Proceedings of the Thirteenth International AAAI Conference on Web and Social Media (ICWSM 2019)

Adapting Sequence to Sequence Models for Text Normalization in Social Media

Ismini Lourentzou, Kabir Manghnani, ChengXiang Zhai
University of Illinois at Urbana-Champaign, Urbana, IL, USA
{lourent2,kabirm2,czhai}@illinois.edu

Abstract

Social media offer an abundant source of valuable raw data, however informal writing can quickly become a bottleneck for many natural language processing (NLP) tasks. Off-the-shelf tools are usually trained on formal text and cannot explicitly handle noise found in short online posts. Moreover, the variety of frequently occurring linguistic variations presents several challenges, even for humans who might not be able to comprehend the meaning of such posts, especially when they contain slang and abbreviations. Text Normalization aims to transform online user-generated text to a canonical form. Current text normalization systems rely on string or phonetic similarity and classification models that work on a local fashion. We argue that processing contextual information is crucial for this task and introduce a social media text normalization hybrid word-character attention-based encoder-decoder model that can serve as a pre-processing step for NLP applications to adapt to noisy text in social media. Our character-based component is trained on synthetic adversarial examples that are designed to capture errors commonly found in online user-generated text. Experiments show that our model surpasses neural architectures designed for text normalization and achieves comparable performance with state-of-the-art related work.

Introduction

Most text data in the world today is user-generated and online. Vast quantities of online blogs and forums, social media posts, customer reviews, and other textual sources are necessary input of useful information for algorithms that understand user intent and preferences, predict trends or recommend items for purchase in targeted advertising. However, social media usually deviates from standard language usage, with high percentages of non-standard words, such as abbreviations, phonetic substitutions, hashtags, acronyms, internet slang, emoticons, grammatical and spelling errors.

Such non-standard words cause problems for both users and text mining applications. Users that are not familiar with domain-specific or peculiar language usage, e.g. acronyms found in Twitter messages, may experience problems in understanding the expressed content. Additionally, due to high out-of-vocabulary word rates, NLP approaches struggle with the noisy and informal nature of social media written language. Natural language follows a Zipfian distribution where most words are rare. Learning "long tail" word representations requires enormous amounts of data (Bahdanau et al. 2017). Recent work showcases the negative impact of noisy text on several NLP tasks and the improvement that text normalization can bring in part-of-speech tagging (Han, Cook, and Baldwin 2013), parsing (Zhang et al. 2013), and machine translation (Hassan and Menezes 2013) tasks. Special pre-processing of informal text is therefore necessary to help users understand content more easily and facilitate NLP algorithms. The task of transforming noisy or informal text to a more standard representation is called ***Text Normalization***.

Normalizing text is challenging and involves a trade-off between high recall, i.e. maximizing the number of corrections and high precision, i.e. minimize the number of incorrect normalizations. In several cases, the task is framed as mapping an out-of-vocabulary (OOV) non-standard word to an in-vocabulary (IV) standard one that preserves the meaning of the sentence. Additionally, text normalization can include modifications that are beyond the framework described above, for example replacing, removing or adding word tokens or punctuation and capitalize or lowercase text. Word mappings might not be unique, i.e. an OOV word can be transformed to more than one IV word, based on context. Due to the dynamic nature of social media text, many words (e.g. named entities) are considered OOV but do not need normalization or there is no appropriate IV word for them.

Although text normalization may appear to be similar to the task of spelling error correction, it is actually much more difficult to handle noisy social media text. Spelling correction focuses on word errors that can usually be handled with edit distance metrics. Additionally, grammatical error correction, which incorporates local spelling errors with global grammatical errors, e.g. preposition or verb usage mistakes, deals with replacing or adding omitted words, which are often caused *unintentionally* by non-native writers. Such errors can be partially identified with syntactic knowledge, e.g. semantic parsing, while it is rather unlikely that text normalization systems will benefit from such linguistic sources (Baldwin et al. 2015). Due to the new challenges in text normalization, it generally requires new approaches that go beyond the traditional spelling error correction methods.

The non-standard forms found in user-generated context can be mostly summarized into several categories:

1. **Misspellings**, e.g. “defenitely” $\rightarrow$ “definitely”
2. **Phonetic substitution** of characters with numbers or letters, e.g. “2morrow” $\rightarrow$ “tomorrow”, “rt” $\rightarrow$ “retweet”
3. **Shortening** of words, e.g. “convo” $\rightarrow$ “conversation”
4. **Acronyms**, e.g. “idk” $\rightarrow$ “i don’t know” that can also include standard words usually used as acronyms, e.g. “goat” $\rightarrow$ “greatest of all time”
5. **Slang**, i.e. metaphoric usage of standard words, e.g. “low key”, “woke” or “broccoli”
6. **Emphasis** given to a certain word, either by capitalization, e.g. “YEAH THIS IS SO COOL” or by vowel elongation e.g. “coooooool” $\rightarrow$ “cool”
7. **Punctuation** deleted or misplaced, e.g. “doesnt” $\rightarrow$ “doesn’t”, “do’nt” $\rightarrow$ “don’t” or intentionally using punctuation instead of letters, e.g. “b@n@n@s”

Early text normalization systems consider a pipeline of statistical language models, dependency parsing, string similarity, spell-checking and slang dictionaries (Liu et al. 2011; Han and Baldwin 2011; Han, Cook, and Baldwin 2013). However, the high-dimensional action space of language (arbitrary word sequences constructed from a vocabulary) makes unsupervised learning inefficient. Additionally, unsupervised text normalization methods often tune hyper-parameters based on annotated (supervised) data, thus are not fully unsupervised. Considering the rapid changes of language in online content, with many emerging words appearing daily, lexicon-based approaches are not able to handle social media text properly. String similarity, such as edit distance, does not work on non-standard words where the number of edits is large, for example abbreviations. In order to achieve better pre-processing performance, we need to develop methods that are specifically designed for the problem at hand.

Recent work relies on candidate generation and ranking (see section “Related work” for a thorough review), with two major deficiencies: Current approaches have mostly ignored the contextual information present in a sentence that can be potentially very useful. More specifically, in most cases the features extracted or the models developed are *limited to a specific context window*, e.g. Min and Mott (2015) work with character-ngrams, while Jin (2015) relies on features that depend on previous, current and next tokens of a candidate term. This requires additional human effort to decide the appropriate ngram order and design features. More importantly, it restricts the system in a way that prevents longer contextual dependencies to be leveraged (see Figure 1 for an example). The second limitation is that correcting complex normalization mappings are harder to tackle and methods that rely on candidate generator functions by definition limit their approach to specific types of errors. For example, it would be difficult for such methods to handle multiple normalization errors at once, e.g. spelling errors on an acronym or a slang term, and combining candidate generator functions results in a combinatorial problem.

