Adaptive Parser-Centric Text Normalization

Congle Zhang*
Dept of Computer Science and Engineering
University of Washington, Seattle, WA 98195, USA
clzhang@cs.washington.edu

Tyler Baldwin **Howard Ho** **Benny Kimelfeld** **Yunyao Li**
IBM Research - Almaden
650 Harry Road, San Jose, CA 95120, USA
{tbaldwi,ctho,kimelfeld,yunyaoli}@us.ibm.com

Abstract

Text normalization is an important first step towards enabling many Natural Language Processing (NLP) tasks over informal text. While many of these tasks, such as parsing, perform the best over fully grammatically correct text, most existing text normalization approaches narrowly define the task in the *word-to-word* sense; that is, the task is seen as that of mapping all out-of-vocabulary non-standard words to their in-vocabulary standard forms. In this paper, we take a parser-centric view of normalization that aims to convert raw informal text into grammatically correct text. To understand the real effect of normalization on the parser, we tie normalization performance directly to parser performance. Additionally, we design a customizable framework to address the often overlooked concept of domain adaptability, and illustrate that the system allows for transfer to new domains with a minimal amount of data and effort. Our experimental study over datasets from three domains demonstrates that our approach outperforms not only the state-of-the-art word-to-word normalization techniques, but also manual word-to-word annotations.

1 Introduction

Text normalization is the task of transforming informal writing into its standard form in the language. It is an important processing step for a wide range of Natural Language Processing (NLP) tasks such as text-to-speech synthesis, speech recognition, information extraction, parsing, and machine translation (Sproat et al., 2001).

The use of normalization in these applications poses multiple challenges. First, as it is most often conceptualized, normalization is seen as the task of mapping all out-of-vocabulary non-standard word tokens to their in-vocabulary standard forms. However, the scope of the task can also be seen as much wider, encompassing whatever actions are required to convert the raw text into a fully grammatical sentence. This broader definition of the normalization task may include modifying punctuation and capitalization, and adding, removing, or reordering words. Second, as with other NLP techniques, normalization approaches are often focused on one primary domain of interest (e.g., Twitter data). Because the style of informal writing may be different in different data sources, tailoring an approach towards a particular data source can improve performance in the desired domain. However, this is often done at the cost of adaptability.

This work introduces a customizable normalization approach designed with domain transfer in mind. In short, customization is done by providing the normalizer with *replacement generators*, which we define in Section 3. We show that the introduction of a small set of domain-specific generators and training data allows our model to outperform a set of competitive baselines, including state-of-the-art word-to-word normalization. Additionally, the flexibility of the model also allows it to attempt to produce fully grammatical sentences, something not typically handled by word-to-word normalization approaches.

Another potential problem with state-of-the-art normalization is the lack of appropriate evaluation metrics. The normalization task is most frequently motivated by pointing to the need for clean text for downstream processing applications, such as syntactic parsing. However, most studies of normalization give little insight into whether and to what degree the normalization process improves

*This work was conducted at IBM.

Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, pages 1159–1168,
Sofia, Bulgaria, August 4-9 2013.

the performance of the downstream application. For instance, it is unclear how performance measured by the typical normalization evaluation metrics of word error rate and BLEU score (Papineni et al., 2002) translates into performance on a parsing task, where a well placed punctuation mark may provide more substantial improvements than changing a non-standard word form. To address this problem, this work introduces an evaluation metric that ties normalization performance directly to the performance of a downstream dependency parser.

The rest of this paper is organized as follows. In Section 2 we discuss previous approaches to the normalization problem. Section 3 presents our normalization framework, including the actual normalization and learning procedures. Our instantiation of this model is presented in Section 4. In Section 5 we introduce the parser driven evaluation metric, and present experimental results of our model with respect to several baselines in three different domains. Finally, we discuss our experimental study in Section 6 and conclude in Section 7.

2 Related Work

Sproat et al. (2001) took the first major look at the normalization problem, citing the need for normalized text for downstream applications. Unlike later works that would primarily focus on specific noisy data sets, their work is notable for attempting to develop normalization as a general process that could be applied to different domains. The recent rise of heavily informal writing styles such as Twitter and SMS messages set off a new round of interest in the normalization problem.

Research on SMS and Twitter normalization has been roughly categorized as drawing inspiration from three other areas of NLP (Kobus et al., 2008): machine translation, spell checking, and automatic speech recognition. The statistical machine translation (SMT) metaphor was the first proposed to handle the text normalization problem (Aw et al., 2006). In this mindset, normalizing SMS can be seen as a translation task from a source language (informal) to a target language (formal), which can be undertaken with typical noisy channel based models. Work by Choudhury et al. (2007) adopted the spell checking metaphor, casting the problem in terms of character-level, rather than word-level, edits. They proposed an HMM based model that takes into account both grapheme and phoneme information. Kobus et al. (2008) undertook a hybrid approach that pulls inspiration from both the machine translation and speech recognition metaphors.

