Neural Multi-Task Text Normalization and Sanitization with Pointer-Generator

Van-Hoang Nguyen
Innovation Lab, PayPal
Singapore
vanguyen@paypal.com

Cavallari Sandro
Innovation Lab, PayPal
Singapore
scavallari@paypal.com

Abstract

Text normalization and sanitization are intrinsic components of Natural Language Inferences. In Information Retrieval or Dialogue Generation, normalization of user queries or utterances enhances linguistic understanding by translating non-canonical text to its canonical form, on which many state-of-the-art language models are trained. On the other hand, text sanitization removes sensitive information to guarantee user privacy and anonymity. Existing approaches to normalization and sanitization mainly rely on hand-crafted heuristics and syntactic features of individual tokens while disregarding the linguistic context. Moreover, such context-unaware solutions cannot dynamically determine whether out-of-vocab tokens are misspelt or are entity names. In this work, we formulate text normalization and sanitization as a multi-task text generation approach and propose a neural pointer-generator network based on multi-head attention. Its generator effectively captures linguistic context during normalization and sanitization while its pointer dynamically preserves the entities that are generally missing in the vocabulary. Experiments show that our generation approach outperforms both token-based text normalization and sanitization, while the pointer-generator improves the generator-only baseline in terms of BLEU4 score, and classical attentional pointer networks in terms of pointing accuracy.

1 Introduction

Early Natural Language Processing (NLP) faced the long-standing limitation of human language understanding, mainly due to linguistic morphology or the wide variance of word forms. Therefore, a crucial requirement to obtain outstanding performance for modern NLP systems is the availability of "standardized" textual data (Guyon et al., 1996; Rahm and Do, 2000). Standardizing or normalizing textual data reduces the domain complexity, hence improves the generalization of the learned model. However, there are challenges to automatic text normalization. Natural language is by nature evolving, *e.g.* Urban Dictionary[1] is a crowdsourced online dictionary for slang words and phrases not typically found in a standard dictionary, but used in an informal setting such as text messages or social media posts. Moreover, abbreviations and emojis allow humans to express rich and informative content with few characters, but troubles machine understanding. Finally, humans are prone to spelling errors while writing or typing.

Due to the reasons mentioned above, developers have designed pre-processing techniques to normalise textual data, including spell correction, tokenisation, stemming, lemmatization and part-of-speech tagging. During the years, multiple libraries have been proposed to facilitate such pre-processing steps: *e.g.* NLTK (Bird, 2006), spaCy[2] or Stanford Core NLP (Manning et al., 2014). However, as textual domains vary greatly from medical records, legal documents to social media posts, there is no single solution or a fixed set of pre-processing steps for text normalization. Thus, up to date, defining a pre-processing pipeline remains an art form which requires a significant engineering effort. While researchers can define hard-policies to eliminate all noisy textual data, they also considerably reduce the amount of information available to the model, thus limit its performance. Such pruning approach appears problematic in the industry where engineers tackle domain-specific problems are given a relatively limited noisy textual dataset.

Enterprises also have to comply with multiple policies concerning privacy. Thus, they are re-

[1] https://www.urbandictionary.com/
[2] https://spacy.io/

Proceedings of the First Workshop on Natural Language Interfaces, pages 37–47

https://doi.org/10.18653/v1/P17

Table 1: Example of well formatted text correctly masked with simple regex rules. Note that all the reported credit card number are artificially generated.

Unmasked Text	**Masked Text**
i need to delete my credit card 5496-9579-4394-2954 the refund will post to your credit card ending in (8077) in the next 3-5 business days	i need to delete my credit card **** the refund will post to your credit card ending in (****) in the next 3-5 business days

Table 2: Examples of text over-masked due to regex application.

Over-Masked Text	**Missing Information**
I sent the faulty product back and provided PayPal with the tracking ********** USPS.	Tracking Number
I was charged a fee for a payment from my son. He owed me *** and I only received 145.90. Can you please refund that fee to my account? Thank you	Money Amount

quired to mask or remove sensitive information rather than cache them inside data centers. Such sensitive information includes credit card numbers, email addresses and Social Security Number (SSN). Note that sanitation issues not only arise during an offline storage/backup process of user-generated content, but they might also happen in real-time. For example, it is common for big enterprises to outsource customer services, like live-chat or chat-bot systems, to third parties. Thus, there is the need to remove all the sensitive information before expose the input text to any third party to prevent information leakage. At the same time, the semantic meaning of a customer's request has to be preserved to deliver good customer support. Enterprises have traditionally addressed sanitization by defining heuristics. Such an approach is effective over well-defined text such as official documents and notes. As shown in Tab. 1, carefully designed regex rules are able to properly mask content following a specific pattern, *e.g.* credit card numbers, from a document[3]. Instead, in an informal setting regex rules can fail due to the presence of typos or sensitive information whose syntax is not accounted in the predefined patterns; for example:

- "my card ending -4810 has being refused."
- "i want to cancel my last transaction 6 9 0 8 2 0 5 7 3 D 1 4 8 0 4 3 3."