Drawing inspiration from neural machine translation (Bahdanau, Cho, and Bengio 2014; Luong, Pham, and Manning 2015), we propose to address both limitations by using end-to-end neural network models, particularly sequence-to-sequence (Seq2Seq) models. Specifically, Seq2Seq models can encode the entire sequence data with hidden neurons that would naturally capture any useful context information in a sentence to improve text normalization performance. Such models can handle complex normalizations without the need of language-specific tools, given enough training data on any language (i.e. pairs of unnormalized and normalized sentences). We propose and compare several variants of Seq2Seq models for solving the problem of text normalization. The first is a straightforward application of the basic word-level Seq2Seq models to text normalization. However, such an approach faces a major challenge of high percentage of OOV tokens. In contrast, character-level Seq2Seq models do not operate on a limited vocabulary but are much slower to train and recent work has shown that character-based sequence-to-sequence models are not robust on noisy data (Belinkov and Bisk 2017). We propose a novel hybrid end-to-end model that takes into account contextual information as well as addresses the OOV problem in a more robust way. Specifically, our model is based on a recurrent neural encoder-decoder architecture that reads the informal text sequences, transforms them in a continuous-space representation that is passed on the decoder to generate the target normalized sequence. To capture local spelling errors and morphological variations of OOV words, we correct unknown words with a character-level encoder-decoder trained on synthetic *adversarial examples* that capture common errors in online user-generated text. Our method obtains open vocabulary coverage while maintaining the lower training time of word-based models, when compared with character-level sequence-to-sequence architectures.

Our contribution is two-fold:

1. We explore variations of encoder-decoder architectures as well as adversarial training for tackling the task of normalizing social media text.
2. We propose a novel hybrid neural network architecture specifically designed for normalizing OOV tokens. Coupled with adversarial training, our model allows for an open set of corrections while seamlessly incorporates context and long-term dependencies. Through carefully designed experimentation, we show that the proposed hybrid model outperforms both word and character based standard Seq2Seq architectures.

Our source code and model files are publicly available [1].

Related Work

We briefly discuss related work on text normalization. The normalization problem was originally framed as standardizing numbers, dates and acronyms in formal text (Sproat et al. 2001) but the definition was later broaden to transform social media informal text into canonical forms that NLP tools were usually trained on (Sproat et al. 2001). Research

[1] https://github.com/Isminoula/TextNormSeq2Seq

source: got **exo** to share, **u** interested? Concert in **hk** !

target: got **extra** to share, **are you** interested? Concert in **hong kong** !

Figure 1: Example of source (unnormalized) tweet and target (normalized) pair where context helps in correcting ambiguous terms. The word "exo" needs to be transformed to "extra", while the word "concert" provides the required context to understand that "exo" refers to an extra ticket.

on this problem adopts several paradigms, from spell checking (Choudhury et al. 2007), machine translation (Aw et al. 2006; Ling et al. 2013) and speech recognition (Kobus and Yvon 2008).

Early unsupervised methods include probabilistic models (Cook and Stevenson 2009), string edit distance metrics (Contractor, Faruquie, and Subramaniam 2010), construction of normalization dictionaries (Han, Cook, and Baldwin 2012; Gouws, Hovy, and Metzler 2011) or extracting training data from search results with carefully designed queries (Liu et al. 2011). Another line of work is lexicon-based methods and classification models. Han and Baldwin (2011) train a classifier that detects non-standard words and then generate candidates based on morphological and phonemic similarity metrics plus a normalization lexicon. Other approaches include word association graphs (Sonmez and Ozgur 2014), random walks (Hassan and Menezes 2013), statistical (Beaufort et al. 2010; Zhang et al. 2013) and log-linear models (Yang and Eisenstein 2013) and ranking candidates with language models (Han, Cook, and Baldwin 2013). These methods are quite limited as they rely primarily on string and phonetic similarity for identifying lexical variations.

Recently, there is a growing trend on applying Deep Learning in a variety of areas, such as Computer Vision or NLP. Such models offer flexibility, can learn representations and tasks jointly and have produced state-of-the-art for several applications, e.g. object recognition, sentiment analysis or machine translation. The representational power of neural models can potentially allow learning of complicated text transformations, automatically handle language drift and work with heterogeneous large streams of user-generated text. We briefly describe state-of-the-art models and models that either leverage deep learning or contain a component that is trained with neural networks. Chrupała (2014) leverages unlabeled data by incorporating character embeddings as features in a model that learns to perform edit operations. Sridhar (2015) used distributed representations of words trained on a large Twitter dataset to extract normalization lexicons based on contextual similarity. Similarly, Ansari, Zafar, and Karim (2017) leverage word embeddings, as well as string and phonetic similarity to match OOV words to IV words (1:1 mapping), however their method does not take into consideration contextual information and thus cannot properly handle cases where multiple canonical forms of a non-standard word are available. In contrast to this line of work, we do not rely on pretrained embeddings; we learn both word and character representations, in addition to the text normalization model in an end-to-end fashion.

Baldwin et al. (2015) present a text normalization task on English tweets as part of the 2015 ACL-IJCNLP Workshop on Noisy User-generated Text (W-NUT). Two categories were introduced based on whether external resources and public tools were used (unconstrained systems) or not (constrained systems). Deep learning methods and lexicon-augmented conditional random fields (CRFs) achieved the best results, while the performance of unconstrained systems was inferior, suggesting that the underlying model produces a bigger impact on the final performance, compared with the usage of additional data or resources. The models of all participating teams are described in (Baldwin et al. 2015). We should note that no team explored deep sequence-to-sequence models or adversarial training.

Jin (2015) achieved the best performance in the W-NUT task, with a method that generates candidates based on the training data. A binary random forest classifier is trained to predict whether a candidate is the correct canonical form for a token found in a tweet instance. Their feature set used during training includes string similarity, POS and statistics such as support and confidence. The final canonical form selected is the one that achieves the highest confidence score. van der Goot and van Noord (2017) extends this work by leveraging additional external resources of clean and informal text data. Min and Mott (2015) combine a lexicon extracted from the training data with a recurrent neural model that performs edit operations trained on character trigrams. Leeman-Munk, Lester, and Cox (2015) use a neural network classifier that predicts whether a word needs normalization and a secondary model that takes as input a word and outputs its canonical form. Framing the task of text normalization as classification in addition to relying on candidate generator functions limits the types of transformations that can be tackled, as candidate generation relies heavily on human engineering effort and existing methods for creating candidates cannot handle multiple complex normalization errors at once.