Many other approaches have been examined, most of which are at least partially reliant on the above three metaphors. Cook and Stevenson (2009) perform an unsupervised method, again based on the noisy channel model. Pennell and Liu (2011) developed a CRF tagger for deletion-based abbreviation on tweets. Xue et al. (2011) incorporated orthographic, phonetic, contextual, and acronym expansion factors to normalize words in both Twitter and SMS. Liu et al. (2011) modeled the generation process from dictionary words to non-standard tokens under an unsupervised sequence labeling framework. Han and Baldwin (2011) use a classifier to detect ill-formed words, and then generate correction candidates based on morphophonemic similarity. Recent work has looked at the construction of normalization dictionaries (Han et al., 2012) and on improving coverage by integrating different human perspectives (Liu et al., 2012).

Although it is almost universally used as a motivating factor, most normalization work does not directly focus on improving downstream applications. While a few notable exceptions highlight the need for normalization as part of text-to-speech systems (Beaufort et al., 2010; Pennell and Liu, 2010), these works do not give any direct insight into how much the normalization process actually improves the performance of these systems. To our knowledge, the work presented here is the first to clearly link the output of a normalization system to the output of the downstream application. Similarly, our work is the first to prioritize domain adaptation during the new wave of text message normalization.

3 Model

In this section we introduce our normalization framework, which draws inspiration from our previous work on spelling correction for search (Bao et al., 2011).

3.1 Replacement Generators

Our input the original, unnormalized text, represented as a sequence $\mathbf{x} = x_1, x_2, \ldots, x_n$ of *tokens* x_i. In this section we will use the following se-

quence as our running example:

$$\mathbf{x} = \texttt{Ay}_1\ \texttt{woudent}_2\ \texttt{of}_3\ \texttt{see}_4\ \texttt{'em}_5$$

where space replaces comma for readability, and each token is subscripted by its position. Given the input $\mathbf{x}$, we apply a series of replacement generators, where a *replacement generator* is a function that takes $\mathbf{x}$ as input and produces a collection of replacements. Here, a *replacement* is a statement of the form "replace tokens $x_i, \ldots, x_{j-1}$ with $\mathbf{s}$." More precisely, a replacement is a triple $\langle i, j, \mathbf{s} \rangle$, where $1 \leq i \leq j \leq n+1$ and $\mathbf{s}$ is a sequence of tokens. Note that in the case where $i = j$, the sequence $\mathbf{s}$ should be *inserted* right before x_i; and in the special case where $\mathbf{s}$ is empty, we simply delete $x_i, \ldots, x_{j-1}$. For instance, in our running example the replacement $\langle 2, 3, \texttt{would not} \rangle$ replaces $x_2 = \texttt{woudent}$ with `would not`; $\langle 1, 2, \texttt{Ay} \rangle$ replaces x_1 with itself (hence, does not change $\mathbf{x}$); $\langle 1, 2, \epsilon \rangle$ (where ϵ is the empty sequence) deletes x_1; $\langle 6, 6, \texttt{.} \rangle$ inserts a period at the end of the sequence.

The provided replacement generators can be either generic (cross domain) or domain-specific, allowing for domain customization. In Section 4, we discuss the replacement generators used in our empirical study.

3.2 Normalization Graph

Given the input $\mathbf{x}$ and the set of replacements produced by our generators, we associate a unique Boolean variable X_r with each replacement r. As expected, X_r being $\mathbf{true}$ means that the replacement r takes place in producing the output sequence.

Next, we introduce dependencies among variables. We first discuss the syntactic consistency of truth assignments. Let $r_1 = \langle i_1, j_1, \mathbf{s}_1 \rangle$ and $r_2 = \langle i_2, j_2, \mathbf{s}_2 \rangle$ be two replacements. We say that r_1 and r_2 are *locally consistent* if the intervals $[i_1, j_1)$ and $[i_2, j_2)$ are disjoint. Moreover, we do not allow two insertions to take place at the same position; therefore, we exclude $[i_1, j_1)$ and $[i_2, j_2)$ from the definition of local consistency when $i_1 = j_1 = i_2 = j_2$. If r_1 and r_2 are locally consistent and $j_1 = i_2$, then we say that r_2 is a *consistent follower* of r_1.