On the other hand, rules-based approaches, begin semantic-unaware, tend to mask most of the insensitive but crucial numerical information, troubling the downstream analysis. For instance, Tab. 2 demonstrates a case when a tracking number is confused with a transaction number. Similarly, in the second case, a transaction amount is confused with a credit card number.

As mentioned, we claim that it is not possible to define a general heuristics that correctly cover all the corner cases while ignoring semantics. Instead, we propose a novel approach for text normalization and sanitization based on the recent advancements made in NLP, specifically in Machine Translation (MT). That is, we formulate the joint text normalization and sanitization task as *learning to translate from non-canonical English to a sequence of well-defined or masked tokens*. For example, Tab. 3 demonstrates how malformed texts are translated into a semantically equivalent sequence of well-defined tokens with properly masked information. To our knowledge, this is the first attempt to formulate the joint text normalization and sanitization under MT framework. In so doing, we propose a novel network architecture for MT that can solve this multi-task learning problem.

Moreover, we address the thorny problem of generating unseen tokens during inference in sequence-to-sequence (seq2seq) learning by making use of pointer networks (Vinyals et al., 2015; See et al., 2017; Merity et al., 2016). In addition to the generator, we integrate the *pointer network*, a module that learns to directly copy a specific segment within the input text to the output sequence. Compare to previous work, our design of the pointer is novel as it learns to predict the start and end positions of the correct text segment to be copied, and is built upon the concept of multi-head attention and positional encoding (Vaswani et al., 2017). Experiments show that using a generating-pointing

[3]Note that all the personal information have been anonimized.

mechanism improves normalization performance compared to a pure generating mechanism. Our model can correctly identify and preserve most named entities contained in the input text, potentially benefits downstream analysis.

2 Related Work

The introduction of word embeddings (Hinton et al., 1986; Mikolov et al., 2013; Goldberg and Levy, 2014) has produced a gigantic leap forward for most NLP-related task. Traditional problems such as vector sparsity and word interaction were solved by a simple, yet effective, methodology that exploits a large corpus rather than a sophisticated algorithm. However, such methods are limited by the challenge of inferring embeddings for words unobserved at training time, *i.e.* Out-Of-Vocabulary (OOV). Such scenarios are common in many social-media related applications where the input text is generated in real-time. Thus, the user's malformed language might affect downstream performance (Hutto and Gilbert, 2014). Another solution is to include all the misspelling words in the training dataset or to impose similar embeddings for all n-character variations of a canonical word. This, would not scale well due to the sheer amount of such non-canonical terms; thus researchers have studied the spelling correction problem since long time (Church and Gale, 1991; Brill and Moore, 2000). However, traditional approaches are based on a word-per-word basis; which has shown acceptable results when applied to formal languages.

There have been many robust approaches to token-level spelling correction and lemmatization. The pioneering work done by Han and Baldwin demonstrated that micro-phonetic similarity could provide valuable insight to correct the spelling in an informal context, as many of these relaxed spellings are often based on the word's phonetic, *e.g thr* for *there* or *d* for *the*. Monoise (van der Goot, 2019a) generates feature-engineered n-character candidates for a misspelt word not found in the vocabulary and ranks them using a Random Forest Classifier. However, to accurately identify misspelt words, let alone normalizing them, optimal approaches need to consider the whole contextual semantics rather than the word-level morphology. For example, the utterance *Can I speak to a reel person?* is not misspelt at word-level as every word is a valid English word. However, if we consider sentence-level semantics, *reel* should be normalized into *real*. To factor in such contextual signals, recent advancements in NLP has considered these sequential nature of a written language as well as the long-term dependencies present in sentences. Thus, the research community has proposed different methodologies to perform micro-text normalisation based on deep learning (Min and Mott, 2015; Edizel et al., 2019; Gu et al., 2019; Satapathy et al., 2019). While we address the problem of text normalisation in the NLP context, it has also been adopted as a key component for speech applications (Sproat and Jaitly, 2016; Zhang et al., 2019).

Pointer Network was first proposed to solve geometric problems where the size of the output classes is a variable not conforming to the fixed multi-label classification of traditional seq2seq learning (Vinyals et al., 2015). Pointer Network becomes widely adopted in many NLP tasks including machine translation (Gulcehre et al., 2016), abstractive summarization (See et al., 2017) and language modeling (Merity et al., 2016) as it aids accurate reproduction of factual details such as unseen proper nouns commonly treated as OOVs. However, existing works formulate the pointing operation as a single position classification task that returns one word (token) position in the encoding sequence to be copied to the decoding sequence. Such formulation is no longer suitable for our char-to-word strategy. Furthermore, with the recent state-of-the-art in seq2seq learning introduced by the Transformer architecture, there has not been a comprehensive comparison between different attention strategies, *i.e.* the classical attention mechanisms (Luong et al., 2015) and multi-head attention (Vaswani et al., 2017) on this pointing objective.

Finally, none of the previous research considered the joint privacy-preserving issue, which is common in commercial NLIs such as virtual agents for customer services. To the best of our knowledge, (Sánchez et al., 2012) is the first model that attempted to solve the sanitization problem at a semantic level, without using a rule-based approach (Sweeney, 1996). However, the former approaches are based on manually defined policies that are application and context-specific or are limited to named entities; thus are not generalizable across domains and applications.