The aforementioned methods do not take into account the ***full context*** in which a token appears. It is quite reasonable then for one to wonder whether Seq2Seq models would be appropriate for the task and in which ways the input or architecture should be adjusted in order to reach comparable performance given the limited training data available. To this end, we explore encoder-decoder models and study how crucial context is for the normalization of user-generated text. Finally, we design a novel hybrid Seq2Seq model that is trained on synthetic adversarial examples of noisy social media text.

Sequence-to-Sequence Learning

Encoder-decoder architectures (Sutskever, Vinyals, and Le 2014; Cho et al. 2014) have been applied in a wide variety of natural language tasks, such as machine translation (Wu et al. 2016), dialogue generation (Vinyals and Le 2015), summarization (Nallapati et al. 2016), question answering (Yin et al. 2015). Several extensions of the Seq2Seq models have been proposed with mechanisms such as attention (Bahdanau, Cho, and Bengio 2014), copying (See, Liu, and Manning 2017) and coverage (Tu et al. 2016).

In most cases only the most frequent words are kept, creating a fixed-sized vocabulary, with OOV words mapped to a common UNK token. Consequently, the performance is affected by the limited vocabulary. Recent work propose methods to mitigate this problem, by treating text as a sequence of characters (Lee, Cho, and Hofmann 2017), inventing new word segmentation methods (Sennrich, Haddow, and Birch 2015; Bojanowski et al. 2017) or hybrid word-level models with an additional character-level model to handle problematic cases (Luong and Manning 2016; Ji et al. 2017). While character-based models outperform models based on subword units, their extremely high computational cost and inability to handle long-distance dependencies makes them unappealing in practice. Moreover, as hybrid models only use the secondary character model for problematic cases, such as unknown words, they rely on large training datasets, making them inappropriate for domains with limited annotated data and frequent word variations. Our work lies on the hybrid models category but builds upon the properties of text normalization to adjust the character-based model training.

Text Normalization

Our architecture consists of two encoder-decoder models, primarily a word-based Seq2Seq model, while for transforming words not found in the word-level model's vocabulary, we either backtrack to a secondary character-based Seq2Seq model when its confidence is high or copy the source token (Figure 3). For completeness, we briefly describe encoder-decoder neural models.

Word-level sequence-to-sequence model

Given an unnormalized text represented as an input sequence of words $\vec{x} = [x_1, \ldots, x_T]$ with length T, we consider generating another output sequence of words $\vec{y} = [y_1, \ldots, y_L]$ with length L that has the same meaning as $\vec{x}$. The task is defined as a sequence-to-sequence learning problem which aims to learn the mapping from one sequence to another. Specifically, the architecture is built based on the encoder-decoder framework (Cho et al. 2014; Sutskever, Vinyals, and Le 2014), both of which are parameterized by attention-based recurrent neural networks (RNN).

The encoder module reads the input sequence $\vec{x}$ and transforms it to a corresponding context-specific sequence of hidden states $\vec{h} = [h_1, \ldots, h_T]$. In bi-directional models, two encoders are used; one reading the text in forward mode and another one reading text backwards. The final hidden state at time t is the concatenation of the two encoder modules $\vec{h}_t = [g_f(x_t, h_{t-1}); g_b(x_t, h_{t+1})]$ where g_f and g_b denote the forward and backward encoder units, respectively. Similarly, the decoder defines a sequence of hidden states $\vec{s}_j = g_s(s_{j-1}, y_{j-1}, c_j)$ that is conditioned on the previous word y_{j-1} and decoder state s_{j-1}, as well as the context vector c_j, computed as a weighted sum of encoder hidden states based on the attention mechanism (Bahdanau, Cho, and Bengio 2014):

$$c_j = \sum_{i=k}^{|T|} \alpha_{jk} h_k$$

where $\alpha_{jk} = \text{Softmax}(f(s_{j-1}, h_k))$ and $f(s_{j-1}, h_k) = s_{j-1}^T W h_k$ is the *general* content-based function described in Luong, Pham, and Manning (2015). Then, each target word is predicted by a Softmax classifier $y_j \sim p(y_j | y_{<j}, \vec{x}) = Softmax(\psi(s_j))$, where ψ is an affine transformation function that maps the decoder state to a vocabulary-sized vector.

Given training data $\mathcal{D}$, Seq2Seq model is trained by maximizing the log-likelihood:

$$L(\theta) = -\sum_{(\vec{x},\vec{y}) \in \mathcal{D}} \sum_{j=1}^{|L|} \log p_\theta(y_j | y_{<j}, \vec{x})$$

Note that during training a wrong prediction will cause an accumulation of errors in the subsequent time steps. Thus, when computing the conditional probability $p_\theta(y_j | y_{<j}, \vec{x})$, Scheduled Sampling (Bengio et al. 2015) is often used, a method that alternates between using the model prediction on the previous time step $\hat{y}_{j-1}$ and the target previous word y_{j-1} in order to alleviate the presence of compounding errors.

The word-based Seq2Seq model can capture semantic meaning at a word level and long-term contextual dependencies that help in disambiguation of multiple correction candidates. Figure 1 presents an example of source and target pair of tweets for which context helps in appropriately normalizing the content.

Handling unknown words with a secondary character-based encoder-decoder model

The model operating on words has a limited vocabulary for both source and target. Words that are beyond this vocabulary are represented with a special UNK symbol. For text normalization, where slight variations occur often due to misspellings, keyboard typing errors and intentionally emphasizing terms by elongation of vowels (e.g. "coooool" or "yaaaay"), many of the words are unseen during training, resulting in loss of information. Three possible solutions can be used to tackle this problem: a) copying source words, b) rely on models fully trained on character-based information and c) design hybrid models that work both on word and character level.

A naive strategy would be to just copy the source word when it is outside the scope of the vocabulary (see Figure 2), however many unseen non-standard words will be left intact and thus the coverage of our models will decrease. Another way to handle vocabulary coverage is to pre-process the data and learn a subword representation that allows to generalize to new words. Byte pair encoding (BPE) (Sennrich, Haddow, and Birch 2015) learns the segmentation of text into subwords, e.g. "showed" could be split into "show" and "ed" while "accepting" would be split "accept" and "ing". Such pre-processing is model-agnostic, i.e. can be used irrespectively of the chosen Seq2Seq model. However, BPE relies on

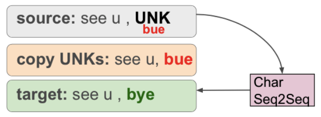

Figure 2: Example of an unseen unnormalized token where copying the source word is insufficient

the cooccurence and order of characters, which in our case is highly noisy.

