A Log-Linear Model for Unsupervised Text Normalization

Yi Yang
School of Interactive Computing
Georgia Institute of Technology
`yiyang@gatech.edu`

Jacob Eisenstein
School of Interactive Computing
Georgia Institute of Technology
`jacobe@gatech.edu`

Abstract

We present a unified unsupervised statistical model for text normalization. The relationship between standard and non-standard tokens is characterized by a log-linear model, permitting arbitrary features. The weights of these features are trained in a maximum-likelihood framework, employing a novel sequential Monte Carlo training algorithm to overcome the large label space, which would be impractical for traditional dynamic programming solutions. This model is implemented in a normalization system called UNLOL, which achieves the best known results on two normalization datasets, outperforming more complex systems. We use the output of UNLOL to automatically normalize a large corpus of social media text, revealing a set of coherent orthographic styles that underlie online language variation.

1 Introduction

Social media language can differ substantially from other written text. Many of the attempts to characterize and overcome this variation have focused on *normalization*: transforming social media language into text that better matches standard datasets (Sproat et al., 2001; Liu et al., 2011). Because there is little available training data, and because social media language changes rapidly (Eisenstein, 2013b), fully supervised training is generally not considered appropriate for this task. However, due to the extremely high-dimensional output space — arbitrary sequences of words across the vocabulary — it is a very challenging problem for unsupervised learning. Perhaps it is for these reasons that the most successful systems are pipeline architectures that cobble together a diverse array of techniques and resources, including statistical language models, dependency parsers, string edit distances, off-the-shelf spellcheckers, and curated slang dictionaries (Liu et al., 2011; Han and Baldwin, 2011; Han et al., 2013).

We propose a different approach, performing normalization in a maximum-likelihood framework. There are two main sources of information to be exploited: local context, and surface similarity between the observed strings and normalization candidates. We treat the local context using standard language modeling techniques; we treat string similarity with a log-linear model that includes features for both surface similarity and word-word pairs.

Because labeled examples of normalized text are not available, this model cannot be trained in the standard supervised fashion. Nor can we apply dynamic programming techniques for unsupervised training of locally-normalized conditional models (Berg-Kirkpatrick et al., 2010), as their complexity is quadratic in the size of label space; in normalization, the label space is the vocabulary itself, with at least 10^4 elements. Instead, we present a new training approach using Monte Carlo techniques to compute an approximate gradient on the feature weights. This training method may be applicable in other unsupervised learning problems with a large label space.

This model is implemented in a normalization system called UNLOL (**u**nsupervised **n**ormalization in a **LO**g-**L**inear model). It is a lightweight probabilistic approach, relying only on a language model for the target domain; it can be adapted to new corpora text or new domains easily and quickly. Our evaluations show that UNLOL outperforms the state-of-the-art on standard normalization datasets. In addition, we demonstrate the linguistic insights that can be obtained from normalization, using UNLOL to identify classes of orthographic transformations that form coherent linguistic styles.

2 Background

The text normalization task was introduced by Sproat et al. (2001), and attained popularity in the context of SMS messages (Choudhury et al., 2007b). It has become still more salient in the era of widespread social media, particularly Twitter. Han and Baldwin (2011) formally define a normalization task for Twitter, focusing on normalizations between single tokens, and excluding multi-word tokens like `lol` (*laugh out loud*). The normalization task has been criticized by Eisenstein (2013b), who argues that it strips away important social meanings. In recent work, normalization has been shown to yield improvements for part-of-speech tagging (Han et al., 2013), parsing (Zhang et al., 2013), and machine translation (Hassan and Menezes, 2013). As we will show in Section 7, accurate automated normalization can also improve our understanding of the nature of social media language.

Supervised methods Early work on normalization focused on labeled SMS datasets, using approaches such as noisy-channel modeling (Choudhury et al., 2007a) and machine translation (Aw et al., 2006), as well as hybrid combinations of spelling correction and speech recognition (Kobus et al., 2008; Beaufort et al., 2010). This work sought to balance language models (favoring words that fit in context) with transformation models (favoring words that are similar to the observed text). Our approach can also be seen as a noisy channel model, but unlike this prior work, no labeled data is required.

Unsupervised methods Cook and Stevenson (2009) manually identify several word formation types within a noisy channel framework. They parametrize each formation type with a small number of scalar values, so that all legal transformations of a given type are equally likely. The scalar parameters are then estimated using expectation maximization. This work stands apart from most of the other unsupervised models, which are pipelines.

Contractor et al. (2010) use string edit distance to identify closely-related candidate orthographic forms and then decode the message using a language model. Gouws et al. (2011) refine this approach by mining an "exception dictionary" of strongly-associated word pairs such as *you*/`u`. Like Contractor et al. (2010), we apply string edit distance, and like Gouws et al. (2011), we capture strongly related word pairs. However, rather than applying these properties as filtering steps in a pipeline, we add them as features in a unified log-linear model.

Recent approaches have sought to improve accuracy by bringing more external resources and complex architectures to bear. Han and Baldwin (2011) begin with a set of string similarity metrics, and then apply dependency parsing to identify contextually-similar words. Liu et al. (2011) extract noisy training pairs from the search snippets that result from carefully designed queries to Google, and then train a conditional random field (Lafferty et al., 2001) to estimate a character-based translation model. They later extend this work by adding a model of visual priming, an off-the-shelf spell-checker, and local context (Liu et al., 2012a). Hassan and Menezes (2013) use a random walk framework to capture contextual similarity, which they then interpolate with an edit distance metric. Rather than seeking additional external resources or designing more complex metrics of context and similarity, we propose a unified statistical model, which learns feature weights in a maximum-likelihood framework.