Table 3: Example of malformed input text is normalized in the output. Note that the tracking number is mapped to an unknown token while the transaction id is masked for security/privacy reasons.

Input Text	Output Text
D show jst gettin started.	the show is just getting started .
R u thr ?	are you there ?
Why it is my transaction with id 781243692BSD0433 on hold ?	why it is my transaction whith id <msk> on hold ?
I can't enter the tracking number 781243692BSD0433 for a refund.	i can not enter the tracking number <unk> for a refund

3 Problem Formulation

Neural seq2seq models (Sutskever et al., 2014; Cho et al., 2014; Vaswani et al., 2017) became the de facto standard for machine translation systems. Such models are composed by an encoder-decoder architecture which takes an input sequence $\mathbf{x} = [\mathrm{x}_1, ..., \mathrm{x}_M]$ and generate the desired output sequence $\mathbf{y} = [\mathrm{y}_1, ..., \mathrm{y}_N]$ according to the conditional probability distribution $P_\theta^{\text{gen}}(\mathbf{y}|\mathbf{x})$, where θ stands for the model parameters. Due to their well-designed factorisation of $P_\theta^{\text{gen}}(\mathbf{y}|\mathbf{x})$ based on an autoregressive approach:

$$P_\theta^{\text{gen}}(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{N} P_\theta(\mathrm{y}_t|\mathrm{y}_{t-1}, ..., \mathrm{y}_1, \mathbf{x}). \quad (1)$$

seq2seq models have been proven capable of solving the translation task with outstanding results. However, in the traditional MT settings $\mathbf{x}$ and $\mathbf{y}$ are tokens' sequences of different languages, instead, in our context $\mathbf{y}$ represents the same input sentence, but rewritten in a formal and anonymised language.

In addition to the next token generation objective, we formulate the pointing objective as outputting two sequences of start positions $\mathbf{u}^s = [\mathrm{u}_1^s, ..., \mathrm{u}_N^s]$ and end positions $\mathbf{u}^e = [\mathrm{u}_1^e, ..., \mathrm{u}_N^e]$ of the input encoding sequence where $\mathrm{u}_i^s, \mathrm{u}_i^e \in [1, ..., M-1]$. Similar to $\mathbf{y}$, $\mathbf{u}^s$ and $\mathbf{u}^e$ are chosen according to the conditional probability distributions $P_\theta^{\text{pt-start}}(\mathbf{u}|\mathbf{x})$ and $P_\theta^{\text{pt-end}}(\mathbf{u}|\mathbf{x})$ which can be factored as:

$$P_\theta^{\text{pt-start}}(\mathbf{u}^e|\mathbf{x}) = \prod_{t=1}^{N} P_\theta^{\text{pt-start}}(\mathrm{u}_t^s|\mathrm{y}_{t-1}, ..., \mathrm{y}_1, \mathbf{x}), \quad (2)$$

$$P_\theta^{\text{pt-end}}(\mathbf{u}^s|\mathbf{x}) = \prod_{t=1}^{N} P_\theta^{\text{pt-end}}(\mathrm{u}_t^e|\mathrm{y}_{t-1}, ..., \mathrm{y}_1, \mathbf{x}). \quad (3)$$

Note that the factorisation proposed in Eq. 2 (and 3), convert the intractable estimation of $\mathbf{u}^s$ conditioned on $\mathbf{x}$ in a sequence of classification tasks over the sequence length (M) predicting u_t^s based on the previous predictions $\mathrm{y}_{<t}$.

Finally, we learn the optimal θ by maximizing the joint likelihood of the distribution for generative normalisation and sanitisation, $P_\theta^{\text{gen}}(\mathbf{y}|\mathbf{x})$, and the distribution for pointing to the start and end positions for normalisation, $P_\theta^{\text{pt-start}}(\mathbf{u}^s|\mathbf{x})$, $P_\theta^{\text{pt-end}}(\mathbf{u}^e|\mathbf{x})$. In other words, our optimisation problem is the minimization of the well-known cross-entropy loss:

$$\begin{aligned}\theta^* = \underset{\theta}{\text{argmin}} - \sum_{t=1}^{T} \Big[& \hat{\mathrm{y}}_t \log P_\theta^{\text{gen}}(\mathrm{y}_t|\mathrm{y}_{<t}, \mathbf{x}) \\ & + \hat{\mathrm{u}}_t^s \log P_\theta^{\text{pt-start}}(\mathrm{u}_t^s|\mathrm{y}_{<t}, \mathbf{x}) \\ & + \hat{\mathrm{u}}_t^e \log P_\theta^{\text{pt-end}}(\mathrm{u}_t^e|\mathrm{y}_{<t}, \mathbf{x}) \Big].\end{aligned} \quad (4)$$

4 Proposed Method

4.1 Generator

It is possible to formalise the text normalisation task as a seq2seq problem, where malformed English is *translated* in well-defined English. In literature seq2seq (Sutskever et al., 2014) models and the similar Memory Networks (Gulcehre et al., 2017; Weston et al., 2014; Graves et al., 2014) have been widely applied to multiple tasks such as machine translation (Vaswani et al., 2017; Cho et al., 2014) , language inference (Sukhbaatar et al., 2015; Devlin et al., 2018; Dai et al., 2019), question answering (Devlin et al., 2018; Yang et al., 2019) and more. Still, in most cases, the model is expected to serve at a single granularity level: *i.e.* sequence of words to sequence of words (W2W), char-to-char (C2C) or subword-to-subword (Sw2Sw). While this guarantees consistency, these approaches are not suitable for our application. On the one hand, the limited vocabulary size is the main advantage of a C2C approach, but it is more computationally expensive and might generate misspelt words.