Character models overcome the bottleneck of restricted vocabularies and do not require any pre-processing or tokenization but are computationally expensive and also suffer from data sparsity. Chung, Cho, and Bengio (2016) provide a detailed analysis regarding the challenges of character-level models. Belinkov and Bisk (2017) recently showed that character-based models fail to translate noisy text that humans can handle easily. They mention that such models are rarely trained to explicitly handle typos and noise, commonly found in natural language.

Hybrid models on the other hand, rely primarily on a word-based representation where the meaning is naturally preserved and backtrack to a secondary character-level model to deal with problematic text. Because the character model is trained only in some cases, it requires a large pool of such problematic aligned text. Due to limited training data available for text normalization and the long tail of rare non-standard words, hybrid architectures that train character-level models only for words outside the vocabulary would be insufficient. Thus, for in-vocabulary words we rely solely on the word-level model, while when a word is OOV, we backtrack to a character encoder-decoder that is trained on word pairs rather than longer token sequences, i.e. each pair of source and target words in our training set is processed separately.

Adversarial training for increased robustness to noisy user-generated text

To improve our model's robustness to noisy text, we incorporate an adversarial training procedure to our character-based secondary model. We augment our data by creating synthetic adversarial examples of words, i.e. unnormalized and canonical forms. More specifically, for all source-target pairs of tweets, we keep words that remain unchanged. This process creates our source and target vocabularies for the character model. We later on inject multiple types of noise during training, by editing the source part of each word. More specifically, we introduce 6 types of errors that are typically found in user-generated text, by randomly:

del: Deleting a character from a word

swap: Swapping the placement of two characters

lastchar: Elongating the last character k times when the word ends with $\{u, y, s, r, a, o, i\}$, where $k \in \{1, \ldots, 6\}$

punct: Deleting e.g. "I'm" $\rightarrow$ "Im" or misplacing apostrophes, e.g. "don't" $\rightarrow$ "do'nt"

keyboard: Replacing characters based on their distance on the keyboard, e.g. "hello" $\rightarrow$ "jello"

Dataset	Tweets	Tokens	Noisy	1:1	1:N	N:1	Our Vocab
train	2950	44385	3942	2,875	1,043	10	10,084
test	1967	29421	2776	2,024	704	10	7,389

Table 1: LexNorm statistics from (Baldwin et al. 2015) and vocabulary statistics after preprocessing

elong: Extending vowel usage k times, where k is a random number

Experiments

In this section we present our experimental setup for assessing the performance of the text normalization model described above. We want to test: a) whether a naive replacement of words with their non-standard form would be sufficient for the text normalization task, b) which Seq2Se2 model is the most effective, c) whether BPE and character level models are appropriate for normalizing OOV social media tokens, d) how crucial is context and long-term dependencies for correctly normalizing noisy text and e) whether adversarial training improves robustness of our hybrid architecture.

Dataset

We use the LexNorm dataset from the 2015 ACL-IJCNLP Workshop on Noisy User-generated Text (W-NUT) (Baldwin et al. 2015). The dataset contains 4,917 tweets with 373 unique non-standard word types, split into 60:40 training/testing ratio. There are 488 non-standard word types that are unseen during training, i.e. not found in the training data. Table 1 lists some statistics of the dataset described in (Baldwin et al. 2015). Note that apart from mapping a source word to a target word ($1:1$ mapping), there are also words that are mapped to more than one target tokens ($1:N$ mapping), e.g. "omw" $\rightarrow$ "on my way".

To reduce vocabulary size, words are lowercased, while mentions were tagged and anonymized with a $\langle mention \rangle$ token. The same anonymization was applied for URLs $\langle url \rangle$ and hashtags $\langle hash \rangle$. At test time, we de-anonymize by looking them up in their source sentences. Additionally, we keep a common vocabulary between source and target text. Each sequence is additionally pre-processed by adding a start $\langle s \rangle$ and end $\langle \backslash s \rangle$ symbol.

Baseline models

We compare our model (**HS2S**) with a diverse set of baselines, including two naive dictionary-based approaches: we begin by constructing a lexicon from the training data and correcting only unique mappings (**Dict1**) or additionally choose randomly when multiple canonical forms are available (**Dict2**). We also compare with a two-staged strategy that first corrects unique mappings based on the dictionary and secondly utilizes a word-level Seq2Seq model trained to correct only multiple mappings (**S2SMult**).

Addditionaly, we include a default attention-based word-level encoder-decoder (**S2S**) as our baseline for comparison. For this model, OOV words are solely copied directly from the source sequence, thus "unseen normalizations" are not

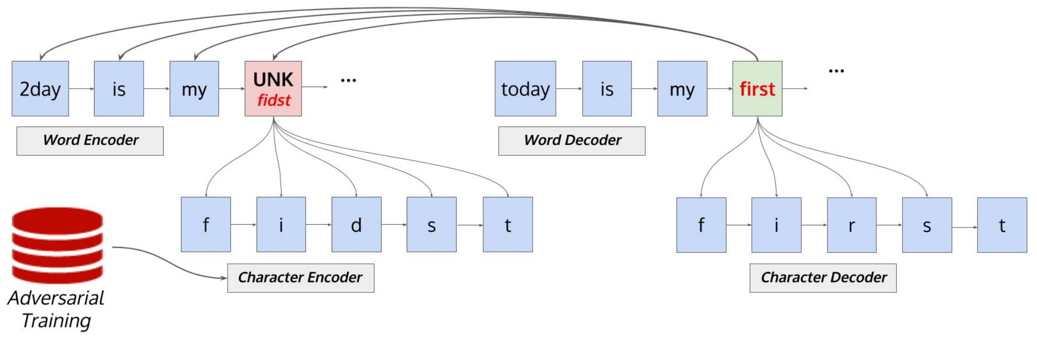

Figure 3: Our Hybrid Sequence-to-Sequence (HS2S) architecture that consists of two nested encoder-decoder architectures, one trained on word-level information and a character-based trained on synthetically generated adversarial examples. The primary model (word encoder-decoder) is trained on sequences of words. When an unknown symbol is encountered, such as token ”fidst” (red box) in our example, we leverage a secondary character-level Seq2Seq model (character encoder-decoder) that is trained on a large pool of synthetic adversarial training examples of words to correctly normalize, e.g. token ”first” (green box)

handled. This baseline should indicate whether targeting unseen normalizations is critical for the model performance. Since character or sub-word level representations can alleviate the problem of limited vocabularies, we also experiment with a character-based model (**S2SChar**) and a model that is trained on subwords with BPE tokenization[2] (**S2SBPE**).

