Historical Text Normalization with Delayed Rewards

Flachs, Simon; Bollmann, Marcel; Søgaard, Anders

Published in:
Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics

DOI:
10.18653/v1/P19-1157

Publication date:
2019

Document version
Publisher's PDF, also known as Version of record

Document license:
CC BY

Citation for published version (APA):
Flachs, S., Bollmann, M., & Søgaard, A. (2019). Historical Text Normalization with Delayed Rewards. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 1614-1619). Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1157

Historical Text Normalization with Delayed Rewards

Simon Flachs, Marcel Bollmann, Anders Søgaard
Department of Computer Science
University of Copenhagen
Denmark
{flachs,marcel,soegaard}@di.ku.dk

Abstract

Training neural sequence-to-sequence models with simple token-level log-likelihood is now a standard approach to historical text normalization, albeit often outperformed by phrase-based models. Policy gradient training enables direct optimization for exact matches, and while the small datasets in historical text normalization are prohibitive of from-scratch reinforcement learning, we show that policy gradient *fine-tuning* leads to significant improvements across the board. Policy gradient training, in particular, leads to more accurate normalizations for long or unseen words.

1 Introduction

Historical text normalization is a common approach to making historical documents accessible and searchable. It is a challenging problem, since most historical texts were written without fixed spelling conventions, and spelling is therefore at times idiosyncratic (Piotrowski, 2012).

Traditional approaches to historical text normalization relied on hand-written rules, but recently, several authors have proposed neural models for historical text normalization (Bollmann and Søgaard, 2016; Bollmann, 2018; Tang et al., 2018). Such models are trained using character-level maximum-likelihood training, which is inconsistent with the objective of historical text normalization; namely, transduction into modern, searchable word forms. The discrepancy between character-level loss and our word-level objective means that model decision costs are biased. Our objective, however, is reflected by the standard evaluation metric, which is computed as the fraction of benchmark words that are translated correctly.

In order to mitigate the discrepancy between the optimization method and the task objective, work has been carried out on using reinforcement learning to optimize directly for the evaluation metric (Ranzato et al., 2016; Shen et al., 2016). Reinforcement learning enables direct optimization of exact matches or other non-decomposable metrics, computing updates based on delayed rewards rather than token-level error signals. This paper contrasts maximum likelihood training and training with delayed rewards, in the context of sequence-to-sequence historical text normalization (Bollmann et al., 2017).

Contributions We show that training with delayed rewards achieves better performance than maximum likelihood training across six different historical text normalization benchmarks; and that training with delayed rewards is particularly helpful for long words, words where maximum likelihood training leads to predicting long words, and for unseen words. We note that our approach differs from other applications in the NLP literature in using the mean reward as our baseline, and in comparing different reward functions; we also fine-tune relying only on rewards, rather than a mixture of cross entropy loss and rewards.

2 Historical text normalization datasets

Historical text normalization datasets are rare and typically rather small. Most of them are based on collections of medieval documents. In our experiments, we include six historical text normalization datasets: the English, Hungarian, Icelandic, and Swedish datasets from Pettersson (2016); the German dataset introduced in Bollmann et al. (2017); and the Slovene "Bohorič" dataset from Ljubešić et al. (2016). We use these datasets in the form provided by Bollmann (2019), i.e., preprocessed to remove punctuation, perform Unicode normalization, replace digits that do not require normalization with a dummy symbol, and lowercase all tokens. Table 1 gives an overview of the datasets.

Language		Time Period	Train	IAT
EN	English	1386–1698	148k	75%
DE	German	14th–16th c.	234k	30%
HU	Hungarian	1440–1541	134k	18%
IS	Icelandic	15th c.	50k	47%
SL	Slovene	1750–1840s	50k	41%
SV	Swedish	1527–1812	24k	60%

Table 1: Historical datasets with the time period they represent, the size of their training sets (in tokens), and the approximate percentage of tokens that are invariant across time (IAT), i.e. where the historical and normalized forms are identical.

Note the differences in the number of words that are invariant across time, i.e., where the original input word form is the correct prediction according to the manual annotations. The differences are reasons to expect performance to be higher on English, but lower on Hungarian, for example; since it is easier to learn to memorize the input than to learn abstract transduction patterns. In practice, we see differences being relatively small. Performance on English, however, is significantly higher than for the other languages (see Table 2).

3 Normalization models

Our baseline model is an LSTM-based encoder-decoder model with attention. The model receives as input a sequence of characters from the source vocabulary $(i_1, \dots, i_N)$. Each character i_t is mapped to the corresponding randomly initialized embedding, which is then given as input to the bi-LSTM encoder. The decoder then uses the Bahdanau attention mechanism (Bahdanau et al., 2014) over the encoded representation to output a sequence of characters from the target vocabulary $(o_1, ..., o_M)$. Note that the input and output sequences may differ in length. Both the encoder and decoder is composed of three layers with dimensionality 256. The character embeddings have 128 dimensions.