On the other hand, a W2W setting is affected by the huge vocabulary size and by the OOV problem, but it guarantees grammatically correct words. Thus, we propose to use a char-to-word (C2V) strategy, where the input sequence is handled as a string of characters, but the output is generated as a distribution over well-formed words. Such a design

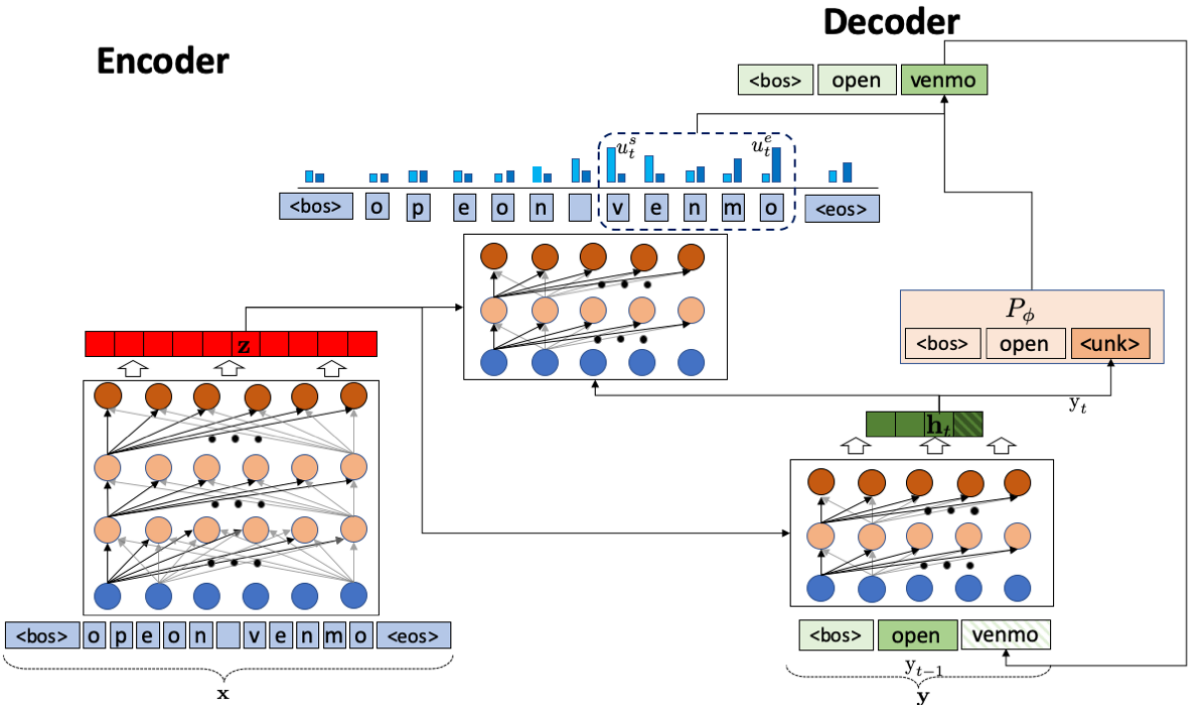

Figure 1: Model architecture. The left part represent a bidirectional encode based on the Transformer architectuGre, while the right part represent an auto-regressive decoder with pointing capabilities also based on the Transformer architecture. Note that, for each decoder timestep, the probabilities of the ith position in the encoder being the start and end positions are calculated from the **start** and **end** pointer distribution. The pointer and vocabulary distribution are derived from the **encoder hidden states** of the input text and **decoder hidden states** of the partial output text.

enables us to handle any input string, solving the problems related to spelling errors while certifying well-formed output. However, it imposes also some challenges; *e.g* how to embed conceptually different objects in the same low dimensional space, or how to learn time dependencies inside a long sequence of characters are only the major problems.

As shown in Fig. 1, given the input embedding of a sequence of characters $\mathbf{x} = [\mathrm{x}_1, ..., \mathrm{x}_M]$, we can formally define an encoder as:

$$\mathbf{q}, \mathbf{k}, \mathbf{v} = \mathbf{x}\mathbf{W}^{\mathbf{q}}, \mathbf{x}\mathbf{W}^{\mathbf{k}}, \mathbf{x}\mathbf{W}^{\mathbf{v}}, \quad (5)$$

$$\mathbf{z} = f(\mathbf{q}, \mathbf{k}, \mathbf{v}) \quad (6)$$

where $f(\cdot)$ is a bidirectional Transformer as defined in (Devlin et al., 2018). Similarly, given the output embedding of a sequence of words $\mathbf{y} = [\mathrm{y}_1, ..., \mathrm{y}_N]$ the decoder is defined as:

$$\mathbf{q}', \mathbf{k}', \mathbf{v}' = \mathbf{y}\mathbf{W}'^{\mathbf{q}}, \mathbf{y}\mathbf{W}'^{\mathbf{k}}, \mathbf{y}\mathbf{W}'^{\mathbf{v}}, \quad (7)$$

$$\mathbf{h}_t = f'(\mathbf{q}', \mathbf{k}', \mathbf{v}', \mathbf{z}) \quad (8)$$

where $f'(\cdot)$ is a traditional Transformer decoder applied in an auto-regressive settings as in (Vaswani et al., 2017). Note that, we are differentiate from the original implementation as we adopt a C2W approach.