Finally, we include a model trained on target sequences preprocessed with a special symbol to indicate that the word should be left intact (**S2SSelf**), e.g. if the source sequence is “see u soon” and the target sequence is “see you soon”, we replace the target sequence with “@self you @self”. During prediction, when we generate the normalized target of a source sentence, we replace this special symbol by copying from the source. Ultimately, we seek to find which sequence-to-sequence model is the most effective for the text normalization task.

Training Details

We keep a shared vocabulary between source and target and also tie the decoder embeddings (Press and Wolf 2017). We optimized all models with Adam (Kingma and Ba 2014) and the gradient is rescaled when the norm exceeds {5,10} (Pascanu, Mikolov, and Bengio 2013). Batch size is set to {32,500}. All of our models are bi-directional and use attention. To compare performance, we tune each model separately with random search. The best hyper-parameters are summarized in Table 2. To tune the hyper-parameters we used a 10% random split of the training data and performed random search on the hyper-parameter space. Once the best combination was found, we retrained our system using the full training data set.

Our adversarial training procedure is guided by an additional hyper-parameter, *noise ratio*, that tunes the number of adversarial instances used. Our best performing model has a noise ratio of 0.1, i.e., more than 10% of instances used are generated with adversarial training (see Table 6). In general, we observed that training on large amounts of adversarial instances, e.g. 50% additional instances, results in decreased performance, e.g. when noise ratio is increased to 0.5, the F1 score is decreased to 82.67% (Table 6: incorporating large quantities of adversarial instances decreases performance). Furthermore, to reduce the amount of false positives we allow the character-level secondary model to correct only words for confident predictions. If the confidence of the character-level model is low, our architecture copies from the source.

Hyper-parameter	Word-level	secondary Char-level	S2SChar	S2SBPE
emb.dimension	100	256	256	100
neurons/layer	200	500	512	300
layers	3	3	3	2
dropout	0.5	0.5	0.2	0.3
learning rate	0.01	0.001	0.001	0.01

Table 2: Best performing hyper-parameter settings of our proposed text normalization models

Comparison of Sequence to Sequence models

We compare our novel hybrid model with word-level baseline Seq2Seq models (Table 3), as well as character-based and BPE-tokenized Seq2Seq models (Table 4). In Table 3 we see that our dictionary based baselines result in lower performance. **Dict2** shows that handling ambiguous cases

Model name	**Precision**	**Recall**	**F1**	**Method highlights**
HS2S	90.66	78.14	83.94	Hybrid word-char Seq2Seq
S2S	93.39	75.75	83.65	Word-level Seq2Seq
Dict1	96.00	52.20	67.62	Dictionary (unique mappings)
Dict2	56.27	63.57	59.70	Dict1 + Random
S2SMulti	93.33	75.57	83.52	Dict1 + S2S
S2SSelf	82.74	65.50	73.11	@Self for tokens that need no normalization

Table 3: Comparison of our S2S models with word-level baselines.

[2]We experimented with the original subword implementation (https://github.com/rsennrich/subword-nmt) as well as a pre-trained version (Heinzerling and Strube 2018) (https://github.com/bheinzerling/bpemb) that produced better results.

Model name	Precision	Recall	F1	Method highlights
HS2S	90.66	78.14	83.94	Hybrid word-char Seq2Seq
S2SChar	67.14	70.50	68.78	Character-level Seq2Seq
S2SBPE	20.00	52.04	28.90	Word Seq2Seq + BPE

Table 4: Comparison of our HS2S model with additional baselines.

inappropriately results in a dramatic drop in precision. It is therefore necessary for a text normalization model to be able to correctly normalize text in the occurrence of multiple mappings. **S2SMulti** is a baseline method that firstly normalizes terms that have a unique mapping, based on source and target tokens found in the training data, and later on utilizes a word-level Seq2Seq model that is trained to correct one-to-many mappings (an unnormalized token that can be transformed into several standard words, e.g. "ur" $\rightarrow$ {"you are", "your"}). We can see that this method has a better performance, which validates our hypothesis that a naive word replacement would not suffice, however the end-to-end Seq2Seq model **(S2S)** performs better than **S2SMulti**, i.e. Seq2Seq models can handle both unique and multiple mappings seamlessly without any additional feature engineering effort apart from tokenization.

Our character-level **S2SChar** model's performance is slightly above the dictionary baseline **Dict1** which suggests that characters do not contain enough semantics to appropriately disambiguate between terms. Our results align with relevant literature (Belinkov and Bisk 2017) that emphasizes on the noise sensitivity of character-based Seq2Seq models. We should also note that character-level Seq2Seq models take longer to train. Our best performing word Seq2Seq model took 22 minutes to train while our top scored character model took 3 hours, for the same number of epochs. The secondary character model of our hybrid architecture, which is trained on pairs of words, took 42 minutes for the same number of epochs.

Despite extensively tuning the hyper-parameters and experimenting with two subword tokenization tools, we were unable to train a good performing model on subword units (**S2SBPE**) successfully. S2SBPE has surprisingly very low performance, the poorest of all models. As BPE relies on co-occurrence of characters to extract frequent subword patterns, one would expect that it would not be able to capture useful information due to the high percentage of noise. This emphasizes the importance of developing models robust to informal text, that can learn from noisy input "on the fly".

Error analysis

We perform an extensive error analysis. First, we check the model output, particularly in which cases our model fails. Table 5 presents the most frequent normalizations that our model performed correctly and the most frequent cases that were missed, as well as the frequency of the source terms. Note that we also keep track of how many times a term remains unchanged, specifically for cases where that term has multiple mappings available (Table 5, information found in parentheses). We can see that our model can handle multiple mappings when those are adequately represented in the training data. Most of the incorrect normalizations appear less than 50 times in our data (very infrequent) or are ambiguously normalized in some examples and left intact in other examples, e.g. "rt" and "2" remain unchanged a few times, and these are the cases that the model missed. These types of errors can be handled by adding more training data. Furthermore, we should note that our model can also be adjusted to work with distantly supervised data or with ensemble methods.

