# Adapting Seq2Seq models for Text Normalization in Social Media

Ismini Lourentzou, Kabir Manghnani, ChengXiang Zhai

# Social Media Data

An abundant source of valuable raw data

**Text today is user-generated and online**

- Online blogs and posts
- Forums
- Customer reviews …

Primary input for algorithms that:

- Understand user intent/preferences
- Predict trends
- Recommender systems
- Targeted advertising


https://www.quora.com/Can-I-make-money-from-a-social-media-management-tool-for-Twitter-despite-a-crowded-market

# Difficulties with noisy text

Text in social media: spelling errors, non-standard words, and acronyms.

- Problems in understanding the expressed content

- NLP tools struggle with noisy informal language


*Source: Twitter*

# Text Normalization

Identifies noisy parts of the text and substitutes with canonical forms

| | | |
|---|---|---|
| 1. Misspellings | defenitely → definitely |
| 2. Phonetic substitution | 2morrow → tomorrow |
| 3. Shortening of words | convo → conversation |
| 4. Slang | low key |
| 5. Capitalization | YEAH |
| 6. Vowel elongation | coooooool |
| 7. Punctuation | doesnt → doesn’t |
| 8. Acronyms | idk → i don’t know |
| standard words | goat → greatest of all time |

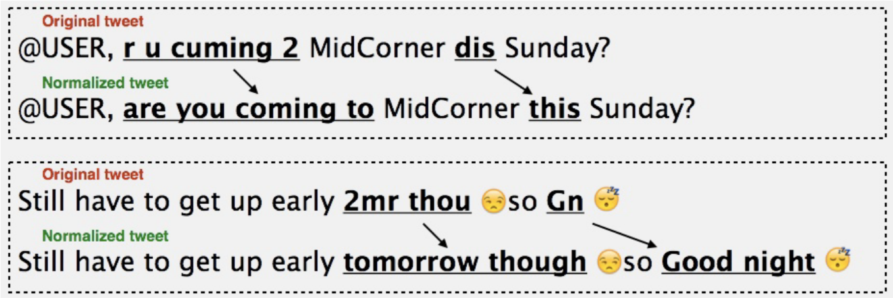



Figure from [1]

Mapping OOV word to IV canonical form
Preserve meaning of sentence

# Related Work

**NN predicts whether a word needs normalization*

**EARLY WORK**

Spell checking
Machine Translation
Speech Recognition

**UNSUPERVISED**

String edit distance metrics
Phonetic similarity
Lexicon-based methods

**SUPERVISED**

**Classification + Candidate generation [2,3]**
Ranking candidates with LM

**ADVANCED**

Word association graphs
Probabilistic Models (e.g. CRFs)

**DEEP LEARNING**

Neural norm/tion detector* **[5]**
Character/Word embeddings
RNNs on character tri-grams **[4]**

# Limitations of related work

- Framing the task as classification + candidate generation limits types of transformations that can be tackled
- Working on local fashion (string or phonetic similarity)
- Not incorporating the full context in which a token appears

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

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

# Dataset

| Dataset | Tweets | Tokens | Noisy | 1:1 | 1:N | N:1 | Our vocab |
|---|---|---|---|---|---|---|---|
| train | 2950 | 44385 | 3942 | 2875 | 1043 | 10 | 10084 |
| test | 1967 | 29421 | 2776 | 2024 | 704 | 10 | 7389 |

**LexNorm 2015** [1]
ACL-IJCNLP 2015 Workshop on Noisy User-generated Text (WNUT)

All words lowercased
mentions → ⟨mention⟩
URLs → ⟨url⟩
Hashtags → ⟨hash⟩

**Source**: 2day is my fidst day in Munich …
**Target**: today is my first day in Munich …

# TextNorm Seq2Seq


**Source**: 2day is my fidst day in Munich …
**Target**: today is my first day in Munich …

Frequent
- misspellings
- keyboard typing errors
- intentional changes

**High OOV rates!**

1. Is **contextual information** is crucial for this task?
2. Would **Seq2Seq models** be appropriate for the task?
3. How should the **input** or **architecture** be **adjusted**?
4. Given very little amounts of training data, can we get **SOTA performance**?