A truth assignment α to our variables X_r is *sound* if every two replacements r and r' with $\alpha(X_r) = \alpha(X_{r'}) = \mathbf{true}$ are locally consistent. We say that α is *complete* if every token of $\mathbf{x}$ is captured by at least one replacement r with $\alpha(X_r) = \mathbf{true}$. Finally, we say that α is *legal* if it is sound and complete. The output (normalized sequence) defined by a legal assignment α is, naturally, the concatenation (from left to right) of the strings $\mathbf{s}$ in the replacements $r = \langle i, j, \mathbf{s} \rangle$ with $\alpha(X_r) = \mathbf{true}$. In Figure 1, for example, if the nodes with a grey shade are the ones associated with $\mathbf{true}$ variables under α, then the output defined by α is `I would not have seen them`.

Our variables carry two types of interdependencies. The first is that of syntactic consistency: the entire assignment is required to be legal. The second captures correlation among replacements. For instance, if we replace `of` with `have` in our running example, then the next `see` token is more likely to be replaced with `seen`. In this work, dependencies of the second type are restricted to pairs of variables, where each pair corresponds to a replacement and a consistent follower thereof.

The above dependencies can be modeled over a standard undirected graph using Conditional Random Fields (Lafferty et al., 2001). However, the graph would be complex: in order to model local consistency, there should be edges between every two nodes that violate local consistency. Such a model renders inference and learning infeasible. Therefore, we propose a clearer model by a directed graph, as illustrated in Figure 1 (where nodes are represented by replacements r instead of the variables X_r, for readability). To incorporate correlation among replacements, we introduce an edge from X_r to $X_{r'}$ whenever r' is a consistent follower of r. Moreover, we introduce two dummy nodes, start and end, with an edge from start to each variable that corresponds to a prefix of the input sequence $\mathbf{x}$, and an edge from each variable that corresponds to a suffix of $\mathbf{x}$ to end.

The principal advantage of modeling the dependencies in such a directed graph is that now, the legal assignments are in one-to-one correspondence with the paths from start to end; this is a straightforward observation that we do not prove here.

We appeal to the log-linear model formulation to define the probability of an assignment. The conditional probability of an assignment α, given an input sequence $\mathbf{x}$ and the weight vector $\Theta = \langle \theta_1, \ldots, \theta_k \rangle$ for our features, is defined as $p(\alpha \mid$

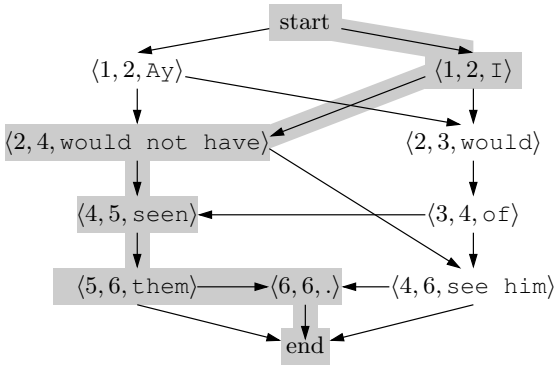

Figure 1: Example of a normalization graph; the nodes are replacements generated by the replacement generators, and every path from start to end implies a legal assignment

$\mathbf{x}, \Theta) = 0$ if α is not legal, and otherwise,

$$p(\alpha \mid \mathbf{x}, \Theta) = \frac{1}{Z(\mathbf{x})} \prod_{X \to Y \in \alpha} \exp(\sum_j \theta_j \phi_j(X, Y, \mathbf{x})) \,.$$

Here, $Z(\mathbf{x})$ is the partition function, $X \to Y \in \alpha$ refers to an edge $X \to Y$ with $\alpha(X) = \mathbf{true}$ and $\alpha(Y) = \mathbf{true}$, and $\phi_1(X, Y, \mathbf{x}), \ldots, \phi_k(X, Y, \mathbf{x})$ are real valued feature functions that are weighted by $\theta_1, \ldots, \theta_k$ (the model's parameters), respectively.

3.3 Inference

When performing inference, we wish to select the output sequence with the highest probability, given the input sequence $\mathbf{x}$ and the weight vector Θ (i.e., MAP inference). Specifically, we want an assignment $\alpha^\star = \arg\max_\alpha p(\alpha \mid \mathbf{x}, \Theta)$.