Comparing with different representations that can handle unknown words, our character level model performs edits that might result in removing words from the text or are close to the gold-truth canonical form but not entirely correct (see Table 7). In contrast to our hybrid model that can preserve contextual and word-level semantic information, our BPE model's poor performance results in editing text that needs no correction.

Futhermore, we perform error analysis on the secondary character-level model that is trained on synthetic adversarial examples of word pairs. In total, our model correctly normalizes 16.22% of unseen source words. In Table 8 we present OOV terms that the model can correctly normalize. Most frequent errors that are correctly normalized are elongating the last characters, deleting last character $[g]$ in gerund, swapping or replacing characters and typos. There are also several typos that the model was not able to correct, such as missing whitespaces or editing words unnecessarily.

How contextual information affects performance One of our main questions is how crucial contextual information is for text normalization. We test the importance of context by performing two experiments. For our first experiment, we perform the following preprocessing of the training data: we constrain the target-side of each example by replacing each word that remains unchanged (no normalization needed) with a special @*self* symbol. With this representation all tokens that need normalization are preserved and the model would learn which words remain the same. We call this model **S2SSelf**. We notice that this representation lowers performance due to loss of contextual information on the target side (see Table 3). In Table 9 we present examples of our hybrid model's predictions and compare with the predictions of S2SSelf. In most cases we observe that our model relies on context to normalize short tokens, while S2SSelf fails to correct such terms.

Moreover, we create ngram representations of our data by splitting the tweets: for example a unigram model is trained on word pairs solely and ignores context when normalizing, as it edits each word separately and similarly a bigram model is trained on phrases that contain two words. We continue with higher-order ngrams, train Seq2Seq models on such ngram-based split of text and analyze the importance of contextual information. By gradually varying the context window, while keeping the rest of the hyper-parameters stable, we can analyze how it affects the performance. In Figure 4 we can see that recall remains fairly unchanged, while precision increases as the context window grows larger, i.e. train on higher-order ngrams and thus incorporating more

Source	Target	Count	Source
u	{you're, you, u, your}	234	335 (2)
lol	laughing out loud	197	272
im	{i, i'm}	153	182
dont	don't	57	92
lmao	laughing my @ss off	45	45
n	{and, in, at, n}	40	57 (8)
omg	oh my god	34	34

Source	Target	Count	Source
2	{to, 2}	9	36 (25)
ya	{ya, you, your, yourself}	9	15 (4)
y	{y, why}	9	17 (8)
yo	{you, your, yo}	7	12 (1)
rt	{rt, retweet}	7	602 (582)
b	{b, be, because, by}	4	20 (9)
nah	{no, nah, not, now}	4	6 (2)

Table 5: Most frequent correct (*left table*) and incorrect (*right table*) normalizations of our word-level Seq2Seq model. We present how many times a source tweet was (in)correctly normalized (Count column) as well as how many times that term appears in the source-side of the examples (Source). For cases where a token can be normalized to itself, we include how many times that term appears unchanged (information in parentheses)

Noise Ratio	**Total examples**	**Noise-injected examples**	**Precision**	**Recall**	**F1**
0.1	34,875	5,739	90.66	78.14	83.94
0.2	37,659	8,523	89.92	78.25	83.68
0.3	40,489	11,353	88.61	78.11	83.03
0.4	43,191	14,055	89.25	78.25	83.39
0.5	45,921	16,155	87.33	78.47	82.67
0.6	48,625	19,489	86.21	79.05	82.47
0.7	51,380	22,244	84.89	78.83	81.75
0.8	54,034	24,898	84.37	79.05	81.62
0.9	56,829	27,693	83.94	78.98	81.38

Table 6: Varying the amount of adversarial examples

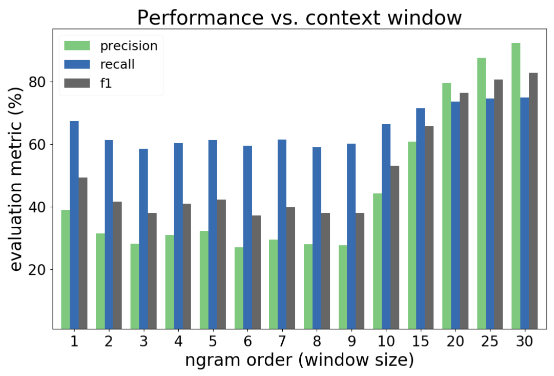

Figure 4: Varying the ngram-wise split of sequences to check how context affects performance of text normalization.

Source:	cmon familia dont mess this up please
Target:	come on familia don't mess this up please
HS2S/S2Multi:	**come** on familia don't mess this up please
S2SSelf:	**cmon** familia don't mess this up please
S2Char:	**comon** familia don't mess this up please
S2SBPE:	**cmon** familia don't just **ess** this up
Source	... i'm not gon diss you on the internet cause ...
Target:	... i'm not gonna disrespect you on the internet because ...
HS2S/S2Multi:	... i'm not gonna **disrespect** you on the internet because ...
S2Char:	... i'm not gonna **thiss** you on the internet because ...
S2SBPE:	... i'm not gonna **be** you on the internet because ...

Table 7: Examples where our model surpasses architectures that rely on lower-level representation of text.

context. Overall, we can observe that F1 measure gets better with additional contextual information.

Comparison with related work

Finally, we present our comparison with related work on Table 10. We see that all previous Deep Learning approaches are close to 82% F1 score. Due to the nature of our hybrid model, we were able to achieve the best performance so far among neural models in related work. In general, we observe comparable performance with state-of-the-art methods that are constrained on utilizing additional resources[3]. We compare the incorrect normalizations that our Seq2Seq model and MoNoise - the best performing method - produce. Both systems appear to have similar results in terms of most frequent incorrect normalizations (Table 13). In many cases our hybrid Seq2Seq model leaves intact terms that are ambiguous in terms of whether they should be normalized or not, while (van der Goot and van Noord 2017) normalize words more often (Table 14), in some cases incorrectly.

Interestingly, there are examples that, despite the lack of correct target annotation, our model normalizes tokens correctly (see Table 12). More specifically, there are several tokens that were not normalized but in an ideal scenario should be, e.g. many punctuation errors that were not normalized or abbreviated tokens that were not converted into their standard form. As a result, despite correctly transforming text, our system gets penalized for such cases.