3 Approach

Our approach is motivated by the following criteria:

- **Unsupervised**. We want to be able to train a model without labeled data. At present, labeled data for Twitter normalization is available only in small quantities. Moreover, as social media language is undergoing rapid change (Eisenstein, 2013b), labeled datasets may become stale and increasingly ill-suited to new spellings and words.

- **Low-resource**. Other unsupervised approaches take advantage of resources such as slang dictionaries and spell checkers (Han and Baldwin, 2011; Liu et al., 2011). Resources that characterize the current state of internet language risk becoming outdated; in this paper we investigate whether high-quality normalization is possible without any such resources.

- **Featurized**. The relationship between any pair of words can be characterized in a number of different ways, ranging from simple character-level rules (e.g., *going*/`goin`) to larger substitutions (e.g., *someone*/`sum1`), and even to patterns that are lexically restricted (e.g., *you*/`u`, *to*/`2`). For these reasons, we seek a model that permits many overlapping features to describe candidate word pairs. These features may include simple string edit distance metrics, as well as lexical features that memorize specific pairs of standard and nonstandard words.

- **Context-driven**. Learning potentially arbitrary word-to-word transformations without supervision would be impossible without the strong additional cue of local context. For example, in the phrase

  `give me suttin to believe in`,

  even a reader who has never before seen the word `suttin` may recognize it as a phonetic transcription of *something*. The relatively high string edit distance is overcome by the strong contextual preference for the word *something* over orthographically closer alternatives such as *button* or *suiting*. We can apply an arbitrary target language model, leveraging large amounts of unlabeled data and catering to the desired linguistic characteristics of the normalized content.

- **Holistic**. While several prior approaches — such as normalization dictionaries — operate at the token level, our approach reasons over the scope of the entire message. The necessity for such holistic, joint inference and learning can be seen by changing the example above to:

  `gimme suttin 2 beleive innnn`.

  None of these tokens are standard (except `2`, which appears in a nonstandard sense here), so without joint inference, it would not be possible to use context to help normalize `suttin`. Only by jointly reasoning over the entire message can we obtain the correct normalization.

These desiderata point towards a featurized sequence model, which must be trained without labeled examples. While there is prior work on training sequence models without supervision (Smith and Eisner, 2005; Berg-Kirkpatrick et al., 2010), there is an additional complication not faced by models for tasks such as part-of-speech tagging and named entity recognition: the potential label space of standard words is large, on the order of at least 10^4. Naive application of Viterbi decoding — which is a component of training for both Contrastive Estimation (Smith and Eisner, 2005) and the locally-normalized sequence labeling model of Berg-Kirkpatrick et al. (2010) — will be stymied by Viterbi's quadratic complexity in the dimension of the label space. While various pruning heuristics may be applied, we instead look to Sequential Monte Carlo (SMC), a randomized algorithm which approximates the necessary feature expectations through weighted samples.

4 Model

Given a set of source-language sentences $S = \{\mathbf{s}_1, \mathbf{s}_2, \ldots\}$ (e.g., Tweets), our goal is to transduce them into target-language sentences $T = \{\mathbf{t}_1, \mathbf{t}_2, \ldots\}$ (standard English). We are given a target language model $P(\mathbf{t})$, which can be estimated from some large set of unlabeled target-language sentences. We denote the vocabularies of source language and target language as ν_S and ν_T respectively.

We define a log-linear model that scores source and target strings, with the form

$$P(\mathbf{s}|\mathbf{t};\theta) \propto \exp\left(\theta^{\mathsf{T}}\mathbf{f}(\mathbf{s},\mathbf{t})\right). \tag{1}$$

The desired conditional probability $P(\mathbf{t}|\mathbf{s})$ can be obtained by combining this model with the target language model, $P(\mathbf{t}|\mathbf{s}) \propto P(\mathbf{s}|\mathbf{t};\theta)P(\mathbf{t})$. Since no labeled data is available, the parameters θ must be estimated by maximizing the log-likelihood of the source-language data. We define the log-likelihood

$\ell_\theta(\mathbf{s})$ for a source-language sentence $\mathbf{s}$ as follows:

$$\ell_\theta(\mathbf{s}) = \log P(\mathbf{s}) = \log \sum_{\mathbf{t}} P(\mathbf{s}|\mathbf{t};\theta)P(\mathbf{t})$$

We would like to maximize this objective by making gradient-based updates.

$$\begin{aligned} \frac{\partial \ell_\theta(\mathbf{s})}{\partial\theta} &= \frac{1}{P(\mathbf{s})}\sum_{\mathbf{t}} P(\mathbf{t})\frac{\partial}{\partial\theta}P(\mathbf{s}|\mathbf{t};\theta) \\ &= \sum_{\mathbf{t}} P(\mathbf{t}|\mathbf{s})\left(\mathbf{f}(\mathbf{s},\mathbf{t}) - \sum_{\mathbf{s}'} P(\mathbf{s}'|\mathbf{t})\mathbf{f}(\mathbf{s}',\mathbf{t})\right) \\ &= E_{\mathbf{t}|\mathbf{s}}[\mathbf{f}(\mathbf{s},\mathbf{t}) - E_{\mathbf{s}'|\mathbf{t}}[\mathbf{f}(\mathbf{s}',\mathbf{t})]] \end{aligned} \tag{2}$$

We are left with a difference in expected feature counts, as is typical in log-linear models. However, unlike the supervised case, here *both* terms are expectations: the outer expectation is over all target sequences (given the observed source sequence), and the nested expectation is over all source sequences, given the target sequence. As the space of possible target sequences $\mathbf{t}$ grows exponentially in the length of the source sequence, it will not be practical to compute this expectation directly.