While exact inference is computationally hard on general graph models, in our model it boils down to finding the longest path in a weighted and acyclic directed graph. Indeed, our directed graph (illustrated in Figure 1) is acyclic. We assign the real value $\sum_j \theta_j \phi_j(X, Y, \mathbf{x})$ to the edge $X \to Y$, as the weight. As stated in Section 3.2, a legal assignment α corresponds to a path from start to end; moreover, the sum of the weights on that path is equal to $\log p(\alpha \mid \mathbf{x}, \Theta) + \log Z(\mathbf{x})$. In particular, a longer path corresponds to an assignment with greater probability. Therefore, we can solve the MAP inference within our model by finding the weighted longest path in the directed acyclic graph. The algorithm in Figure 2 summarizes the inference procedure to normalize the input sequence $\mathbf{x}$.

Input:
1. A sequence $\mathbf{x}$ to normalize;
2. A weight vector $\Theta = \langle \theta_1, \ldots, \theta_k \rangle$.

Generate replacements: Apply all replacement generators to get a set of replacements r, each r is a triple $\langle i, j, \mathbf{s} \rangle$.

Build a normalization graph:
1. For each replacement r, create a node X_r.
2. For each r' and r, create an edge X_r to $X_{r'}$ if r' is a consistent follower of r.
3. Create two dummy nodes start and end, and create edges from start to all prefix nodes and end to all suffix nodes.
4. For each edge $X \to Y$, compute the features $\phi_j(X, Y, \mathbf{x})$, and weight the edge by $\sum_j \theta_j \phi_j(X, Y, \mathbf{x})$.

MAP Inference: Find a weighted longest path P from start to end, and return α^*, where $\alpha^*(X_r) = \mathbf{true}$ iff $X_r \in P$.

Figure 2: Normalization algorithm

3.4 Learning

Our labeled data consists of pairs $(\mathbf{x}_i, \mathbf{y}_i^{\mathsf{gold}})$, where $\mathbf{x}_i$ is an input sequence (to normalize) and $\mathbf{y}_i^{\mathsf{gold}}$ is a (manually) normalized sequence. We obtain a truth assignment α_i^{gold} from each $\mathbf{y}_i^{\mathsf{gold}}$ by selecting an assignment α that minimizes the edit distance between $\mathbf{y}_i^{\mathsf{gold}}$ and the normalized text implied by α:

$$\alpha_i^{\mathsf{gold}} = \arg\min_\alpha \mathrm{DIST}(y(\alpha), \mathbf{y}_i^{\mathsf{gold}}) \qquad (1)$$

Here, $y(\alpha)$ denotes the normalized text implied by α, and DIST is a token-level edit distance. We apply a simple dynamic-programming algorithm to compute α_i^{gold}. Finally, the items in our training data are the pairs $(\mathbf{x}_i, \alpha_i^{\mathsf{gold}})$.

Learning over similar models is commonly done via maximum likelihood estimation:

$$L(\Theta) = \log \prod_i p(\alpha_i = \alpha_i^{\mathsf{gold}} \mid \mathbf{x}_i, \Theta)$$

Taking the partial derivative gives the following:

$$\sum_i \Big(\Phi_j(\alpha_i^{\mathsf{gold}}, \mathbf{x}_i) - E_{p(\alpha_i \mid \mathbf{x}_i, \Theta)} \Phi_j(\alpha_i, \mathbf{x}_i) \Big)$$

where $\Phi_j(\alpha, \mathbf{x}) = \sum_{X \to Y} \phi_j(X, Y, \mathbf{x})$, that is, the sum of values for the jth feature along the

Input:

1. A set $\{(\mathbf{x}_i, \mathbf{y}_i^{\mathsf{gold}})\}_{i=1}^{n}$ of sequences and their gold normalization;
2. Number T of iterations.

Initialization: Initialize each θ_j as zero, and obtain each α_i^{gold} according to (1).

Repeat T times:

1. Infer each α_i^* from $\mathbf{x}_i$ using the current Θ;
2. $\theta_j \leftarrow \theta_j + \sum_i (\Phi_j(\alpha_i^{\mathsf{gold}}, \mathbf{x}_i) - \Phi_j(\alpha_i^*, \mathbf{x}_i))$ for all $j = 1, \ldots, k$.

Output: $\Theta = \langle \theta_1, \ldots, \theta_k \rangle$

Figure 3: Learning algorithm

path defined by α, and $E_{p(\alpha_i|\mathbf{x}_i,\Theta)}\Phi_j(\alpha_i, \mathbf{x}_i)$ is the expected value of that sum (over all legal assignments α_i), assuming the current weight vector.