# TextNorm Seq2Seq


Frequent

- misspellings
- keyboard typing errors
- intentional changes

**High OOV rates!**

- Copying source words

**Source**: 2day is my fidst …

**Target**: today is my first …

**Copy UNKs**: today is my fidst …

# TextNorm Seq2Seq


Frequent
- misspellings
- keyboard typing errors
- intentional changes

**High OOV rates!**

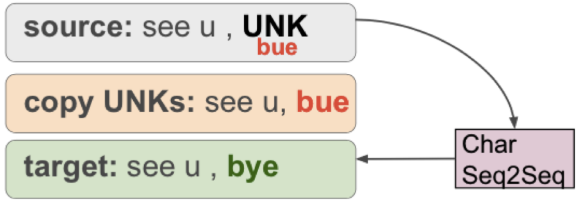
- Character-based models

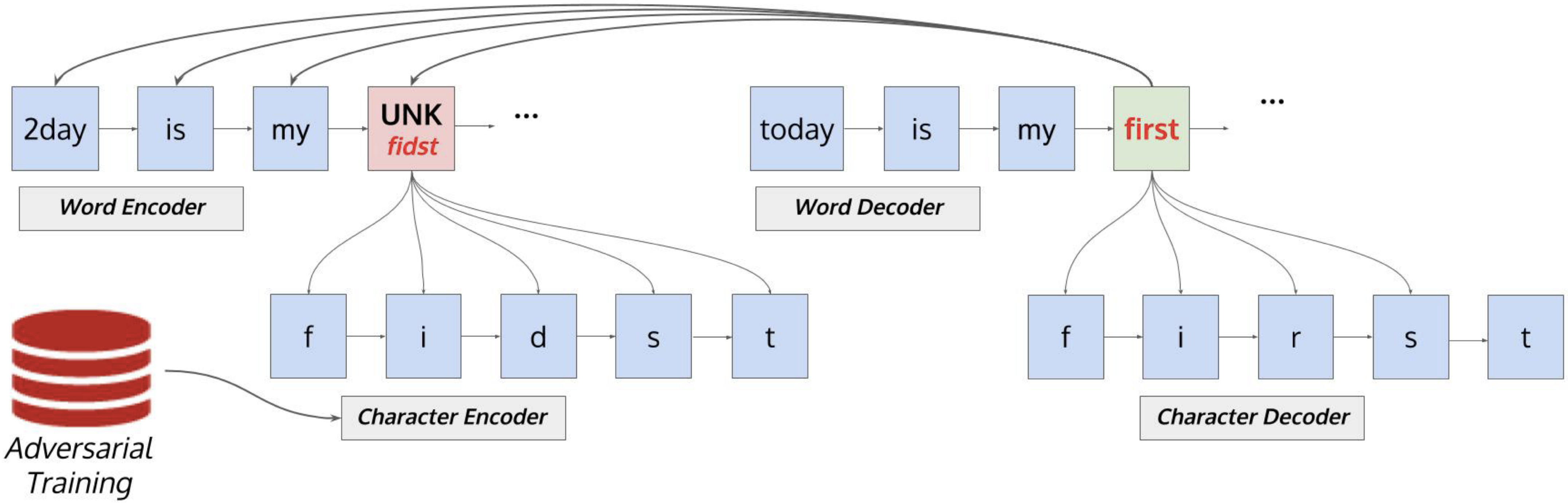
  **t o d a y  i s  m y  f i r s t**

- Subword representations, e.g. BPE

  **to day is my fir st**

# TextNorm Seq2Seq




**Hybrid** Seq2Seq model

Trained on synthetic adversarial examples of noisy social media text

# Types of noise

Introduce 6 types of errors typically found in user-generated text

| | |
|---|---|
| **del:** | Deleting a character from a word |
| **swap:** | Swapping the placement of two characters |
| **lastchar:** | Elongating last character when word ends with {u, y, s, r, a, o, i} |
| **punct:** | Deleting or misplacing apostrophes |
| **keyboard:** | Replacing characters based on keyboard distance, e.g. hello→ jello |
| **elong:** | Extending vowel usage |