Dynamic programming is the typical solution for computing feature expectations, and can be applied to sequence models when the feature function decomposes locally. There are two reasons this will not work in our case. First, while the forward-backward algorithm would enable us to compute $E_{\mathbf{t}|\mathbf{s}}$, it would not give us the nested expectation $E_{\mathbf{t}|\mathbf{s}}[E_{\mathbf{s}'|\mathbf{t}}]$; this is the classic challenge in training globally-normalized log-linear models without labeled data (Smith and Eisner, 2005). Second, both forward-backward and the Viterbi algorithm have time complexity that is quadratic in the dimension of the label space, at least 10^4 or 10^5. As we will show, Sequential Monte Carlo (SMC) algorithms have a number of advantages in this setting: they permit the efficient computation of both the outer and inner expectations, they are trivially parallelizable, and the number of samples provides an intuitive tuning tradeoff between accuracy and speed.

4.1 Sequential Monte Carlo approximation

Sequential Monte Carlo algorithms are a class of sampling-based algorithms in which latent variables are sampled sequentially (Cappe et al., 2007). They are particularly well-suited to sequence models, though they can be applied more broadly. SMC algorithms maintain a set of weighted hypotheses; the weights correspond to probabilities, and in our case, the hypotheses correspond to target language word sequences. Specifically, we approximate the conditional probability,

$$P(\mathbf{t}_{1:n}|\mathbf{s}_{1:n}) \approx \sum_{k=1}^{K} \omega_n^k \delta_{\mathbf{t}_{1:n}^k}(\mathbf{t}_{1:n}),$$

where ω_n^k is the normalized weight of sample k at word n ($\tilde{\omega}_n^k$ is the unnormalized weight), and $\delta_{\mathbf{t}_{1:n}^k}$ is a delta function centered at $\mathbf{t}_{1:n}^k$.

At each step, and for each hypothesis k, a new target word is sampled from a *proposal distribution*, and the weight of the hypothesis is then updated. We maintain feature counts for each hypothesis, and approximate the expectation by taking a weighted average using the hypothesis weights. The proposal distribution will be described in detail later.

We make a Markov assumption, so that the emission probability $P(\mathbf{s}|\mathbf{t})$ decomposes across the elements of the sentence $P(\mathbf{s}|\mathbf{t}) = \prod_n^N P(s_n|t_n)$. This means that the feature functions $\mathbf{f}(\mathbf{s},\mathbf{t})$ must decompose on each $\langle s_n, t_n\rangle$ pair. We can then rewrite (1) as

$$P(\mathbf{s}|\mathbf{t};\theta) = \prod_n^N \frac{\exp\left(\theta^\mathsf{T}\mathbf{f}(s_n,t_n)\right)}{Z(t_n)} \tag{3}$$

$$Z(t_n) = \sum_s \exp\left(\theta^\mathsf{T}\mathbf{f}(s,t_n)\right). \tag{4}$$

In addition, we assume that the target language model $P(\mathbf{t})$ can be written as an N-gram language model, $P(\mathbf{t}) = \prod_n P(t_n|t_{n-1},\ldots t_{n-k+1})$. With these assumptions, we can view normalization as a finite state-space model in which the target language model defines the prior distribution of the process and Equation 3 defines the likelihood function. We are able to compute the the posterior probability $P(\mathbf{t}|\mathbf{s})$ using *sequential importance sampling*, a member of the SMC family.

The crucial idea in sequential importance sampling is to update the hypotheses $\mathbf{t}_{1:n}^k$ and their weights ω_n^k so that they approximate the posterior distribution at the next time step, $P(\mathbf{t}_{1:n+1}|\mathbf{s}_{1:n+1})$.

Assuming the proposal distribution has the form $Q(\mathbf{t}^k_{1:n}|\mathbf{s}_{1:n})$, the importance weights are given by

$$\omega^k_n \propto \frac{P(\mathbf{t}^k_{1:n}|\mathbf{s}_{1:n})}{Q(\mathbf{t}^k_{1:n}|\mathbf{s}_{1:n})} \tag{5}$$

In order to update the hypotheses recursively, we rewrite $P(\mathbf{t}_{1:n}|\mathbf{s}_{1:n})$ as:

$$\begin{aligned}P(\mathbf{t}_{1:n}|\mathbf{s}_{1:n}) &= \frac{P(s_n|\mathbf{t}_{1:n},\mathbf{s}_{1:n-1})P(\mathbf{t}_{1:n}|\mathbf{s}_{1:n-1})}{P(s_n|\mathbf{s}_{1:n-1})}\\ &= \frac{P(s_n|t_n)P(t_n|\mathbf{t}_{1:n-1},\mathbf{s}_{1:n-1})P(\mathbf{t}_{1:n-1}|\mathbf{s}_{1:n-1})}{P(s_n|\mathbf{s}_{1:n-1})}\\ &\propto P(s_n|t_n)P(t_n|t_{n-1})P(\mathbf{t}_{1:n-1}|\mathbf{s}_{1:n-1}),\end{aligned}$$

assuming a bigram language model. We further assume the proposal distribution Q can be factored as:

$$\begin{aligned}Q(\mathbf{t}_{1:n}|\mathbf{s}_{1:n}) &= Q(t_n|\mathbf{t}_{1:n-1},\mathbf{s}_{1:n})Q(\mathbf{t}_{1:n-1}|\mathbf{s}_{1:n-1})\\ &= Q(t_n|t_{n-1},s_n)Q(\mathbf{t}_{1:n-1}|\mathbf{s}_{1:n-1}).\end{aligned} \tag{6}$$

