review*, 59(3).

Dan Garrette, Jason Mielens, and Jason Baldridge. 2013. Real-world semi-supervised learning of postaggers for low-resource languages. In *ACL*.

Bo Han and Timothy Baldwin. 2011. Lexical normalisation of short text messages: Makn sens a #twitter. In *ACL*.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H Clark, and Philipp Koehn. 2013. Scalable modified kneser-ney language model estimation. In *ACL*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8).

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *EMNLP*.

Mans Hulden. 2009. Fast aproximate string matching with finite automata. *Procesamiento del lenguaje natural*, 43.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. 1999. An introduction to variational methods for graphical models. *Machine learning*.

Bryan Jurish. 2010. More than words: Using token context to improve canonicalization of historical german. *Journal For Language Technology And Computational Linguistics*.

Katharina Kann, Kyunghyun Cho, and Samuel R Bowman. 2019. Towards realistic practices in low-resource natural language processing: The development set. In *EMNLP*.

Kazuya Kawakami, Chris Dyer, and Phil Blunsom. 2019. Learning to discover, ground and use words with segmental neural language models. In *ACL*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL*.

Lingpeng Kong, Chris Dyer, and Noah A Smith. 2016. Segmental recurrent neural networks. In *ICLR*.

Natalia Korchagina. 2017. Normalizing medieval german texts: from rules to deep learning. In *Proceedings of the NoDaLiDa 2017 Workshop on Processing Historical Language*.

Julia Krasselt, Marcel Bollmann, Stefanie Dipper, and Florian Petran. 2015. *Guidelines für die Normalisierung historischer deutscher Texte / Guidelines for Normalizing Historical German Texts*. Bochumer Linguistische Arbeitsberichte: 15.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics doklady*, 10(8).

Nikola Ljubešić, Katja Zupan, Darja Fišer, and Tomaž Erjavec. 2016. Normalising Slovene data: historical texts vs. user-generated content. In *KONVENS*.

Massimo Lusetti, Tatyana Ruzsics, Anne Göhring, Tanja Samardžić, and Elisabeth Stark. 2018. Encoder-decoder methods for text normalization. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*.

Peter Makarov and Simon Clematide. 2018. Neural transition-based string transduction for limited-resource setting in morphology. In *COLING*.

Manfred Markus. 1999. *Manual of ICAMET (Innsbruck Computer Archive of Machine-Readable English Texts)*. Leopold-Franzens-Universitat Innsbruck.

Tomáš Mikolov. 2012. *Statistical language models based on neural networks*. Ph.D. thesis, Brno University of Technology, Czech Republic.

Mehryar Mohri. 1997. Finite-state transducers in language and speech processing. *Computational linguistics*.

Radford M Neal and Geoffrey E Hinton. 1998. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*. Springer.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *ACL*.

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

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research*.

Eva Pettersson. 2016. *Spelling normalisation and linguistic analysis of historical text for information extraction*. Ph.D. thesis, Acta Universitatis Upsaliensis.

Eva Pettersson, Beáta Megyesi, and Joakim Nivre. 2013a. Normalisation of historical text using context-sensitive weighted Levenshtein distance and compound splitting. In *NODALIDA*.

Eva Pettersson, Beáta Megyesi, and Jörg Tiedemann. 2013b. An SMT approach to automatic annotation of historical text. In *Proceedings of the Nodalida Workshop on Computational Historical Linguistics*.

Sujith Ravi and Kevin Knight. 2011. Deciphering foreign language. In *ACL-HLT*.

Alexander Robertson and Sharon Goldwater. 2018. Evaluating historical text normalization systems: How well do they generalize? In *NAACL*.

Eiríkur Rögnvaldsson, Anton Karl Ingason, Einar Freyr Sigurdhsson, and Joel Wallenberg. 2012. The icelandic parsed historical corpus (icepahc). In *LREC*.

Tanja Samardžić, Yves Scherrer, and Elvira Glaser. 2015. Normalising orthographic and dialectal variants for the automatic processing of swiss german. In *Language and Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics*.