# Baselines & Seq2Seq variations

| | |
|---|---|
| **HS2S** | Hybrid word-char Seq2Seq |
| **S2S** | Standard word-level Seq2Seq + Copy OOV words from SRC |
| **Dict1** | Dictionary for unique mappings **2day → today** |
| **Dict2** | Dictionary + ***random*** for non-unique mappings **ur → {your, you are}** |
| **S2SMult** | Dictionary + **S2S** for non-unique mappings |
| **S2SChar** | Character-level Seq2Seq |
| **S2SBPE** | Seq2Seq on subword units (BPE encoding) |
| **S2SSelf** | Special symbol @self for tokens that need no normalization<br>SRC: “see u soon” → “@self you @self”<br>TGT: “see you soon” |

+ SOTA from related work [2,3,4,5]

# Experimental Results

| Model name | Precision | Recall | F1 | Method highlights |
|---|---|---|---|---|
| S2SChar | 67.14 | 70.50 | 68.78 | Character-level Seq2Seq |
| S2SBPE | 20.00 | 52.04 | 28.90 | Word Seq2Seq + BPE |
| 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 |
| HS2S | 90.66 | 78.14 | 83.94 | Hybrid word-char Seq2Seq |
| S2S | 93.39 | 75.75 | 83.65 | Word-level Seq2Seq |

Comparison with other Seq2Seq models

1. Is **contextual information** is crucial for this task?
2. Would **Seq2Seq models** be appropriate for the task?
3. How should the **input** or **architecture** be **adjusted**?
4. Given very little amounts of training data, can we get **SOTA performance**?