Then the unnormalized importance weights simplify to a recurrence:

$$\tilde{\omega}^k_n = \frac{P(s_n|t^k_n)P(t^k_n|t^k_{n-1})P(\mathbf{t}^k_{1:n-1}|\mathbf{s}_{1:n-1})}{Q(t_n|t_{n-1},s_n)Q(\mathbf{t}^k_{1:n-1}|\mathbf{s}_{1:n-1})} \tag{7}$$

$$= \omega^k_{n-1}\frac{P(s_n|t^k_n)P(t^k_n|t^k_{n-1})}{Q(t_n|t_{n-1},s_n)} \tag{8}$$

Therefore, we can approximate the posterior distribution $P(t_n|\mathbf{s}_{1:n}) \approx \sum_{k=1}^K \omega^k_n \delta_{t^k_n}(t_n)$, and compute the outer expectation as follows:

$$E_{\mathbf{t}|\mathbf{s}}[\mathbf{f}(\mathbf{s},\mathbf{t})] = \sum_{k=1}^{K}\omega^k_N \sum_{n=1}^{N}\mathbf{f}(s_n,t^k_n) \tag{9}$$

We compute the nested expectation using a non-sequential Monte Carlo approximation, assuming we can draw $s^{\ell,k} \sim P(s|t^k_n)$.

$$E_{\mathbf{s}|\mathbf{t}^k}[\mathbf{f}(\mathbf{s},\mathbf{t}^k)] = \frac{1}{L}\sum_{n=1}^{N}\sum_{\ell=1}^{L}\mathbf{f}(s^{\ell,k}_n,t^k_n)$$

This gives the overall gradient computation:

$$\begin{aligned}E_{\mathbf{t}|\mathbf{s}}[\mathbf{f}(\mathbf{s},\mathbf{t}) - E_{\mathbf{s}'|\mathbf{t}}[\mathbf{f}(\mathbf{s}',\mathbf{t})]] &= \frac{1}{\sum_{k=1}^K \tilde{\omega}^k_N}\sum_{k=1}^{K}\tilde{\omega}^k_N\\ &\times \sum_{n=1}^{N}\left(\mathbf{f}(s_n,t^k_n) - \frac{1}{L}\sum_{\ell=1}^{L}\mathbf{f}(s^{\ell,k}_n,t^k_n)\right)\end{aligned} \tag{10}$$

where we sample t^k_n and update ω^k_n while moving from left-to-right, and sample $s^{\ell,k}_n$ at each n. Note that although the sequential importance sampler moves left-to-right like a filter, we use only the final weights ω_N to compute the expectation. Thus, the resulting expectation is based on the distribution $P(\mathbf{s}_{1:N}|\mathbf{t}_{1:N})$, so that no backwards "smoothing" pass (Godsill et al., 2004) is needed to eliminate bias. Other applications of sequential Monte Carlo make use of resampling (Cappe et al., 2007) to avoid degeneration of the hypothesis weights, but we found this to be unnecessary due to the short length of Twitter messages.

4.2 Proposal distribution

The major computational challenge for dynamic programming approaches to normalization is the large label space, equal to the size of the target vocabulary. It may appear that all we have gained by applying sequential Monte Carlo is to convert a computational problem into a statistical one: a naive sampling approach will have little hope of finding the small high-probability region of the high-dimensional label space. However, sequential importance sampling allows us to address this issue through the proposal distribution, from which we sample the candidate words t_n. Careful design of the proposal distribution can guide sampling towards the high-probability space. In the asymptotic limit of an infinite number of samples, any non-pathological proposal distribution will ultimately arrive at the desired estimate, but a good proposal distribution can greatly reduce the number of samples needed.

Doucet et al. (2001) note that the optimal proposal — which minimizes the variance of the importance weights conditional on $\mathbf{t}_{1:n-1}$ and $\mathbf{s}_{1:n}$ — has the following form:

$$Q(t^k_n|s_n,t^k_{n-1}) = \frac{P(s_n|t^k_n)P(t^k_n|t^k_{n-1})}{\sum_{t'}P(s_n|t')P(t'|t^k_{n-1})} \tag{11}$$

Sampling from this proposal requires computing the normalized distribution $P(s_n|t_n^k)$; similarly, the update of the hypothesis weights (Equation 8) requires the calculation of Q in its normalized form. In each case, the total cost is the product of the vocabulary sizes, $\mathcal{O}(\#|\nu_T|\#|\nu_S|)$, which is not tractable as the vocabularies become large.

In low-dimensional settings, a convenient solution is to set the proposal distribution equal to the transition distribution, $Q(t_n^k|s_n, t_{n-1}^k) = P(t_n^k|t_{n-1}^k, \ldots, t_{n-k+1}^k)$. This choice is called the "bootstrap filter," and it has the advantage that the weights $\omega^{(k)}$ are exactly identical to the product of emission likelihoods $\prod_n P(s_n|t_n^k)$. The complexity of computing the hypothesis weights is thus $\mathcal{O}(\#|\nu_S|)$. However, because this proposal ignores the emission likelihood, the bootstrap filter has very little hope of finding a high-probability sample in high-entropy contexts.

We strike a middle ground between efficiency and accuracy, using a proposal distribution that is closely related to the overall likelihood, yet is tractable to sample and compute:

$$\begin{aligned} Q(t_n^k|s_n, t_{n-1}^k) &\overset{def}{=} \\ &\frac{P(s_n|t_n^k)Z(t_n^k)P(t_n^k|t_{n-1}^k)}{\sum_{t'} P(s_n|t')Z(t')P(t'|t_{n-1}^k)} \\ =&\frac{\exp\left(\theta^{\mathsf{T}}\mathbf{f}(s_n, t_n)\right) P(t_n^k|t_{n-1}^k)}{\sum_{t'} \exp\left(\theta^{\mathsf{T}}\mathbf{f}(s_n, t')\right) P(t'|t_{n-1}^k)} \end{aligned} \quad (12)$$

Here, we simply replace the likelihood distribution in (11) by its unnormalized version.