For training our maximum likelihood baseline, we use the Adam optimiser initialized with a learning rate of 0.001 and default decay rates. In addition, we use a dropout probability of 20%. The model is trained with batch size 16 for 10 epochs with early stopping. All hyper-parameters were tuned on English development data.

Algorithm 1: Reinforcement learning for the neural encoder-decoder model

```
Input  : Parallel Corpus, PC; MLE pretrained parameters, $\theta$
Output: Model parameters $\hat{\theta}$
1  for $(X, Y) \in PC$ do
2      $(\hat{Y}_1, ...\hat{Y}_k), (P(\hat{Y}_1), ...P(\hat{Y}_k)) = sample(X, k, \hat{\theta})$;
3      $Q(\hat{Y}) = normalise(P(\hat{Y}))$;
4      $\bar{r} = 0$ ;              // expected reward
5      for $\hat{Y}_i \in \hat{Y}$ do
6          $\bar{r} += Q(\hat{Y}_i) \cdot A(\hat{Y}_i)$;
7      end
8      $backprop(\hat{\theta}, \bar{r})$ ;    // policy gradient
9  end
10 return $\hat{\theta}$
```

Policy gradient fine-tuning We use policy gradient training with delayed rewards for fine-tuning our models: We use maximum likelihood pretraining for 10 epochs (see above) and update our model based on policy gradients computed using the REINFORCE algorithm (Williams, 1992; Sutton et al., 1999). This enables us to optimize for delayed rewards that are non-decomposable. Specifically, we directly minimize a distance function between strings, e.g., Levenshtein distance, by using negative distance as a delayed reward:[1] $R(\hat{Y}) = -Levenshtein(Y, \hat{Y})$.

REINFORCE maximizes the expected reward, under some probability distribution $P(\hat{Y}|\theta)$, parameterized by some θ. This way, the cost function, $J(\theta)$, is defined as the negative expected reward: $J(\theta) = -E_{\hat{Y} \sim P(\hat{Y}|\theta)}[R(\hat{Y})]$. From this cost function, the PG can be derived as:

$$PG = \nabla_\theta J(\theta) \quad (1)$$

$$= -E_{\hat{Y} \sim P(\hat{Y}|\theta)}[\nabla_\theta \log P(\hat{Y}) \cdot R(\hat{Y})] \quad (2)$$

We refer the reader to prior work for the full derivation (Williams, 1992; Karpathy, 2016). In Equation (2), there is no need to differentiate $R(\hat{Y})$, and policy gradient training therefore is possible with non-differentiable reward functions (Karpathy, 2016). To explore the search space, we use a stochastic sampling function $S(X)$ that, given an input sequence X, produces k sample

[1] In §4, we compare using Levenshtein, Hamming, and Jaro-Winkler distance, with Levenshtein being consistently superior.

hypotheses $\hat{Y}_1, \ldots, \hat{Y}_k$. The hypotheses are generated by, at each time step, sampling actions based on the multinomial probability distribution of the policy. In order to reduce the search space, we sample only from the ten most likely actions at each time step. Furthermore, duplicate samples are filtered.

In practice, we do not optimize directly for the reward $R(\hat{Y})$. Instead we replace it with the advantage score (Weaver and Tao, 2001; Mnih and Gregor, 2014): $A(\hat{Y}) = R(\hat{Y}) - b$, where b is a baseline reward (Weaver and Tao, 2001), introduced to reduce the variance in the gradients. We use the mean reward over the samples as our baseline reward. This way, the advantage scores of the samples will be centered around 0, meaning that about half of the produced samples will be encouraged and about half will be discouraged (Karpathy, 2016).

We also found it necessary to normalize the probability distribution $P(\hat{Y}|X;\theta)$ over the samples from $S(X)$. We follow Shen et al. (2016) and define a probability distribution $Q(\hat{Y}|X;\theta,\alpha)$ over the subspace of $S(X)$.

$$Q(\hat{Y}|X;\theta,\alpha) = \frac{P(\hat{Y}|X;\theta)^{\alpha}}{\sum_{\hat{Y}' \in S(X)} P(\hat{Y}'|X;\theta)^{\alpha}} \quad (3)$$

This function is essentially a smoothing function over the sample probabilities, with a hyper-parameter α that controls the level of smoothing. We follow Shen et al. (2016) and set $\alpha = 0.005$. With these alterations, our cost function and gradients can be defined as:

$$J(\theta) = -E_{\hat{Y} \in S(X)}[A(\hat{Y})] \quad (4)$$

$$PG = -E_{\hat{Y} \in S(X)}[\nabla_{\theta}\, log\ Q(\hat{Y}) \cdot A(\hat{Y})] \quad (5)$$

The algorithm is described in pseudocode in Algorithm 1. We optimized hyper-parameters the same way we optimized our baseline model hyper-parameters. Compared to the baseline, the policy gradient model's optimal batch size is bigger (64), and the learning rate is smaller (0.00001). Both strategies are known to increase generalization, by increasing the scale of random fluctuations in the SGD dynamics (Smith and Le, 2018; Balles et al., 2017).