How to efficiently compute $E_{p(\alpha_i|\mathbf{x}_i,\Theta)}\Phi_j(\alpha_i, \mathbf{x}_i)$ in our model is unclear; naively, it requires enumerating all legal assignments. We instead opt to use a more tractable perceptron-style algorithm (Collins, 2002). Instead of computing the expectation, we simply compute $\Phi_j(\alpha_i^*, \mathbf{x}_i)$, where α_i^* is the assignment with the highest probability, generated using the current weight vector. The result is then:

$$\sum_i \Big(\Phi_j(\alpha_i^{\mathsf{gold}}, \mathbf{x}_i) - \Phi_j(\alpha_i^*, \mathbf{x}_i) \Big)$$

Our learning applies the following two steps iteratively. (1) Generate the most probable sequence within the current weights. (2) Update the weights by comparing the path generated in the previous step to the gold standard path. The algorithm in Figure 3 summarizes the procedure.

4 Instantiation

In this section, we discuss our instantiation of the model presented in the previous section. In particular, we describe our replacement generators and features.

4.1 Replacement Generators

One advantage of our proposed model is that the reliance on replacement generators allows for strong flexibility. Each generator can be seen as a black box, allowing replacements that are created heuristically, statistically, or by external tools to be incorporated within the same framework.

Generator	From	To
leave intact	good	good
edit distance	bac	back
lowercase	NEED	need
capitalize	it	It
Google spell	disspaear	disappear
contraction	wouldn't	would not
slang language	ima	I am going to
insert punctuation	ϵ	.
duplicated punctuation	!?	!
delete filler	lmao	ϵ

Table 1: Example replacement generators

To build a set of generic replacement generators suitable for normalizing a variety of data types, we collected a set of about 400 Twitter posts as development data. Using that data, a series of generators were created; a sample of them are shown in Table 1. As shown in the table, these generators cover a variety of normalization behavior, from changing non-standard word forms to inserting and deleting tokens.

4.2 Features

Although the proposed framework supports real valued features, all features in our system are binary. In total, we used 70 features. Our feature set pulls information from several different sources:

N-gram: Our n-gram features indicate the frequency of the phrases induced by an edge. These features are turned into binary ones by bucketing their log values. For example, on the edge from $\langle 1, 2, \texttt{I} \rangle$ to $\langle 2, 3, \texttt{would} \rangle$ such a feature will indicate whether the frequency of `I would` is over a threshold. We use the Corpus of Contemporary English (Davies, 2008) to produce our n-gram information.

Part-of-speech: Part-of-speech information can be used to produce features that encourage certain behavior, such as avoiding the deletion of noun phrases. We generate part-of-speech information over the original raw text using a Twitter part-of-speech tagger (Ritter et al., 2011). Of course, the part-of-speech information obtained this way is likely to be noisy, and we expect our learning algorithm to take that into account.

Positional: Information from positions is used primarily to handle capitalization and punctuation insertion, for example, by incorporating features for capitalized words after stop punctuation or the insertion of stop punctuation at the end of the sentence.

Lineage: Finally, we include binary features

that indicate which generator spawned the replacement.

5 Evaluation

In this section, we present an empirical study of our framework. The study is done over datasets from three different domains. The goal is to evaluate the framework in two aspects: (1) usefulness for downstream applications (specifically dependency parsing), and (2) domain adaptability.

5.1 Evaluation Metrics

A few different metrics have been used to evaluate normalizer performance, including word error rate and BLEU score. While each metric has its pros and cons, they all rely on word-to-word matching and treat each word equally. In this work, we aim to evaluate the performance of a normalizer based on how it affects the performance of downstream applications. We find that the conventional metrics are not directly applicable, for several reasons. To begin with, the assumption that words have equal weights is unlikely to hold. Additionally, these metrics tend to ignore other important non-word information such as punctuation or capitalization. They also cannot take into account other aspects that may have an impact on downstream performance, such as the word reordering as seen in the example in Figure 4. Therefore, we propose a new evaluation metric that directly equates normalization performance with the performance of a common downstream application—dependency parsing.

To realize our desired metric, we apply the following procedure. First, we produce gold standard normalized data by manually normalizing sentences to their full grammatically correct form. In addition to the word-to-word mapping performed in typical normalization gold standard generation, this annotation procedure includes all actions necessary to make the sentence grammatical, such as word reordering, modifying capitalization, and removing emoticons. We then run an off-the-shelf dependency parser on the gold standard normalized data to produce our gold standard parses. Although the parser could still produce mistakes on the grammatical sentences, we feel that this provides a realistic benchmark for comparison, as it represents an upper bound on the possible performance of the parser, and avoids an expensive second round of manual annotation.