To update the unnormalized hypothesis weights $\tilde{\omega}_n^k$, we have

$$\tilde{\omega}_n^k = \omega_{n-1}^k \frac{\sum_{t'} \exp\left(\theta^{\mathsf{T}}\mathbf{f}(s_n, t')\right) P(t'|t_{n-1}^k)}{Z(t_n^k)} \quad (13)$$

The numerator requires summing over all elements in ν_T and the denominator $Z(t_n^k)$ requires summing over all elements in ν_S, for a total cost of $\mathcal{O}(\#|\nu_T| + \#|\nu_S|)$.

4.3 Decoding

Given an input source sentence $\mathbf{s}$, the decoding problem is to find a target sentence $\mathbf{t}$ that maximizes $P(\mathbf{t}|\mathbf{s}) \propto P(\mathbf{s}|\mathbf{t})P(\mathbf{t}) = \prod_n^N P(s_n|t_n)P(t_n|t_{n-1})$.

Feature name	Description
word-word pair	A set of binary features for each source/target word pair $\langle s, t\rangle$
string similarity	A set of binary features indicating whether s is one of the top N string similar non-standard words of t, for $N \in \{5, 10, 25, 50, 100, 250, 500, 1000\}$

Table 1: The feature set for our log-linear model

As with learning, we cannot apply the usual dynamic programming algorithm (Viterbi), because of its quadratic cost in the size of the target language vocabulary. This must be multiplied by the cost of computing the normalized probability $P(s_n|t_n)$, resulting in a prohibitive time complexity of $\mathcal{O}(\#|\nu_S|\#|\nu_T|^2 N)$.

We consider two approximate decoding algorithms. The first is to simply apply the proposal distribution, with linear complexity in the size of the two vocabularies. However, this decoder is not identical to $P(\mathbf{t}|\mathbf{s})$, because of the extra factor of $Z(t)$ in the numerator. Alternatively, we can apply the proposal distribution for selecting target word candidates, then apply the Viterbi algorithm only within these candidates. The total cost is $\mathcal{O}(\#|\nu_S|T^2 N)$, where T is the number of target word candidates we consider; this will asymptotically approach $P(\mathbf{t}|\mathbf{s})$ as $T \to \#|\nu_T|$. Our evaluations use the more expensive proposal+Viterbi decoding, but accuracy with the more efficient proposal-based decoding is very similar.

4.4 Features

Our system uses the feature types described in Table 1. The word pair features are designed to capture lexical conventions, e.g. *you*/u. We only consider word pair features that fired during training. The string similarity features rely on the similarity function proposed by Contractor et al. (2010), which has proven effective for normalization in prior work. We bin this similarity to create binary features indicating whether a string s is in the top-N most similar strings to t; this binning yields substantial speed improvements without negatively impacting accuracy.

5 Implementation and data

The model and inference described in the previous section are implemented in a software system for normalizing text on twitter, called UNLOL: **u**nsupervised **n**ormalization in a **LO**g-**L**inear model. The final system can process roughly 10,000 Tweets per hour. We now describe some implementation details.

5.1 Normalization candidates

Most tokens in tweets do not require normalization. The question of how to identify which words are to be normalized is still an open problem. Following Han and Baldwin (2011), we build a dictionary of words which are permissible in the target domain, and make no attempt to normalize source strings that match these words. As with other comparable approaches, we are therefore unable to normalize strings like `ill` into *I'll*. Our set of "in-vocabulary" (IV) words is based on the GNU aspell dictionary (v0.60.6), containing 97,070 words. From this dictionary, we follow Liu et al. (2012a) and remove all the words with a count of less than 20 in the Edinburgh Twitter corpus (Petrović et al., 2010) — resulting in a total of 52,449 target words. All single characters except `a` and `i` are excluded, and `rt` is treated as in-vocabulary. For all in-vocabulary words, we define $P(s_n|t_n) = \delta(s_n, t_n)$, taking the value of zero when $s_n \neq t_n$. This effectively prevents our model from attempting to normalize these words.

In addition to words that are in the target vocabulary, there are many other strings that should not be normalized, such as names and multiword shortenings (e.g. *going to*/`gonna`).[1] We follow prior work and assume that the set of normalization candidates is known in advance during test set decoding (Han et al., 2013). However, the unlabeled training data has no such information. Thus, during training we attempt to normalize all tokens that (1) are not in our lexicon of IV words, and (2) are composed of letters, numbers and the apostrophe. This set includes contractions like "gonna" and "gotta", which would not appear in the test set, but are nonetheless normalized during training. For each OOV token, we conduct a pre-normalization step by reducing any repetitions of more than two letters in the nonstandard words to exactly two letters (e.g., `cooool` $\rightarrow$ `cool`).

[1]Whether multiword shortenings should be normalized is arguable, but they are outside the scope of current normalization datasets (Han and Baldwin, 2011).

5.2 Language modeling

The Kneser-Ney smoothed trigram target language model is estimated with the SRILM toolkit Stolcke (2002), using Tweets from the Edinburgh Twitter corpus that contain no OOV words besides hashtags and username mentions (following (Han et al., 2013)). We use this language model for both training and decoding. We occasionally find training contexts in which the trigram $\langle t_n, t_{n-1}, t_{n-2} \rangle$ is unobserved in the language model data; features resulting from such trigrams are not considered when computing the weight gradients.

5.3 Parameters

The Monte Carlo approximations require two parameters: the number of samples for sequential Monte Carlo (K), and the number of samples for the non-sequential sampler of the nested expectation (L, from Equation 10). The theory of Monte Carlo approximation states that the quality of the approximation should only improve as the number of samples increases; we obtained good results with $K = 10$ and $L = 1$, and found relatively little improvement by increasing these values. The number of hypotheses considered by the decoder is set to $T = 10$; again, the performance should only improve with T, as we more closely approximate full Viterbi decoding.