Rajhans Samdani, Ming-Wei Chang, and Dan Roth. 2012. Unified expectation maximization. In *NAACL-HLT*.

Yves Scherrer and Nikola Ljubešić. 2016. Automatic normalisation of the Swiss German ArchiMob corpus using character-level machine translation. In *KONVENS*.

Claude Elwood Shannon. 1948. A mathematical theory of communication. *Bell system technical journal*.

Eszter Simon. 2014. Corpus building from Old Hungarian codices. In *The evolution of functional left peripheries in Hungarian syntax*.

Valentin I Spitkovsky, Hiyan Alshawi, Daniel Jurafsky, and Christopher D Manning. 2010. Viterbi training improves unsupervised dependency parsing. In *CoNLL*.

Gongbo Tang, Fabienne Cap, Eva Pettersson, and Joakim Nivre. 2018. An evaluation of neural machine translation models on historical spelling normalization. In *COLING*.

Gael Vaamonde. 2017. Userguide for digital edition of texts in PS Post Scriptum.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Matthew D Zeiler. 2012. ADADELTA: an adaptive learning rate method. arXiv:1212.5701.

Appendix

Candidate set generation. We extract the text from the Wikipedia dumps with Wikiextractor,[3] tokenize it with multifit,[4] and preprocess the tokens using the procedure of Bollmann (2019). We remove all words containing characters that have a relative frequency ≤ 0.0001. We partition the types into frequency bins with upper limits of $1, 2, 5, 10, 100, 10^3, 10^4, 10^5, 10^6, \infty$. For each frequency bin, we extract an initial candidate list based on minimum edit distance, $\textsc{ed}(h, m)$. For the computation of edit distance, we ignore all diacritical marks of the letters (by ignoring composing characters according to the Unicode standard). We rerank our modern candidates based on a frequency ratio $f_{h,m} = \frac{\#h}{\#m}$ that punishes rare

[3] https://github.com/attardi/wikiextractor

[4] https://github.com/n-waves/multifit

modern candidates for frequent historical words. Using the smallest supervised development set (500 tokens)—required by all our experiments—we compute coverage over all languages, and set the log base for squeezing the frequency ratio to 200. Finally, a penalty for rare modern forms based on their frequency is added. The score for reranking the candidate list is: $s_{h,m} = \mathrm{ED}(h,m) + max(log_{200}(f_{h,m}), 0) + \frac{1}{\#(m)}$.

Neural channel training. In every M-step (the maximization of Eq. 4), we start training from the previous EM iteration's best parameters $\theta^{(t-1)}$ and train for 25 epochs with 15 epochs of patience. We optimize the parameters with AdaDelta (Zeiler, 2012) using mini-batches of size 20. We do not update on candidates whose posterior probability is below $\epsilon = 0.01 \cdot \lambda^{-1}$. If the development set scores $\sum_{(\mathbf{x},\mathbf{y}) \in \text{dev}} p_\theta(\mathbf{x} \mid \mathbf{y})$ do not increase compared to the previous EM iteration, we restart training from randomly initialized parameters and using the type-level posterior probability from the best generative model found so far. That model, decoded using MAP decoding, has so far produced the highest normalization accuracy on the development set. In the semi-supervised scenario, we initially pretrain the channel for 50 epochs and 15 epochs of patience using mini-batches of size 1, as suggested by Makarov and Clematide (2018).

Sentence-wise reranker. Table 7 shows the features used in the sentence-wise PRO reranker (Hopkins and May, 2011). We learn the reranker parameters on the development set using perceptron as our binary classification learning algorithm (we also experimented with different losses and a stochastic gradient learner from the `sklearn` library (Pedregosa et al., 2011), but this did not produce any gains).