4.2 Pointer

We address the limitation of generating unseen tokens in our design of the pointer network. As our generator module predicts a token from a fixed dictionary (vocabulary), it fails to normalise OOVs. We add a *pointer* module to our neural network that allows it to copy a segment of the input text if an unknown word is detected. Although previous works designed their pointer module to point to a single position, for our char-to-word learning problem where each position indicates a character, we propose to jointly point to a start and an end position, while coping all characters in-between. As the output token often consists of consecutive characters, this strategy effectively avoids copying a long continuous character sequence over multiple steps.

Formally, at the decoder timestep t, we learn to output the start position u_t^s and end position u_t^e by maximisation of Eq. 2 and Eq. 3 respectively. The pointer distribution for the start position is a function of the encoder representation $\mathbf{z}$ and the decoder representation $\mathbf{h}_t$ at t, or $P_\theta^{\text{pt-start}}(\mathrm{u}_t^s | \mathrm{y}_{<t}, \mathbf{x}) = g^s(\mathbf{h}_t, \mathbf{z})$. Given that, we can formally define the attention mechanism of the

Transformer architecture as:

$$\text{attn}_i(\mathbf{q}, \mathbf{k}) = \text{softmax}(\frac{\mathbf{q}_i \cdot \mathbf{k}_i^T}{\sqrt{d_K}}), \quad (9)$$

$$\mathbf{z} = [\text{attn}_0(\mathbf{q}, \mathbf{k}), ..., \text{attn}_N(\mathbf{q}, \mathbf{k})]\mathbf{W}^O \quad (10)$$

where d_K stands for the output dimension of $\mathbf{W}^{\mathbf{k}}$, $[\cdot, ..., \cdot]$ is the concatenation of N different attention heads and $\mathbf{W}^O$ is a linear transformation.

Our pointer distribution can be formulated as the attention probability of the last decoder hidden state at timestep t towards each position of the encoder hidden state $\mathbf{z}$. Specifically, we treat $\mathbf{h}_t$ as the query vector $\mathbf{q}$; while $\mathbf{z}$ is the key sequence $\mathbf{k}$ in Eq. 10. We derive the probability of the i-th position of the encoding sequence being the start position as:

$$g_i^s(\mathbf{h}_t, \mathbf{z}) = [\text{attn}_0(\mathbf{h}_t, \mathbf{z}), ..., \text{attn}_N(\mathbf{h}_t, \mathbf{z})]_i \mathbf{W}^s.$$

Notice that unlike the original multihead attention, we did not concern about the value sequence $\mathbf{v}$, but we directly use the attention output to detect the pointing position. Similarly, we define the probability of the j-th position being the end position to copy as:

$$g_j^e(\mathbf{h}_t, \mathbf{z}) = [\text{attn}_0(\mathbf{h}_t, \mathbf{z}), ..., \text{attn}_N(\mathbf{h}_t, \mathbf{z})]_j \mathbf{W}^e.$$

5 Experiments

We conducted 3 experiments to verify the effectiveness of our proposed model. 1) For improved joint normalization and sanitization, we compare our context-aware model with: 1.1) a traditional token-level lemmatizer and spelling corrector, and 1.2) a LSTM W2W encoder-decoder model. 2) For improved normalization of proper nouns, we compare our multi-head attentional pointer-generator with 2.1) a generator-only and a pointer-only baseline, and 2.2) the traditional attention encoder-decoder model. 3) Finally, to address the utility of text normalization we evaluate the performance's improvement obtained on a text classification task with or without text normalization.

The seq2seq transformer architecture we used has 4 attention heads and 5 layers with 100 hidden units. The maximum number input characters and output words are 600 and 300 respectively. During evaluation we maintain a beam size of 3. We determine the correct positions for the pointer network by matching any output word to its character list if the characters appear consecutively in the input character sequence, and noting the start and end position of that character list. Words whose characters are not found consecutively are assign a start and end position of 0 (the beginning of the sequence). We fix the start and end position to the nearest left and right space respectively in the input character sequence to select a complete word. We use the pointer output instead of the generator output whenever the predicted probability for generation is less than 0.6.

Table 4: The datasets' statistics used for evaluation.

	Conversational	**Classification**
Total size	66151	17851
Training Set	54110	6851
Validation Set	6020	5500
Test Set	6021	5500

5.1 Datasets

We conducted the experiments on two datasets:

• The former dataset contains conversations occurred between a customer and a live-chat agent. Human annotators provide the normalized and sanitized version as ground turth. We will refer to this as the *Conversational* dataset and use it for the evaluation of the first two experiments.