6 Experiments

Datasets We use two existing labeled Twitter datasets to evaluate our approach. The first dataset — which we call LWWL11, based on the names of its authors Liu et al. (2011) — contains 3,802 individual "nonstandard" words (i.e., words that are not in the target vocabulary) and their normalized forms. The rest of the message in which the words is appear is not available. As this corpus does not provide linguistic context, its decoding must use a unigram target language model. The second dataset — which is called LexNorm1.1 by its authors Han and Baldwin (2011) — contains 549 complete tweets with 1,184 nonstandard tokens (558 unique word types).

Method	Dataset	Precision	Recall	F-measure
(Liu et al. 2011)	LMML11	68.88	68.88	68.88
(Liu et al. 2012)		69.81	69.81	69.81
UNLOL		**73.04**	**73.04**	**73.04**
(Han and Baldwin, 2011)	LexNorm 1.1	75.30	75.30	75.30
(Liu et al. 2012)		84.13	78.38	81.15
(Hassan et al. 2013)		**85.37**	56.4	69.93
UNLOL		82.09	**82.09**	**82.09**
UNLOL	LexNorm 1.2	82.06	82.06	82.06

Table 2: Empirical results

In this corpus, we can decode with a trigram language model.

Close analysis of LexNorm1.1 revealed some inconsistencies in annotation (for example, `y'all` and `2` are sometimes normalized to *you* and *to*, but are left unnormalized in other cases). In addition, several annotations disagree with existing resources on internet language and dialectal English. For example, `smh` is normalized to *somehow* in LexNorm1.1, but `internetslang.com` and `urbandictionary.com` assert that it stands for *shake my head*, and this is evident from examples such as `smh at this girl`. Similarly, `finna` is normalized to *finally* in LexNorm1.1, but from the literature on African American English (Green, 2002), it corresponds to *fixing to* (e.g., `i'm finna go home`). To address these issues, we have produced a new version of this dataset, which we call LexNorm1.2 (after consulting with the creators of LexNorm1.1). LexNorm1.2 differs from version 1.1 in the annotations for 172 of the 2140 OOV words. We evaluate on LexNorm1.1 to compare with prior work, but we also present results on LexNorm1.2 in the hope that it will become standard in future work on normalization in English. The dataset is available at `http://www.cc.gatech.edu/~jeisenst/lexnorm.v1.2.tgz`.

To obtain unlabeled training data, we randomly sample 50 tweets from the Edinburgh Twitter corpus Petrović et al. (2010) for each OOV word. Some OOV words appear less than 50 times in the corpus, so we obtained more training tweets for them through the Twitter search API.

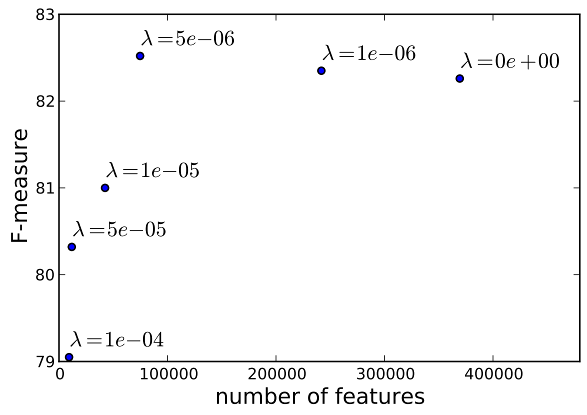
Metrics Prior work on these datasets has assumed perfect detection of words requiring normalization, and has focused on finding the correct normalization for these words (Han and Baldwin, 2011; Han et al., 2013). Recall has been defined as the proportion of words requiring normalization which are normalized correctly; precision is defined as the proportion of normalizations which are correct.

Results We run our training algorithm for two iterations (pass the training data twice). The results are presented in Table 2. Our system, UNLOL, achieves the highest published F-measure on both datasets. Performance on LexNorm1.2 is very similar to LexNorm1.1, despite the fact that roughly 8% of the examples were relabeled.

In the normalization task that we consider, the tokens to be normalized are specified in advance. This is the same task specification as in the prior work against which we compare. At test time, our system attempts normalizes all such tokens; every error is thus both a false positive and false negative, so precision equals to recall for this task; this is also true for Han and Baldwin (2011) and Liu et al. (2011).

It is possible to trade recall for precision by refusing to normalize words when the system‘s confidence falls below a threshold. A good setting of this threshold can improve the F-measure, but we did not report these results because we have no development set for parameter tuning.

Regularization One potential concern is that the number of non-zero feature weights will continually increase until the memory cost becomes overwhelming. Although we did not run up against mem-



λ	Dataset	F-measure	# of features
10^{-4}	LexNorm 1.1	79.05	9,281
5×10^{-5}		80.32	11,794
10^{-5}		81.00	42,466
5×10^{-6}		**82.52**	74,744
10^{-6}		82.35	241,820
0		82.26	369,366
5×10^{-6}	LexNorm 1.2	82.23	74,607

Figure 1: Effect of L1 regularization on the F-measure and the number of features with non-zero weights

ory limitations in the experiments producing the results in Table 2, this issue can be addressed through the application of L1 regularization, which produces sparse weight vectors by adding a penalty of $\lambda||\theta||_1$ to the log-likelihood. We perform online optimization of the L1-regularized log-likelihood by applying the truncated gradient method (Langford et al., 2009). We use an exponential decreasing learning rate $\eta_k = \eta_0 \alpha^{k/N}$, where k is the iteration counter and N is the size of training data. We set $\eta_0 = 1$ and $\alpha = 0.5$. Experiments were run until 300,000 training instances were observed, with a final learning rate of less than $1/32$. As shown in Figure 1, a small amount of regularization can dramatically decrease the number of active features without harming performance.