$p_{\text{WORD-RNN-LM}}(\hat{\mathbf{y}}_{1:m})/m$
$p_{\text{CHAR-RNN-LM}}(\hat{\mathbf{y}}_{1:m})/m$
$p_{\text{WORD-TRIGRAM-LM}}(\hat{\mathbf{y}}_{1:m})/m$
length m
$p(\mathbf{x}_{1:m}, \hat{\mathbf{y}}_{1:m})/m$
$1/m \sum_{i=1}^{m} \mathrm{ED}(\mathbf{x}_i, \hat{\mathbf{y}}_i)$
$1 - \#OOV(\hat{\mathbf{y}}_{1:m})/m$
$1/m \sum_{i=1}^{m} \hat{p}_{\text{TRAIN}}(\mathbf{x}_i, \hat{\mathbf{y}}_i)$
$1/m \sum_{i=1}^{m} \mathbb{1}\{(\mathbf{x}_i, \hat{\mathbf{y}}_i) \in \text{TRAIN}\}$
$1/m \sum_{i=1}^{m} \mathbb{1}\{\text{same-suffix}_k(\mathbf{x}_i, \hat{\mathbf{y}}_i)\}$
$1/m \sum_{i=1}^{m} \mathbb{1}\{\text{same-prefix}_k(\mathbf{x}_i, \hat{\mathbf{y}}_i)\}$

Table 7: Features for sentence reranker. $\mathbf{x}_{1:m}$ is a non-standard sentence and $\hat{\mathbf{y}}_{1:m}$ is a standardized sentence candidate.

Direct model training. We optimize direct models with AdaDelta using mini-batches of size 10. We train for 60 epochs with 15 epochs of patience. We decode them with beam search with beam width eight. We learn a PRO reranker on the development set using hypotheses from the beam. To represent hypotheses, we use features such as the direct model probability of the hypothesis $q_\phi(\hat{\mathbf{y}} \mid \mathbf{x})$, its channel model probability $p_\theta(\mathbf{x} \mid \hat{\mathbf{y}})$, its unigram probability, the relative frequency of the (historical word, hypothesis) pair in the training data, or the edit distance between the hypothesis and the historical input word (Table 8). We rank hypotheses with a combination of normalized edit distance (NED) and accuracy:

$$\Delta(\mathbf{y}, \hat{\mathbf{y}}) = \mathbb{1}\{\mathbf{y} = \hat{\mathbf{y}}\} - \mathrm{NED}(\mathbf{y}, \hat{\mathbf{y}}) \qquad (7)$$

Thus, a hypothesis $\hat{\mathbf{y}}$ attains the highest score of $+1$ if it is identical to the target $\mathbf{y}$ of a development set sample and the lowest score of -1 if the number of edits from $\hat{\mathbf{y}}$ to $\mathbf{y}$ equals the maximum of their lengths.

$p_{\text{UNIGRAM-LM}}(\hat{\mathbf{y}})$	$p_{\text{CHAR-RNN-LM}}(\hat{\mathbf{y}})$
$p_\phi(\mathbf{x} \mid \hat{\mathbf{y}})$	$q_\phi(\hat{\mathbf{y}} \mid \mathbf{x})$
$\mathrm{NED}(\mathbf{x}, \hat{\mathbf{y}})$	$\mathrm{ED}(\mathbf{x}, \hat{\mathbf{y}})$
$\hat{p}_{\text{TRAIN}}(\mathbf{x}, \hat{\mathbf{y}})$	$(\mathbf{x}, \hat{\mathbf{y}}) \in \text{TRAIN}?$
$\text{same-suffix}_k(\mathbf{x}, \hat{\mathbf{y}})?$	$\text{subsequence}(\mathbf{x}, \hat{\mathbf{y}})?$
$\text{same-prefix}_k(\mathbf{x}, \hat{\mathbf{y}})?$	$\text{subsequence}(\hat{\mathbf{y}}, \mathbf{x})?$

Table 8: Features used to rerank hypotheses generated from the direct model. $\mathbf{x}$ is a historical word and $\hat{\mathbf{y}}$ is a modern language hypothesis.