Test *I kinda wanna get ipad NEW*	**Gold** *I kind of want to get a new iPad.*	**SVO**
verb(get)	verb(want) **verb(get)**	$\text{precision}_{\text{v}} = \frac{1}{1}$ $\text{recall}_{\text{v}} = \frac{1}{2}$
subj(get,I) subj(get,wanna) obj(get,NEW)	subj(want,I) **subj(get,I)** obj(get,iPad)	$\text{precision}_{\text{so}} = \frac{1}{3}$ $\text{recall}_{\text{so}} = \frac{1}{3}$

Figure 4: The subjects, verbs, and objects identified on example test/gold text, and corresponding metric scores

To compare the parses produced over automatically normalized data to the gold standard, we look at the subjects, verbs, and objects (SVO) identified in each parse. The metric shown in Equations (2) and (3) below is based on the identified subjects and objects in those parses. Note that SO denotes the set of identified subjects and objects whereas SO^{gold} denotes the set of subjects and objects identified when parsing the gold-standard normalization.

$$\text{precision}_{\text{so}} = \frac{|SO \cap SO^{\text{gold}}|}{|SO|} \quad (2)$$

$$\text{recall}_{\text{so}} = \frac{|SO \cap SO^{\text{gold}}|}{|SO^{\text{gold}}|} \quad (3)$$

We similarly define $\text{precision}_{\text{v}}$ and recall_{v}, where we compare the set V of identified verbs to V^{gold} of those found in the gold-standard normalization. An example is shown in Figure 4.

5.2 Results

To establish the extensibility of our normalization system, we present results in three different domains: Twitter posts, Short Message Service (SMS) messages, and call-center logs. For Twitter and SMS messages, we used established datasets to compare with previous work. As no established call-center log dataset exists, we collected our own. In each case, we ran the proposed system with two different configurations: one using only the generic replacement generators presented in Section 4 (denoted as *generic*), and one that adds additional domain-specific generators for the corresponding domain (denoted as *domain-specific*). All runs use ten-fold cross validation for training and evaluation. The Stanford parser[1] (Marneffe et al., 2006) was used to produce all dependency

[1]Version 2.0.4, `http://nlp.stanford.edu/software/lex-parser.shtml`

parses. We compare our system to the following baseline solutions:

w/oN: No normalization is performed.

Google: Output of the Google spell checker.

w2wN: The output of the word-to-word normalization of Han and Baldwin (2011). Not available for call-center data.

Gw2wN: The manual gold standard word-to-word normalizations of previous work (Choudhury et al., 2007; Han and Baldwin, 2011). Not available for call-center data.

Our results use the metrics of Section 5.1.

5.2.1 Twitter

To evaluate the performance on Twitter data, we use the dataset of randomly sampled tweets produced by (Han and Baldwin, 2011). Because the gold standard used in this work only provided word mappings for out-of-vocabulary words and did not enforce grammaticality, we reannotated the gold standard data[2]. Their original gold standard annotations were kept as a baseline.

To produce Twitter-specific generators, we examined the Twitter development data collected for generic generator production (Section 4). These generators focused on the Twitter-specific notions of hashtags (#), ats (@), and retweets (RT). For each case, we implemented generators that allowed for either the initial symbol or the entire token to be deleted (e.g., @Hertz to Hertz, @Hertz to ϵ).

The results are given in Table 2. As shown, the domain-specific generators yielded performance significantly above the generic ones and all baselines. Even without domain-specific generators, our system outperformed the word-to-word normalization approaches. Most notably, both the generic and domain-specific systems outperformed the gold standard word-to-word normalizations. These results validate the hypothesis that simple word-to-word normalization is insufficient if the goal of normalization is to improve dependency parsing; even if a system could produce perfect word-to-word normalization, it would produce lower quality parses than those produced by our approach.

[2]Our results and the reannotations of the Twitter and SMS data are available at `https://www.cs.washington.edu/node/9091/`

System	Verb			Subject-Object		
	Pre	Rec	F1	Pre	Rec	F1
w/oN	83.7	68.1	75.1	31.7	38.6	34.8
Google	88.9	78.8	83.5	36.1	46.3	40.6
w2wN	87.5	81.5	84.4	44.5	58.9	50.7
Gw2w	89.8	83.8	86.7	46.9	61.0	53.0
generic	91.7	88.9	90.3	53.6	70.2	60.8
domain specific	95.3	88.7	**91.9**	72.5	76.3	**74.4**

Table 2: Performance on Twitter dataset

5.2.2 SMS

To evaluate the performance on SMS data, we use the Treasure My Text data collected by Choudhury et al. (2007). As with the Twitter data, the word-to-word normalizations were reannotated to enforce grammaticality. As a replacement generator for SMS-specific substitutions, we used a mapping dictionary of SMS abbreviations.[3] No further SMS-specific development data was needed.