4 Experiments

Our experiments compare maximum likelihood training and policy gradient training across six historical text normalization datasets (cf. Table 1). We optimized hyper-parameters on the English development data and used the same hyper-parameters across the board (see above).

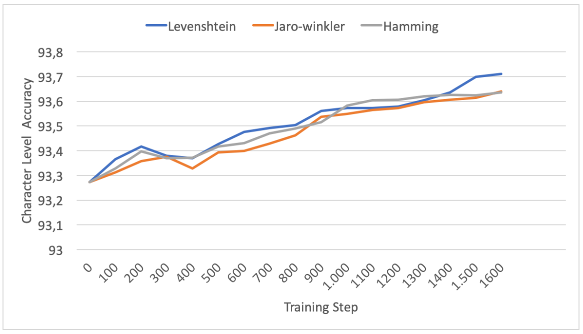

Figure 1: Different reward functions on Icelandic (dev)

	MLE	MLE+PG	Error red.
EN	92.76	**94.18**	20%
DE	87.36	**88.42**	8%
HU	86.68	**88.15**	11%
IS	85.03	**86.05**	7%
SL	91.16	**93.92**	31%
SV	92.99	**93.74**	11%

Table 2: Comparison of maximum likelihood training (MLE) and policy gradient fine-tuning (MLE+PG), given in word-level accuracy in percent, as well as the error reduction between MLE and MLE+PG.

Distance metric We also treated the distance metric used as our reward function as a hyper-parameter. Figure 1 shows a comparison of three reward functions on the Icelandic development data: (i) the Levenshtein distance, which is the number of character operations (substitute, insert, delete) to transform one string into another; (ii) the Hamming distance, which is the number of positions at which the corresponding characters of two strings of equal length are different (we pad the shorter of the two strings with spaces); and (iii) the Jaro-Winkler distance (Cohen et al., 2003), which is a distance metric designed and best suited for short strings such as person names. Levenshtein outperforms Hamming and Jaro-Winkler distance on the English development data, as well as on the Icelandic development data. We therefore use the Levenshtein distance as the reward function in our experiments.

	GOLD LENGTH	MLE LENGTH	MLE BACKOFF	IDENTICAL	UNSEEN WORDS
EN	**0.09	**0.14	**-0.20	**-0.07	**0.10
DE	**0.11	**0.10	**-0.08	-0.05	**0.12
HU	**0.09	**0.11	**-0.07	**-0.03	**0.10
IS	**0.04	**0.05	0.02	**0.08	**0.05

Table 3: Correlations (Pearson's r) between improvements with reinforcement learning and datapoint characteristics; ** denotes significance with $p < 0.001$.

Results The results are presented in Table 2.[2] Generally, we see that policy gradient fine-tuning improves results across the board. For English, the error reduction is 20%. For German, Hungarian, Icelandic, Slovene, and Swedish, the error reduction is smaller (7–16%), but still considerable and highly significant ($p < 0.01$). Tang et al. (2018) do show, however, that multi-headed attention architectures (Vaswani et al., 2017) generally seem to outperform sequence-to-sequence models with attention for historical text normalization. This is orthogonal to the analysis presented here, and similar improvements can likely be obtained by multi-headed attention architectures.

Analysis To avoid bias from small, high variance datasets, we limit error analysis to English, German, Hungarian, and Icelandic. In Table 3, we present correlation scores between our observed improvements and characteristics of the data.[3] We consider the following characteristics:

1. GOLD LENGTH: Reinforcement learning with delayed rewards can potentially mitigate error propagation, and we do observe that gains from reinforcement learning, i.e., the distribution of correct normalizations by reinforcement learning that our baseline architecture classified wrongly, correlate significantly with the length of the input across all four languages.

2. MLE LENGTH: The correlations are even stronger with the length of the output of the MLE model. This suggests that reinforcement learning – or policy gradient training – is particularly effective on examples for which maximum likelihood training tends to predict long normalizations.