7 Analysis

We apply our normalization system to investigate the orthographic processes underlying language variation in social media. Using a dataset of 400,000 English language tweets, sampled from the month of August in each year from 2009 to 2012, we apply UNLOL to automatically normalize each token. We then treat these normalizations as labeled training data, and examine the Levenshtein alignment between the source and target tokens. This alignment gives approximate character-level transduction rules to explain each OOV token. We then examine which rules are used by each author, constructing a matrix of authors and rules.[2]

Factorization of the author-rule matrix reveals sets of rules that tend to be used together; we might call these rulesets “orthographic styles.” We apply non-negative matrix factorization (Lee and Seung, 2001), which characterizes each author by a vector of k style loadings, and simultaneously constructs k style dictionaries, which each put weight on different orthographic rules. Because the loadings are constrained to be non-negative, the factorization can be seen as sparsely assigning varying amounts of each style to each author. We choose the factorization that minimizes the Frobenius norm of the reconstruction error, using the NIMFA software package (`http://nimfa.biolab.si/`).

The resulting styles are shown in Table 3, for $k = 10$; other values of k give similar overall results with more or less detail. The styles incorporate a number of linguistic phenomena, including: expressive lengthening (styles 7-9; see Brody and Diakopoulos, 2011); g- and t-dropping (style 5, see Eisenstein 2013a) ; th-stopping (style 6); and the dropping of several word-final vowels (styles 1-3). Some of these styles, such as t-dropping and th-stopping, have direct analogues in spoken language varieties (Tagliamonte and Temple, 2005; Green, 2002), while others, like expressive lengthening, seem more unique to social media. The relationships between these orthographic styles and social variables such as geography and demograph-

[2]We tried adding these rules as features and retraining the normalization system, but this hurt performance.

style	rules	examples
1. you; o-dropping	y/_ ou/_u *y/⋆_ o/_	u, yu, 2day, knw, gud, yur, wud, yuh, u've, toda, everthing, everwhere, ourself
2. e-dropping, u/o	be/b_ e/_ o/u e*/_⋆	b, r, luv, cum, hav, mayb, bn, remembr, btween, gunna, gud
3. a-dropping	a/_ *a/⋆_ re/r_ ar/_r	r, tht, wht, yrs, bck, strt, gurantee, elementry, wr, rlly, wher, rdy, preciate, neway
4. g-dropping	g*/_⋆ ng/n_ g/_	goin, talkin, watchin, feelin, makin
5. t-dropping	t*/_⋆ st/s_ t/_	jus, bc, shh, wha, gota, wea, mus, firts, jes, subsistutes
6. th-stopping	h/_ *t/⋆d th/d_ t/d	dat, de, skool, fone, dese, dha, shid, dhat, dat's
7. (kd)-lengthening	i_/id _/k _/d _*/k⋆	idk, fuckk, okk, backk, workk, badd, andd, goodd, bedd, elidgible, pidgeon
8. o-lengthening	o_/oo _*/o⋆ _/o	soo, noo, doo, oohh, loove, thoo, helloo
9. e-lengthening	_/i e_/ee _/e _*/e⋆	mee, ive, retweet, bestie, lovee, nicee, heey, likee, iphone, homie, ii, damnit
10. a-adding	_/a __/ma _/m _*/a⋆	ima, outta, needa, shoulda, woulda, mm, comming, tomm, boutt, ppreciate

Table 3: Orthographic styles induced from automatically normalized Twitter text

ics must be left to future research, but they offer a promising generalization of prior work that has focused almost exclusively on exclusively on lexical variation (Argamon et al., 2007; Eisenstein et al., 2010; Eisenstein et al., 2011), with a few exceptions for character-level features (Brody and Diakopoulos, 2011; Burger et al., 2011).

Note that style 10 is largely the result of mistaken normalizations. The tokens ima, outta, and needa all refer to multi-word expressions in standard English, and are thus outside the scope of the normalization task as defined by Han et al. (2013). UNLOL has produced incorrect single-token normalizations for these terms: *i*/ima, *out*/outta, and *need*/needa. But while these normalizations are wrong, the resulting style nonetheless captures a coherent orthographic phenomenon.

8 Conclusion

We have presented a unified, unsupervised statistical model for normalizing social media text, attaining the best reported performance on the two standard normalization datasets. The power of our approach comes from flexible modeling of word-to-word relationships through features, while exploiting contextual regularity to train the corresponding feature weights without labeled data. The primary technical challenge was overcoming the large label space of the normalization task; we accomplish this using sequential Monte Carlo. Future work may consider whether sequential Monte Carlo can offer similar advantages in other unsupervised NLP tasks. An additional benefit of our joint statistical approach is that it may be combined with other downstream language processing tasks, such as part-of-speech tagging (Gimpel et al., 2011) and named entity resolution (Liu et al., 2012b).

Acknowledgments

We thank the reviewers for thoughtful comments on our submission. This work also benefitted from discussions with Timothy Baldwin, Paul Cook, Frank Dellaert, Arnoud Doucet, Micha Elsner, and Sharon Goldwater. It was supported by NSF SOCS-1111142.

References

S. Argamon, M. Koppel, J. Pennebaker, and J. Schler. 2007. Mining the blogosphere: age, gender, and the varieties of self-expression. *First Monday*, 12(9).

AiTi Aw, Min Zhang, Juan Xiao, and Jian Su. 2006. A phrase-based statistical model for SMS text normalization. In *Proceedings of ACL*, pages 33–40.