Table 3 gives the results on the SMS data. The SMS dataset proved to be more difficult than the Twitter dataset, with the overall performance of every system being lower. While this drop of performance may be a reflection of the difference in data styles between SMS and Twitter, it is also likely a product of the collection methodology. The collection methodology of the Treasure My Text dataset dictated that every message must have at least one mistake, which may have resulted in a dataset that was noisier than average.

Nonetheless, the trends on SMS data mirror those on Twitter data, with the domain-specific generators achieving the greatest overall performance. However, while the generic setting still manages to outperform most baselines, it did not outperform the gold word-to-word normalization. In fact, the gold word-to-word normalization was much more competitive on this data, outperforming even the domain-specific system on verbs alone. This should not be seen as surprising, as word-to-word normalization is most likely to be beneficial for cases like this where the proportion of non-standard tokens is high.

It should be noted that the SMS dataset as available has had all punctuation removed. While this may be appropriate for word-to-word normalization, this preprocessing may have an effect on the parse of the sentence. As our system has the ability to add punctuation but our baseline systems do not, this has the potential to artificially inflate our results. To ensure a fair comparison, we manually

[3]`http://www.netlingo.com/acronyms.php`

System	Verb			Subject-Object		
	Rec	Pre	F1	Rec	Pre	F1
w/oN	76.4	48.1	59.0	19.5	21.5	20.4
Google	85.1	61.6	71.5	22.4	26.2	24.1
w2wN	78.5	61.5	68.9	29.9	36.0	32.6
Gw2wN	87.6	76.6	**81.8**	38.0	50.6	43.4
generic	86.5	77.4	81.7	35.5	47.7	40.7
domain specific	88.1	75.0	81.0	41.0	49.5	**44.8**

Table 3: Performance on SMS dataset

System	Verb			Subject-Object		
	Pre	Rec	F1	Pre	Rec	F1
w/oN	98.5	97.1	97.8	69.2	66.1	67.6
Google	99.2	97.9	**98.5**	70.5	67.3	68.8
generic	98.9	97.4	98.1	71.3	67.9	69.6
domain specific	99.2	97.4	98.3	87.9	83.1	**85.4**

Table 4: Performance on call-center dataset

added punctuation to a randomly selected small subset of the SMS data and reran each system. This experiment suggested that, in contrast to the hypothesis, adding punctuation actually improved the results of the proposed system more substantially than that of the baseline systems.

5.2.3 Call-Center

Although Twitter and SMS data are unmistakably different, there are many similarities between the two, such as the frequent use of shorthand word forms that omit letters. The examination of call-center logs allows us to examine the ability of our system to perform normalization in more disparate domains. Our call-center data consists of text-based responses to questions about a user's experience with a call-center (e.g., their overall satisfaction with the service). We use call-center logs from a major company, and collect about 150 responses for use in our evaluation. We collected an additional small set of data to develop our call-center-specific generators.

Results on the call-center dataset are in Table 4. As shown, the raw call-center data was comparatively clean, resulting in higher baseline performance than in other domains. Unlike on previous datasets, the use of generic mappings only provided a small improvement over the baseline. However, the use of domain-specific generators once again led to significantly increased performance on subjects and objects.

6 Discussion

The results presented in the previous section suggest that domain transfer using the proposed normalization framework is possible with only a small amount of effort. The relatively modest set of additional replacement generators included in each data set allowed the domain-specific approaches to significantly outperform the generic approach. In the call-center case, performance improvements could be seen by referencing a very small amount of development data. In the SMS case, the presence of a domain-specific dictionary allowed for performance improvements without the need for any development data at all. It is likely, though not established, that employing further development data would result in further performance improvements. We leave further investigation to future work.

The results in Section 5.2 establish a point that has often been assumed but, to the best of our knowledge, has never been explicitly shown: performing normalization is indeed beneficial to dependency parsing on informal text. The parse of the normalized text was substantially better than the parse of the original raw text in all domains, with absolute performance increases ranging from about 18-25% on subjects and objects. Furthermore, the results suggest that, as hypothesized, preparing an informal text for a parsing task requires more than simple word-to-word normalization. The proposed approach significantly outperforms the state-of-the-art word-to-word normalization approach. Perhaps most interestingly, the proposed approach performs on par with, and in several cases superior to, gold standard word-to-word annotations. This result gives strong evidence for the conclusion that parser-targeted normalization requires a broader understanding of the scope of the normalization task.