Conclusions and future work

Text normalization is an important preprocessing part that helps users understand online content and increases performance of off-the-shelve pretrained NLP tools. However, due to the inherent constraints of existing feature engineering methods used, existing work cannot capture longer contextual information and is limited to handling specific types of normalization corrections. Neural Seq2Seq models can naturally correct complex normalization errors by learning edits on large pools of text data. Additionally, improving robustness of Seq2Seq models on real-word noisy text data is a crucial problem that remains fairly unexplored. To this end we have introduced a novel hybrid neural model for social

[3]MoNoise (van der Goot and van Noord 2017) leverages large collections of Twitter and Wikipedia data.

Source	Prediction	Confidence (%)
perfomance	**performance**	83.90
considerin	**considering**	78.94
birthdayyyyy	**birthday**	74.30
brothas	**brother**	72.53
pepole	**people**	72.02
tomorroww	**tomorrow**	69.03
yesssssss	**yes**	68.35
iight	**alright**	59.35

Source	Prediction	Confidence (%)
pantsareneveranoption	**pantsarentien**	83.29
judgemental	**judgmmental**	79.11
kissimmee	**kissimme**	74.08
bsidez	**baidez**	67.67
coldplay	**coldolay**	59.43
knob	**know**	58.12
donuts	**doughs**	57.78
becos	**becouse**	55.72

Table 8: OOV words that our secondary character model has normalized correctly (blue) or incorrectly (red)

Source:	think tht took everything off ma mind for tha night
Target:	think that took everything off my mind for the night
HS2S: (80%)	think **that** took everything off **ma** mind for **the** night
S2SSelf: (50%)	think **that** took everything off **ma** mind for **the tha** night
Source:	death penalty would b d verdict @general_marley murder will b d case ...
Target:	death penalty would be the verdict @general_marley murder will be the case ...
HS2S: (88.8%)	death penalty would **be the** verdict @general_marley murder will **b the** case ...
S2SSelf: (0%)	death penalty would **b d** verdict @general_marley murder will **b d** case ...

Table 9: Comparing HS2S with S2SSelf shows context is crucial for correct normalization, especially for short tokens.

Model	Precision	Recall	F1
Hybrid Seq2Seq (HS2S)	90.66	78.14	83.94
Random Forest (Jin 2015)	90.61	78.65	84.21
Lexicon +LSTM (Min and Mott 2015)	91.36	73.98	81.75
ANN (Leeman-Munk, Lester, and Cox 2015)	90.12	74.37	81.49
MoNoise* (van der Goot and van Noord 2017)	93.53	80.26	86.39

Table 10: Comparison of our hybrid Seq2Seq model with related work on Text Normalization. *In contrast with the rest of the presented related work, Monoise leverages additional textual resources.

Source:	@ifumi0819 i see , u can comeee
Target:	@ifumi0819 i see , you can come
HS2S:	@ifumi0819 i see , you can **comeee**
Our_RF:	@ifumi0819 i see , you can **comes**
Source:	startin to get into this type of musik @vinnyvitale
Target:	starting to get into this type of music @vinnyvitale
HS2S:	**starting** to get into this type of **musik** @vinnyvitale
Our_RF:	**startin** to get into this type of **musik** @vinnyvitale
Source:	#youarebeautiful allly this hashtag should be for you im ugly
Target:	#youarebeautiful allly this hashtag should be for you i'm ugly
HS2S:	#youarebeautiful allly this hashtag should be for you **i'm** ugly
Our_RF:	#youarebeautiful **alli** this hashtag should be for you **i'm** ugly

Table 11: Examples of corrent and incorrect normalizations of our model (HS2S) and Jin (2015) (our implementation)

Source:	rt @foxtramedia tony parker sportscenter convo ...
Target:	rt @foxtramedia tony parker sportscenter **convo** ...
Prediction:	rt @foxtramedia : tony parker sportscenter **conversation** ...
Source:	... but looking back defo do now, haha !
Target:	... but looking back **defo** do now, haha !
Prediction:	... but looking back **definitely** do now, haha !

Table 12: Examples where our model performs correct normalization but during the annotation process, some tokens remained unnormalized in the target sequence, which results in lower performance in evaluation.

Source	Target	HS2S (ours)	MoNoise
youd	would	youd	you'd
ya	your	you	ya
yknow	you know	yknow	know
werent	weren't	werent	were
tiz	this	tizket	tiz
swthat	shout out	switht	swthat
shite	shitty	shit	shite
rts	retweets	rts	rts
pleeze	please	pleaze	pleeze
nah	now	no	nah
judgemental	judgmental	judgmmental	judgemental
championssssss	champions	championsssss	championssssss

Table 13: Examples that both our hybrid (HS2S) model and MoNoise (van der Goot and van Noord 2017) incorrectly normalized.

media text normalization that utilizes a word-based encoder-decoder architecture for IV tokens and a character-level sequence-to-sequence model to handle problematic OOV cases. Our character-based component is trained on adversarial examples of word pairs. Experimental results show that our hybrid architecture improves robustness to noisy user-generated text and shows superior performance, when compared with open vocabulary models. Without relying on any external sources of additional data, we built a system that improves the performance of neural models of text normalization and produces results comparable with other models found in the recent related literature. Our system can be deployed as a preprocessor for various NLP applications and off-the-shelve tools to improve their performance on social media text.

We plan to apply the approach to more languages and compare our adversarial training to other methods, e.g. perturbations applied directly to the embedding space instead of the input. While normalizing informal text, it is worth to consider whether the meaning of a noisy version remains the same, for example the extended usage of vowels ("yaaaaaaay") indicates emphasis while capitalization represents raising the tone. We leave the analysis of the trade-off between retaining such information and normalizing noisy text as future work.

	Source	Target	Prediction
HS2S (ours)	2night	tonight	2night
	aboul	about	aboul
	asap	as soon as possible	asap
	bermudez	bermudez	bermudes
	bfor	before	boor
	cruz	cruz	crus
	outta	outta	out of
	pppp	people	pppp
MoNoise	wildin	wilding	wildin
	weeknd	weekend	weeknd
	tix	tickets	ticket
	da	the	da
	rip	rest in peace	rip
	probs	problems	probably
	gf	girlfriend	gf
	broo	brother	bro

Table 14: Examples of incorrect normalizations that are distinct between our hybrid (HS2S) model and and MoNoise (van der Goot and van Noord 2017).

Acknowledgments. The authors would like to thank the anonymous reviewers for their helpful comments. This material is based upon work supported by the National Science Foundation under Grant No. 1801652.

References

Ansari, S. A.; Zafar, U.; and Karim, A. 2017. Improving text normalization by optimizing nearest neighbor matching. *arXiv preprint arXiv1712.09518*.