Richard Beaufort, Sophie Roekhaut, Louise-Amélie Cougnon, and Cédrick Fairon. 2010. A hybrid rule/model-based finite-state framework for normalizing sms messages. In *Proceedings of ACL*, pages 770–779.

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Proceedings of NAACL*, pages 582–590.

Samuel Brody and Nicholas Diakopoulos. 2011. Cooooooooooooooollllllllllllll!!!!!!!!!!!!!!: using word lengthening to detect sentiment in microblogs. In *Proceedings of EMNLP*.

John D. Burger, John C. Henderson, George Kim, and Guido Zarrella. 2011. Discriminating gender on twitter. In *Proceedings of EMNLP*.

Olivier Cappe, Simon J. Godsill, and Eric Moulines. 2007. An overview of existing methods and recent advances in sequential monte carlo. *Proceedings of the IEEE*, 95(5):899–924, May.

M. Choudhury, R. Saraf, V. Jain, A. Mukherjee, S. Sarkar, and A. Basu. 2007a. Investigation and modeling of the structure of texting language. *International Journal on Document Analysis and Recognition*, 10(3):157–174.

Monojit Choudhury, Rahul Saraf, Vijit Jain, Animesh Mukherjee, Sudeshna Sarkar, and Anupam Basu. 2007b. Investigation and modeling of the structure of texting language. *International Journal of Document Analysis and Recognition (IJDAR)*, 10(3-4):157–174.

Danish Contractor, Tanveer A. Faruquie, and L. Venkata Subramaniam. 2010. Unsupervised cleansing of noisy text. In *Proceedings of COLING*, pages 189–196.

Paul Cook and Suzanne Stevenson. 2009. An unsupervised model for text message normalization. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, CALC '09, pages 71–78, Stroudsburg, PA, USA. Association for Computational Linguistics.

A. Doucet, N.J. Gordon, and V. Krishnamurthy. 2001. Particle filters for state estimation of jump markov linear systems. *Trans. Sig. Proc.*, 49(3):613–624, March.

Jacob Eisenstein, Brendan O'Connor, Noah A. Smith, and Eric P. Xing. 2010. A latent variable model for geographic lexical variation. In *Proceedings of EMNLP*.

Jacob Eisenstein, Noah A. Smith, and Eric P. Xing. 2011. Discovering sociolinguistic associations with structured sparsity. In *Proceedings of ACL*.

Jacob Eisenstein. 2013a. Phonological factors in social media writing. In *Proceedings of the NAACL Workshop on Language Analysis in Social Media*.

Jacob Eisenstein. 2013b. What to do about bad language on the internet. In *Proceedings of NAACL*, pages 359–369.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for twitter: annotation, features, and experiments. In *Proceedings of ACL*.

Simon J. Godsill, Arnaud Doucet, and Mike West. 2004. Monte carlo smoothing for non-linear time series. In *Journal of the American Statistical Association*, pages 156–168.

Stephan Gouws, Dirk Hovy, and Donald Metzler. 2011. Unsupervised mining of lexical variants from noisy text. In *Proceedings of the First Workshop on Unsupervised Learning in NLP*, EMNLP '11.

Lisa J. Green. 2002. *African American English: A Linguistic Introduction*. Cambridge University Press, September.

Bo Han and Timothy Baldwin. 2011. Lexical normalisation of short text messages: makn sens a #twitter. In *Proceedings of ACL*, pages 368–378.

Bo Han, Paul Cook, and Timothy Baldwin. 2013. Lexical normalization for social media text. *ACM Transactions on Intelligent Systems and Technology*, 4(1):5.

Hany Hassan and Arul Menezes. 2013. Social text normalization using contextual graph random walks. In *Proceedings of ACL*.

Catherine Kobus, François Yvon, and Géraldine Damnati. 2008. Normalizing sms: are two metaphors better than one? In *Proceedings of COLING*, pages 441–448.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pages 282–289.

John Langford, Lihong Li, and Tong Zhang. 2009. Sparse online learning via truncated gradient. *The Journal of Machine Learning Research*, 10:777–801.

D. D. Lee and H. S. Seung. 2001. Algorithms for Non-Negative Matrix Factorization. In *Advances in Neural Information Processing Systems (NIPS)*, volume 13, pages 556–562.

Fei Liu, Fuliang Weng, Bingqing Wang, and Yang Liu. 2011. Insertion, deletion, or substitution?: normalizing text messages without pre-categorization nor supervision. In *Proceedings of ACL*, pages 71–76.

Fei Liu, Fuliang Weng, and Xiao Jiang. 2012a. A broad-coverage normalization system for social media language. In *Proceedings of ACL*, pages 1035–1044.

Xiaohua Liu, Ming Zhou, Xiangyang Zhou, Zhongyang Fu, and Furu Wei. 2012b. Joint inference of named entity recognition and normalization for tweets. In *Proceedings of ACL*.

Saša Petrović, Miles Osborne, and Victor Lavrenko. 2010. The edinburgh twitter corpus. In *Proceedings of the NAACL HLT Workshop on Computational Linguistics in a World of Social Media*, pages 25–26.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 354–362, Stroudsburg, PA, USA. Association for Computational Linguistics.

R. Sproat, A.W. Black, S. Chen, S. Kumar, M. Ostendorf, and C. Richards. 2001. Normalization of non-standard words. *Computer Speech & Language*, 15(3):287–333.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proceedings of ICSLP*, pages 901–904.

Sali Tagliamonte and Rosalind Temple. 2005. New perspectives on an ol' variable: (t,d) in british english. *Language Variation and Change*, 17:281–302, September.

Congle Zhang, Tyler Baldwin, Howard Ho, Benny Kimelfeld, and Yunyao Li. 2013. Adaptive parser-centric text normalization. In *Proceedings of ACL*, pages 1159–1168.