• The later one contains utterances collected from a task-oriented chatbot service where customers interact with an agent to solve 27 possible tasks. Each utterance has been manually inspected and assigned to one of the possible class. We will refer to this as the *Classification* datasets and we will adopt it for the last experiments in Sec. 6.3.

We report the dataset statistics in Tab. 4 and the detailed descriptions in Sec. 8.1.

5.2 Baselines

We adopted two baselines to benchmark the ability of the proposed model in the task of normalizing and sanitizing a sentence. The first baseline, Monoise (van der Goot, 2019b) – a lexical normalization tool, is adopted to confirm our model's effectiveness over token-based approaches. Monoise performs normalization via two subtasks: *candidate generation* and *candidate ranking*. The first subtask uses heuristics to select potential normalized forms of each token, including nearest neighbors in word embedding space, edit distance and phonetic distance, and crafted lookup list derived from training, and more. The second subtask first

Table 5: Performance of our proposed Transformer versus baseline in text normalization and sanitization.

Systems	Normalization		Sanitization	
	BLEU4	WER	BLEU4	WER
Monoise	0.9536	0.0206	-	-
LSTM	0.9955	0.0015	0.9827	0.0076
Transformer (Our model)	**0.9986**	**0.0007**	**0.9880**	**0.0052**

engineers features for each candidate, including word embedding distance, n-gram probability, character order, and more. This baseline is used to demonstrate the improvement of our approaches over a heuristic token-based model not only in terms of effectiveness but also efficiency.

The second baseline, LSTM implemented using Fairseq (Ott et al., 2019), is used to highlight the effectiveness of our char-to-word Transformer-based proposal over traditional word-to-word RNN.

Based on the previous research done in the MT filed, we report the test performances of normalization and sanitization in terms of BLEU4 and Word Error Rate (Klakow and Peters, 2002) (WER). The experiment results are described in Tab. 5.

We also (2.1) compare the performance of our proposed pointer-generator model against generator-only model in text normalisation objective and, (2.2) compare multi-head attention against classical attention mechanisms described in a previous work (Luong et al., 2015). The alternative attention formulation considered for benchmarking are:

- **General attention**

$$g_i^s(\mathbf{h}_t, \mathbf{z}, \theta_s) = \mathbf{h}_t^T W_s \mathbf{z}_i$$
$$g_i^e(\mathbf{h}_t, \mathbf{z}, \theta_e) = \mathbf{h}_t^T W_e \mathbf{z}_i$$

- **Concat attention**

$$g_i^s(\mathbf{h}_t, \mathbf{z}, \theta_s) = v_s^T \tanh(W_s[\mathbf{h}_t^T, \mathbf{z}_i])$$
$$g_i^e(\mathbf{h}_t, \mathbf{z}, \theta_s) = v_e^T \tanh(W_e[\mathbf{h}_t^T, \mathbf{z}_i]).$$

Note that, for an overall comparison of the different network architectures considered we used the BLEU4 score. Instead, to evaluate the pointing mechanisms, we compute the accuracy score of the start and end position w.r.t. the correct text's segment, as well as the improved F_1 score of the proposed model and baselines. The experiment results are described in Tab. 7.

Finally, the classification is done using a linear classifier with a bag-of-word approach; which is a common settings in the industry. The performance are evaluated in terms of accuracy and F_1 score. The results are reported in Tab. 9.

6 Results and Evaluation

6.1 Generator

At the macro-scopic level, all translation models, *i.e.* LSTM and the proposed Transformer-based out-perform Monoise. Specifically, our model outperforms Monoise by 0.045 absolute margin or reduces the error by 33 times in terms of BLEU4 score. In terms of WER, the result is performance is consistent where our model reduces the error by 0.02 or by 29 times. Overall, this highlights the improvement of context-aware translation models from context-unaware token-based lemmatizers. We also highlight the superiority of our Transformer-based architecture over the RNN baseline. On normalization task, it is able to reduce LSTM's error by approximately 3 times in terms of BLEU4 and 2 times in terms of WER. On Sanitization task, the proposed model consistently reduces LSTM's error by approximately 1.5 times in terms of both BLEU4 and WER.

For a deeper understanding of the models behavior, we examine the results at micro-scopic level in Tab. 6. We observe that in the first example, Monoise, being unaware of the context, normalize *shipping address* to *ship address*. This can be confusing as the phrase *shipping address* specifically means the delivery address of a package, while *ship address* possibly means the docking location of a large watercraft. Instead, the proposed model is able to consider the contextual information such as *my order*, indicating a package to be delivered, and leave the word *shipping* as it is. In the second example, Monoise leaves the word *real* unlemmatized as *reel* is an existing English word. However, when we factor in the context of virtual agent and the followed word *person*, normalizing *reel* as *real* is more sensible. Overall, also the analysis of the last example demonstrate how the proposed model is able to consider the semantics of an utterance; which eventualy lead to a better results w.r.t. a token-based approach.