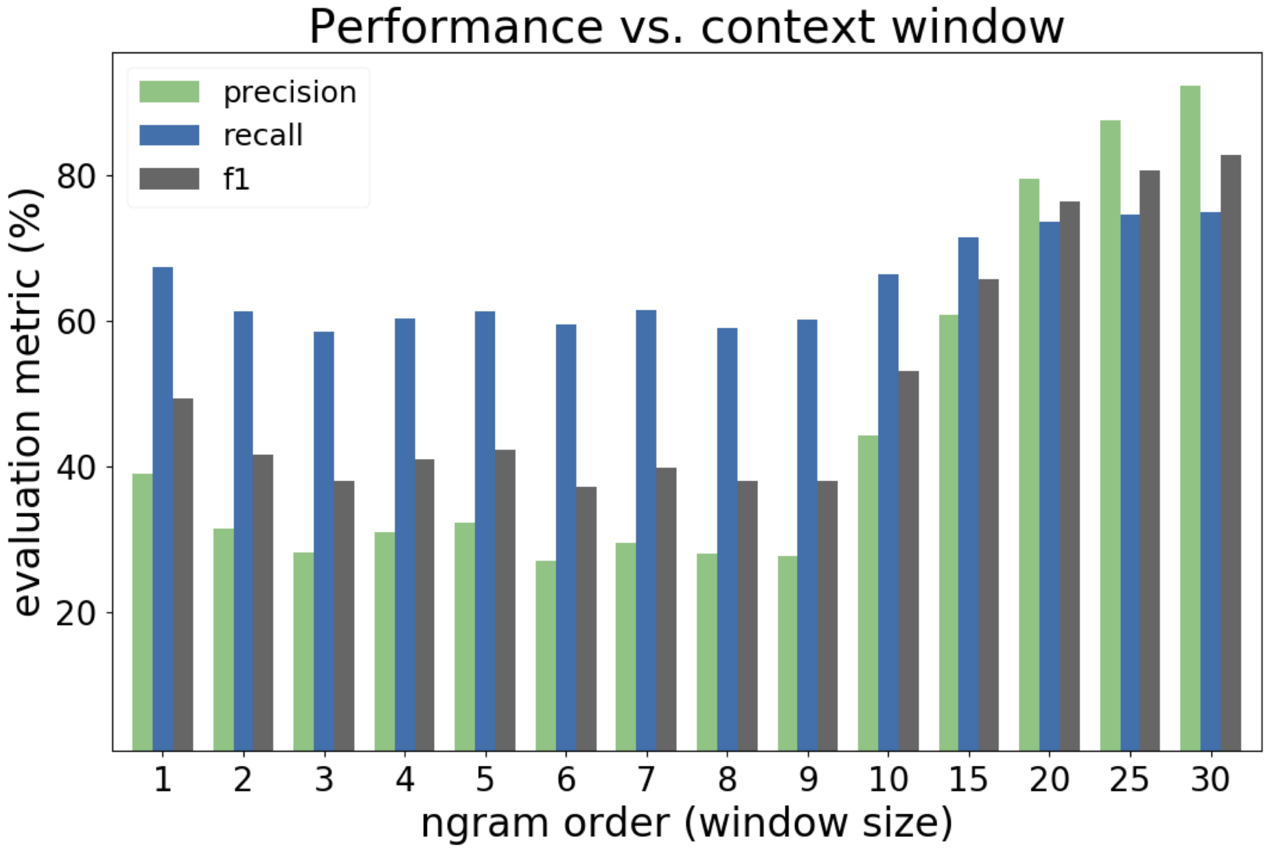
# Window-based split of sequences

2day is my fidst day in Munich

Bigram (2) → {2day is, is my, my fidst, fidst day, day in, in Munich}

5-gram → {2day is my fidst day, is my fidst day in, my fidst day in Munich}

…



# When context helps?

| | |
|---|---|
| **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 ... |

Context is crucial for correct normalization,
especially for short tokens and long sentences

1. Is **contextual information** is crucial for this task?

2. Would **Seq2Seq models** be appropriate for the task?

3. How should the **input** or **architecture** be **adjusted**?

4. Given very little amounts of training data, can we get **SOTA performance**?

# Experimental Results (SOTA)

| 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 |

Comparison with state-of-the-art text normalization systems

# Code is open sourced!

https://github.com/Isminoula/TextNormSeq2Seq

### Requirements

- torch==0.4.1
- python 2.7

### Download the Lexnorm2015 dataset

```
mkdir dataset
cd dataset
wget https://github.com/noisy-text/noisy-text.github.io/raw/master/2015/files/lexnorm2015.tgz
tar -zxvf lexnorm2015.tgz
cp lexnorm2015/* .
rm -rf lexnorm2015 lexnorm2015.tgz
cd ..
```

### Training a hybrid Seq2Seq model from scratch

The hybrid model is a combination of two Seq2Seq models: a word-level one (**S2S**) and a secondary character-level trained on pairs of words (spelling with noise augmented data).

i) Train a word-level model, save results in folder `word_model`

```
python main.py -logfolder -save_dir word_model -gpu 0 -input word -attention -bias -lowercase -bos -eos -b
```

ii) Train a secondary character-level model, save results in folder `spelling_model`

```
python main.py -logfolder -save_dir spelling_model -gpu 0 -input spelling -data_augm -noise_ratio 0.1 -att
```

- ✓ **Pretrained models**
- ✓ **LexNorm 2015 predictions**
- ✓ **Interactive Mode**
- ✓ **Full usage instructions**
- ✓ **Minimal dependencies**

# Some References

[1] T. Baldwin et al., Shared tasks of the 2015 workshop on noisy user generated text: Twitter lexical normalization and named entity recognition, WNUT 2015

[2] N. Jin, NCSU_SAS_NING: Candidate generation and feature engineering for supervised lexical normalization, WNUT 2015

[3] Van der Goot, R., and Van Noord, G. 2017. Monoise: modeling noiseusing a modular normalization system. arXiv:1710.03476

[4] Min, W., and Mott, B. 2015. Ncsu sas wookhee: a deep contextual long-short term memory model for text normalization, WNUT 2015

[5] Leeman-Munk, S.; Lester, J.; and Cox, J. 2015. Ncsu sas sam: deep encoding and reconstruction for normalization of noisy text, WNUT 2015

# Questions?