3. MLE BACKOFF: We also correlate gains with the distribution of instances on which the MLE backed off to predicting the original input word form. Here, we see a negative correlation, suggesting our baseline is good at predicting when the word form is invariant across time.

4. IDENTICAL: The three trends above are all quite strong. Our fourth variable is when input and output are identical (invariant across time). Here, we see mixed results. Policy gradient gains correlate *negatively* with invariance in English, but positively in Icelandic.

5. UNSEEN WORDS: Finally, we correlate gains with whether words had been previously seen at training time. Our policy gradient fine-tuned model performs much better on unseen words, and especially for English, German, and Hungarian, we see strong correlations between improvements and unseen words. Our predictions also exhibit smaller Levenshtein distances to the annotations compared to our baseline model, e.g., 0.11 vs. 0.14 for English, respectively, and 0.20 vs. 0.23 for German.

5 Conclusions

Our experiments show that across several languages, policy gradient fine-tuning outperforms maximum likelihood training of sequence-to-sequence models for historical text normalization. Since historical text normalization is a character-level transduction task, it is feasible to experiment with reinforcement learning, and we believe

[2]Note that for the MLE baseline, we performed our own hyperparameter tuning, which results in a different configuration than used in previous work (e.g., Bollmann et al., 2017; Tang et al., 2018). We observe that our baseline is weaker than the models reported in Bollmann (2019), but even so, the MLE+PG approach yields state-of-the-art performance on the Slovene dataset.

[3]Correlations are Pearson's r. Samples are big enough to motivate a parametric test, but we obtain similar coefficients and significance levels with Spearman's ρ.

our results are very promising. In our error analysis, we, in addition, observe that reinforcement learning is particularly beneficial for long words and unseen words, which are probably the hardest challenges in historical text normalization.

Acknowledgments

Simon Flachs was supported by a PhD grant from Innovation Fund Denmark. Anders Søgaard was supported by a Google Focused Research Award. Marcel Bollmann was partly funded from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 845995. We gratefully acknowledge the donation of a Titan Xp GPU by the NVIDIA Corporation.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Lukas Balles, Javier Romero, and Philipp Hennig. 2017. Coupling adaptive batch sizes with learning rates. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence (UAI)*.

Marcel Bollmann. 2018. Normalization of historical texts with neural network models. *Bochumer Linguistische Arbeitsberichte*, 22.

Marcel Bollmann. 2019. A large-scale comparison of historical text normalization systems. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3885–3898. Association for Computational Linguistics.

Marcel Bollmann, Joachim Bingel, and Anders Søgaard. 2017. Learning attention for historical text normalization by learning to pronounce. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 332–344, Vancouver, Canada. Association for Computational Linguistics.

Marcel Bollmann and Anders Søgaard. 2016. Improving historical spelling normalization with bidirectional LSTMs and multi-task learning. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 131–139, Osaka, Japan. The COLING 2016 Organizing Committee.

William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. 2003. A comparison of string metrics for matching names and records. In *KDD Workshop on Data Cleaning and Object Consolidation*.

Andrej Karpathy. 2016. Deep reinforcement learning: Pong from pixels. Retrieved on 21-10-2017.

Nikola Ljubešić, Katja Zupan, Darja Fišer, and Tomaž Erjavec. 2016. Normalising Slovene data: historical texts vs. user-generated content. In *Proceedings of the 13th Conference on Natural Language Processing (KONVENS 2016)*, volume 16 of *Bochumer Linguistische Arbeitsberichte*, pages 146–155, Bochum, Germany.

Andriy Mnih and Karol Gregor. 2014. Neural variational inference and learning in belief networks. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1791–1799, Bejing, China. PMLR.

Eva Pettersson. 2016. *Spelling Normalisation and Linguistic Analysis of Historical Text for Information Extraction*. Doctoral dissertation, Uppsala University, Department of Linguistics and Philology, Uppsala.

Michael Piotrowski. 2012. *Natural Language Processing for Historical Texts*. Number 17 in Synthesis Lectures on Human Language Technologies. Morgan & Claypool.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2–4, 2016, Conference Track Proceedings*.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1683–1692, Berlin, Germany. Association for Computational Linguistics.

Samuel L. Smith and Quoc V. Le. 2018. A Bayesian perspective on generalization and stochastic gradient descent. In *International Conference on Learning Representations*.

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, pages 1057–1063, Cambridge, MA, USA. MIT Press.

Gongbo Tang, Fabienne Cap, Eva Pettersson, and Joakim Nivre. 2018. An evaluation of neural machine translation models on historical spelling normalization. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1320–1331, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Lex Weaver and Nigel Tao. 2001. The optimal reward baseline for gradient-based reinforcement learning. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI)*.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256.