6.2 Pointer

As shown in Tab. 7, all the networks with pointing capabilities outperform the Generator-only baseline in terms of BLEU score. Multihead Pointer-Generator improves Generator-only model by the

Table 6: Test examples highlight the behaviours of different methods. Note that, misspelled phrases are highlighted in **red** and correctly normalised phrases are highlighted in **blue**.

Input Text	Transformer (Our model)	Monoise
shipping address is incorrect on my order.	shipping address is incorrect on my order .	**ship** address is incorrect on my order .
can i speak with a **reel** person?	can i speak with a **real** person ?	can i speak with a **reel** person ?
i had money that was refunded to me and i tried sending to my bank account but **its was** on hold	i have money that was refund to me and i try send it to my bank account but **it was** on hold	i have money that was refund to me and i try send it to my bank account but **it is was** on hold

Table 7: Performance of our proposed Multihead Pointer-Generator versus baselines in text normalisation. The *Pointer Start Acc.* and *Pointer End Acc.* denote the accuracy of each system in pointing to the correct start and end position. The *Generating* F_1 denotes the F_1 score of each system in generating the correct next token.

Systems	BLEU4	Pointer Start Acc.	Pointer End Acc.	Generating F_1
Generator-only	0.9532	–	–	0.9243
General Pointer-Generator	0.9583	0.7083	0.7084	0.9229
Concat Pointer-Generator	0.9582	0.707	0.7065	0.928
Multihead Pointer-Generator	**0.9606**	0.7076	0.7076	**0.9334**

Table 8: Test examples highlight the behaviours of different methods.Note that: misspelled phrases are highlighted in **red**, correctly normalised phrases are highlighted in **blue**, mishandled OOVs are highlighted in **gray**, and, correctly pointed OOVs are highlighted in **green**.

Input Text	Generator-only Output	Pointer-only Ouput	Pointer-Generator Output
sennd mony from PayPal to venmo account	**send money** from PayPal to **UNK** account .	**sennd mony** from PayPal to **venmo** account .	**send money** from PayPal to **venmo** account .
how can I **conect** my venmo account with hsbc and citibank account?	how can i **connect** my **UNK** account with **hub** and **UNK** account ?	how can i **conect** my **venmo** account with **hsbc** and **citibank** account ?	how can i **connect** my **venmo** account with **hsbc** and **citibank** account ?
followw PayPal on twitter	**follow PayPal** on **UNK** .	**followw PayPal** on **twitter** .	**follow PayPal** on **twitter** .

largest absolute margin of .0074 or 15.8% error reduction, compared with .0061 absolute margin or 10.89% error reduction and 0.006 absolute margin or 10.68% error reduction from General and Concat Pointer-Generator respectively. These statics confirm our hypothesis that jointly using a pointing and generating mechanism improves the performance of neural models. Moreover, our Multihead Pointer-Generator being highly compatible with the end-to-end transformer-based architecture is the most effective amongst the proposed pointer-based models.

We further seek to understand the improvement brought about by our proposed Multihead Pointer-Generator by examining its accuracy in pointing to the correct start and end positions of the text segment to be copied. Experiment results from *Pointer Start Acc.* and *Pointer End Acc.* Table 8 suggest that there is no significant difference in pointing to the correct positions between the three pointer models. However, the Multihead Pointer-Generator shows a performance boost in terms of

Table 9: Classification performances with and without text normalized and sanitized input.

Systems	Accuracy	F_1 score
with text-norm	0.7696	0.7175
without text-norm.	0.7583	0.6855

F_1 score, where our proposed model enhances the Generator baseline by 0.0091 absolute margin or 12.02% error reduction. This is significantly higher than the changes brought about by General (+1.84% error), and Concat (+4.89% error). This implies that our network design is capable of enhancing a traditional *Generator*-only module when applied to the text normalisation tasks.

6.3 Classification

Finally, we want to evaluate if text normalization and sanitization is also beneficial for downstream tasks. That is, we hypothesized that a normalized and sanitized utterance would be easier to process by another model such as a text classifier. Re-

cent advancement in NLP claims that BERT-like models can easily overcome limitations related to misspelling errors due to their tokenization and pre-training process. However, such models are computational expensive thus are not yet widely adopted in commercial applications that require high-throughput like chatbot services.

It has to be noted that our Conversational dataset contains a broader set of topics and a more variegate lexicon than this dataset. Thus, for this experiment, we directly apply the best performing model of task 1 and 2 to obtain a normalized and sanitized version of the input utterances.

Tab. 9 reports the impact of text normalization and sanitization on a downstream text classification task in our NLI that requires strong natural language understanding. Overall, our proposed model yields a relative improvement of $+1.08\%$ in terms of accuracy and $+4.67\%$ in terms of F_1 score. This indicates that text normalization is beneficial in detecting the classes characterized by a limited amount of training examples.

7 Conclusion

We addressed the importance of context awareness in joint normalization and sanitization. We verified our C2W Transformer-based model's quality over context-unaware word-level lemmatizer and traditional W2W seq-to-seq model at both macroscopic and microscopic level. Moreover, we tackled the limitation of representing and producing OOVs during generation with a pointer-generator that learns to copy the relevant text segments from the source input to the translated output. Experiments at both macroscopic and microscopic level verified improved normalization and sanitization fluency previously limited by OOVs.