While the work presented here gives promising results, there are still many behaviors found in informal text that prove challenging. One such example is the word reordering seen in Figure 4. Although word reordering could be incorporated into the model as a combination of a deletion and an insertion, the model as currently devised cannot easily link these two replacements to one another. Additionally, instances of reordering proved hard to detect in practice. As such, no reordering-based replacement generators were implemented in the presented system. Another case that proved difficult was the insertion of missing tokens. For instance, the informal sentence "`Day 3 still don't freaking`

feel good!:(" could be formally rendered as "**It is** day 3 **and I** still do not feel good!". Attempts to address missing tokens in the model resulted in frequent false positives. Similarly, punctuation insertion proved to be challenging, often requiring a deep analysis of the sentence. For example, contrast the sentence "I'm watching a movie I don't know its name." which would benefit from inserted punctuation, with "I'm watching a movie I don't know.", which would not. We feel that the work presented here provides a foundation for future work to more closely examine these challenges.

7 Conclusions

This work presents a framework for normalization with an eye towards domain adaptation. The proposed framework builds a statistical model over a series of replacement generators. By doing so, it allows a designer to quickly adapt a generic model to a new domain with the inclusion of a small set of domain-specific generators. Tests over three different domains suggest that, using this model, only a small amount of domain-specific data is necessary to tailor an approach towards a new domain.

Additionally, this work introduces a parser-centric view of normalization, in which the performance of the normalizer is directly tied to the performance of a downstream dependency parser. This evaluation metric allows for a deeper understanding of how certain normalization actions impact the output of the parser. Using this metric, this work established that, when dependency parsing is the goal, typical word-to-word normalization approaches are insufficient. By taking a broader look at the normalization task, the approach presented here is able to outperform not only state-of-the-art word-to-word normalization approaches but also manual word-to-word annotations.

Although the work presented here established that more than word-to-word normalization was necessary to produce parser-ready normalizations, it remains unclear which specific normalization tasks are most critical to parser performance. We leave this interesting area of examination to future work.

Acknowledgments

We thank the anonymous reviewers of ACL for helpful comments and suggestions. We also thank Ioana R. Stanoi for her comments on a preliminary version of this work, Daniel S. Weld for his support, and Alan Ritter, Monojit Choudhury, Bo Han, and Fei Liu for sharing their tools and data. The first author is partially supported by the DARPA Machine Reading Program under AFRL prime contract numbers FA8750-09-C-0181 and FA8750-09-C-0179. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA, AFRL, or the US government. This work is a part of IBM's SystemT project (Chiticariu et al., 2010).

References

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

Zhuowei Bao, Benny Kimelfeld, and Yunyao Li. 2011. A graph approach to spelling correction in domain-centric search. In *ACL*, pages 905–914.

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

Laura Chiticariu, Rajasekar Krishnamurthy, Yunyao Li, Sriram Raghavan, Frederick Reiss, and Shivakumar Vaithyanathan. 2010. SystemT: An algebraic approach to declarative information extraction. In *ACL*, pages 128–137.

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

Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *EMNLP*, pages 1–8.

Paul Cook and Suzanne Stevenson. 2009. An unsupervised model for text message normalization. In *CALC*, pages 71–78.

Mark Davies. 2008-. The corpus of contemporary american english: 450 million words, 1990-present. Avialable online at: http://corpus.byu.edu/coca/.

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

Bo Han, Paul Cook, and Timothy Baldwin. 2012. Automatically constructing a normalisation dictionary for microblogs. In *EMNLP-CoNLL*, pages 421–432.

Catherine Kobus, François Yvon, and Géraldine Damnati. 2008. Normalizing SMS: are two metaphors better than one? In *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 *ICML*, pages 282–289.

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

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

Marie-Catherine De Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC-06*, pages 449–454.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

Deana Pennell and Yang Liu. 2010. Normalization of text messages for text-to-speech. In *ICASSP*, pages 4842–4845.

Deana Pennell and Yang Liu. 2011. A character-level machine translation approach for normalization of SMS abbreviations. In *IJCNLP*, pages 974–982.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in Tweets: An experimental study. In *EMNLP*, pages 1524–1534.

Richard Sproat, Alan W. Black, Stanley F. Chen, Shankar Kumar, Mari Ostendorf, and Christopher Richards. 2001. Normalization of non-standard words. *Computer Speech & Language*, 15(3):287–333.

Zhenzhen Xue, Dawei Yin, and Brian D. Davison. 2011. Normalizing microtext. In *Analyzing Microtext*, volume WS-11-05 of *AAAI Workshops*.