Aw, A.; Zhang, M.; Xiao, J.; and Su, J. 2006. A phrase-based statistical model for sms text normalization. In *Proceedings of the COLING/ACL on Main conference poster sessions*. Association for Computational Linguistics.

Bahdanau, D.; Bosc, T.; Jastrzebski, S.; Grefenstette, E.; Vincent, P.; and Bengio, Y. 2017. Learning to compute word embeddings on the fly. *arXiv preprint arXiv:1706.00286*.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Baldwin, T.; de Marneffe, M.-C.; Han, B.; Kim, Y.-B.; Ritter, A.; and Xu, W. 2015. Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. In *Proceedings of the Workshop on Noisy User-generated Text*.

Beaufort, R.; Roekhaut, S.; Cougnon, L.-A.; and Fairon, C. 2010. A hybrid rule/model-based finite-state framework for normalizing sms messages. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Belinkov, Y., and Bisk, Y. 2017. Synthetic and natural noise both break neural machine translation. *arXiv preprint arXiv:1711.02173*.

Bengio, S.; Vinyals, O.; Jaitly, N.; and Shazeer, N. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*.

Bojanowski, P.; Grave, E.; Joulin, A.; and Mikolov, T. 2017. Enriching word vectors with subword information. *Transactions of the Association of Computational Linguistics*.

Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Choudhury, M.; Saraf, R.; Jain, V.; Mukherjee, A.; Sarkar, S.; and Basu, A. 2007. Investigation and modeling of the structure of texting language. *International Journal of Document Analysis and Recognition (IJDAR)*.

Chrupała, G. 2014. Normalizing tweets with edit scripts and recurrent neural embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Chung, J.; Cho, K.; and Bengio, Y. 2016. A character-level decoder without explicit segmentation for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Contractor, D.; Faruquie, T. A.; and Subramaniam, L. V. 2010. Unsupervised cleansing of noisy text. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*. Association for Computational Linguistics.

Cook, P., and Stevenson, S. 2009. An unsupervised model for text message normalization. In *Proceedings of the workshop on computational approaches to linguistic creativity*. Association for Computational Linguistics.

Gouws, S.; Hovy, D.; and Metzler, D. 2011. Unsupervised mining of lexical variants from noisy text. In *Proceedings of the First workshop on Unsupervised Learning in NLP*. Association for Computational Linguistics.

Han, B., and Baldwin, T. 2011. Lexical normalisation of short text messages: Makn sens a# twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics.

Han, B.; Cook, P.; and Baldwin, T. 2012. Automatically constructing a normalisation dictionary for microblogs. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*. Association for Computational Linguistics.

Han, B.; Cook, P.; and Baldwin, T. 2013. Lexical normalization for social media text. *ACM Transactions on Intelligent Systems and Technology (TIST)*.

Hassan, H., and Menezes, A. 2013. Social text normalization using contextual graph random walks. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Heinzerling, B., and Strube, M. 2018. BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages. In chair), N. C. C.; Choukri, K.; Cieri, C.; Declerck, T.; Goggi, S.; Hasida, K.; Isahara, H.; Maegaard, B.; Mariani, J.; Mazo, H.; Moreno, A.; Odijk, J.; Piperidis, S.;

and Tokunaga, T., eds., *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Miyazaki, Japan: European Language Resources Association (ELRA).

Ji, J.; Wang, Q.; Toutanova, K.; Gong, Y.; Truong, S.; and Gao, J. 2017. A nested attention neural hybrid model for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1.

Jin, N. 2015. Ncsu-sas-ning: Candidate generation and feature engineering for supervised lexical normalization. In *Proceedings of the Workshop on Noisy User-generated Text*.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kobus, C., and Yvon, F. 2008. Normalizing sms: are two metaphors better than one? In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*. Association for Computational Linguistics.

Lee, J.; Cho, K.; and Hofmann, T. 2017. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association of Computational Linguistics*.

Leeman-Munk, S.; Lester, J.; and Cox, J. 2015. Ncsu_sas_sam: deep encoding and reconstruction for normalization of noisy text. In *Proceedings of the Workshop on Noisy User-generated Text*.

Ling, W.; Dyer, C.; Black, A. W.; and Trancoso, I. 2013. Paraphrasing 4 microblog normalization. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Liu, F.; Weng, F.; Wang, B.; and Liu, Y. 2011. Insertion, deletion, or substitution?: normalizing text messages without pre-categorization nor supervision. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*. Association for Computational Linguistics.

Luong, M.-T., and Manning, C. D. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Luong, M.-T.; Pham, H.; and Manning, C. D. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Min, W., and Mott, B. 2015. Ncsu_sas_wookhee: a deep contextual long-short term memory model for text normalization. In *Proceedings of the Workshop on Noisy User-generated Text*.

Nallapati, R.; Zhou, B.; dos Santos, C.; Gulcehre, C.; and Xiang, B. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*.

Pascanu, R.; Mikolov, T.; and Bengio, Y. 2013. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*.

Press, O., and Wolf, L. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*.

See, A.; Liu, P. J.; and Manning, C. D. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Sennrich, R.; Haddow, B.; and Birch, A. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Sonmez, C., and Ozgur, A. 2014. A graph-based approach for contextual text normalization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Sproat, R.; Black, A. W.; Chen, S.; Kumar, S.; Ostendorf, M.; and Richards, C. 2001. Normalization of non-standard words. *Computer speech & language*.

Sridhar, V. K. R. 2015. Unsupervised text normalization using distributed representations of words and phrases. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*.

Tu, Z.; Lu, Z.; Liu, Y.; Liu, X.; and Li, H. 2016. Modeling coverage for neural machine translation. *arXiv preprint arXiv:1601.04811*.

van der Goot, R., and van Noord, G. 2017. Monoise: modeling noise using a modular normalization system. *arXiv preprint arXiv:1710.03476*.

Vinyals, O., and Le, Q. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Wu, Y.; Schuster, M.; Chen, Z.; Le, Q. V.; Norouzi, M.; Macherey, W.; Krikun, M.; Cao, Y.; Gao, Q.; Macherey, K.; et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Yang, Y., and Eisenstein, J. 2013. A log-linear model for unsupervised text normalization. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Yin, J.; Jiang, X.; Lu, Z.; Shang, L.; Li, H.; and Li, X. 2015. Neural generative question answering. *arXiv preprint arXiv:1512.01337*.

Zhang, C.; Baldwin, T.; Ho, H.; Kimelfeld, B.; and Li, Y. 2013. Adaptive parser-centric text normalization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.