Our formulation of text normalization as a learning-to-translate problem avoids the tedious engineering of domain specific preprocessing heuristics for textual data. The proposal of pointer-generator is highly generalizable to other NLP tasks such as summarization or machine translation.

References

Steven Bird. 2006. NLTK: The Natural Language Toolkit. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pages 69–72, Sydney, Australia. Association for Computational Linguistics.

Eric Brill and Robert C Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 286–293. Association for Computational Linguistics.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Kenneth W Church and William A Gale. 1991. Probability scoring for spelling correction. *Statistics and Computing*, 1(2):93–103.

Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Bora Edizel, Aleksandra Piktus, Piotr Bojanowski, Rui Ferreira, Edouard Grave, and Fabrizio Silvestri. 2019. Misspelling oblivious word embeddings. *arXiv preprint arXiv:1905.09755*.

Yoav Goldberg and Omer Levy. 2014. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

Rob van der Goot. 2019a. Monoise: A multi-lingual and easy-to-use lexical normalization tool. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 201–206.

Rob van der Goot. 2019b. MoNoise: A multi-lingual and easy-to-use lexical normalization tool. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 201–206, Florence, Italy. Association for Computational Linguistics.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Jiatao Gu, Changhan Wang, and Jake Zhao. 2019. Levenshtein transformer. *arXiv preprint arXiv:1905.11006*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. *arXiv preprint arXiv:1603.08148*.

Caglar Gulcehre, Sarath Chandar, and Yoshua Bengio. 2017. Memory augmented neural networks with wormhole connections. *arXiv preprint arXiv:1701.08718*.

Isabelle Guyon, Nada Matic, Vladimir Vapnik, et al. 1996. Discovering informative patterns and data cleaning.

Bo Han and Timothy Baldwin. 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*, pages 368–378. Association for Computational Linguistics.

Geoffrey E Hinton et al. 1986. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, volume 1, page 12. Amherst, MA.

Clayton J Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth international AAAI conference on weblogs and social media*.

Dietrich Klakow and Jochen Peters. 2002. Testing the correlation of word error rate and perplexity. *Speech Communication*, 38(1-2):19–28.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Wookhee Min and Bradford Mott. 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*, pages 111–119.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Erhard Rahm and Hong Hai Do. 2000. Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*, 23(4):3–13.

David Sánchez, Montserrat Batet, and Alexandre Viejo. 2012. Detecting sensitive information from textual documents: an information-theoretic approach. In *International Conference on Modeling Decisions for Artificial Intelligence*, pages 173–184. Springer.

R. Satapathy, Y. Li, S. Cavallari, and E. Cambria. 2019. Seq2seq deep learning models for microtext normalization. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083.

Richard Sproat and Navdeep Jaitly. 2016. Rnn approaches to text normalization: A challenge. *arXiv preprint arXiv:1611.00068*.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Latanya Sweeney. 1996. Replacing personally-identifying information in medical records, the scrub system. In *Proceedings of the AMIA annual fall symposium*, page 333. American Medical Informatics Association.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Hao Zhang, Richard Sproat, Axel H Ng, Felix Stahlberg, Xiaochang Peng, Kyle Gorman, and Brian Roark. 2019. Neural models of text normalization for speech applications. *Computational Linguistics*, 45(2):293–337.

8 Appendix

8.1 Dataset Details

As described in Sec. 5.1, we adopted two distinct datasets in our evaluation. Here we are going to describe their characteristics and the annotation process used.

Conversational Dataset

As over-mentioned, this dataset is formed by utterances collected from a chat service where clients interact with customers service agents. Note that such conversations happen in real-time. Thus they contain a huge variety of topics as well as a huge lexicon. Clients can access this chat service from any device, this translate in many syntactic errors present in the utterances as well as an informal language. All the above considerations suggest that many customers adopt mobile devices to interact with these services. The topics covered in such conversations can vary from issues related to financial services to trust problem, which involves third parties not directly participating in the conversations or general chitchatting.

Human annotator has been used to reduce each word o its canonical form, *i.e.* lemmas. In contrast, misspelt words and sensitive/personal information are corrected or masked according to the contextual meaning of the conversation. Note that this labelling process contains little uncertainty; thus, we used a single annotator per utterance to maximise the dataset size.

Classification Dataset

The second dataset is a traditional text classification dataset collected from a task-oriented chatbot system where customers can interact with a chatbot agent to solve 27 possibles task. Note that the user interface is equal for both dataset, but in this case, instead of a human agent, there is a chatbot agent. It has to be noted that we collect only the first utterance typed from the customer since it is the only part needed to classify the customer's need on the 27 classes correctly.

3 skilled annotators have manually annotated each utterance, and we have discharged all the utterances that do not present 100% of agreement. The classes used in this dataset are a subset of the topics appearing in the previous dataset. For example, we have classes related to transactions status, transactions that are declined, dispute for item not received, scam emails or problems related to the account of a customer. Note that, if the chatbot is not able to address the customer's need the conversation would be redirected to an human agent. Thus, a system able to normalize and sanitize utterances from the live-chat service (Conversational dataset), would be directly applicable also to this dataset.