Encoder-Decoder Methods for Text Normalization

Massimo Lusetti[*‡]
massimo.lusetti@uzh.ch

Tatyana Ruzsics[†]
tatiana.ruzsics@uzh.ch

Anne Göhring[*‡]
goehring@cl.uzh.ch

Tanja Samardžić[†]
tanja.samardzic@uzh.ch

Elisabeth Stark[‡]
estark@rom.uzh.ch

[*]Institute of Computational Linguistics, University of Zurich, Switzerland
[†]URPP Language and Space, University of Zurich, Switzerland
[‡]Institute of Romance Studies, University of Zurich, Switzerland

Abstract

Text normalization is the task of mapping non-canonical language, typical of speech transcription and computer-mediated communication, to a standardized writing. It is an up-stream task necessary to enable the subsequent direct employment of standard natural language processing tools and indispensable for languages such as Swiss German, with strong regional variation and no written standard. Text normalization has been addressed with a variety of methods, most successfully with character-level statistical machine translation (CSMT). In the meantime, machine translation has changed and the new methods, known as neural encoder-decoder (ED) models, resulted in remarkable improvements. Text normalization, however, has not yet followed. A number of neural methods have been tried, but CSMT remains the state-of-the-art. In this work, we normalize Swiss German WhatsApp messages using the ED framework. We exploit the flexibility of this framework, which allows us to learn from the same training data in different ways. In particular, we modify the decoding stage of a plain ED model to include target-side language models operating at different levels of granularity: characters and words. Our systematic comparison shows that our approach results in an improvement over the CSMT state-of-the-art.

1 Introduction

Largely influenced by the work on English and other languages with a strong orthographic tradition (e.g. German, Spanish, French), the natural language processing (NLP) pipeline typically requires standardized text as input. Recently, however, text processing has extended to non-standard varieties, including historical texts, transcribed spoken language and user-generated content (blogs, comments, social media posts, messaging). Modern NLP is also increasingly multilingual, starting to address languages that have no writing standard at all.

What is characteristic of non-standard text is a non-uniform way of writing the same word types (e.g. *u* instead of *you* in English). While this might appear as a marginal stylistic variation in English, it is a substantial feature of less standardized varieties. This is the case, for instance, with Swiss German. The German-speaking part of Switzerland is characterized by a phenomenon known as diglossia, i.e. two different varieties of the same language are used within a community in different social situations. One variety is known as standard Swiss German, that is the variety of standard German that is accepted as the norm in Switzerland. It is used in most written contexts (literature, newspapers, private correspondence, official documents), in formal and official spoken contexts (education, parliament speeches) and in interactions with foreigners. The second variety, that is the dialect, is known as Swiss German and is used in everyday life, within the family as well as in most radio and television programs.[1] Since Swiss

[1]See Rash (1998), among other sources, for a comprehensive survey of Swiss German.

German does not have a standardized orthography, it is rarely used in written contexts. However, nowadays we observe an increasing use of the dialect in written computer-mediated communication (CMC). This phenomenon has multiple and interesting repercussions, as it makes valuable material available for NLP tasks, thus granting Swiss German a stronger position among the languages studied in the NLP community. However, given the high degree of variation, the need for text normalization, i.e. mapping different variants of the same word type to a single string, becomes immediately evident. The aim of this work is to normalize WhatsApp messages written in Swiss German. Several factors contribute to the high degree of variation of the source text. Firstly, the lack of a standardized spelling is further complicated by the strong regional variation and the numerous local variants of the same word. As a result, the word *viel* ('much') can appear as *viel*, *viil*, *vill*, *viu*, and many other potential variations. Secondly, CMC is characterized by various peculiarities, such as vowel reduplication and unconventional abbreviations, which increase variation.

A major breakthrough in performing text normalization was achieved when this task was approached as a case of character-level statistical machine translation (CSMT) (Sánchez-Martínez et al., 2013; De Clercq et al., 2013). With a small modification of the input, so that the models are estimated over characters rather than over words, well-known off-the-shelf SMT tools like Moses (Koehn et al., 2007) could be used to obtain significant improvements in comparison to previous solutions.

Currently widely used for text normalization, SMT is slowly abandoned in proper machine translation. New neural methods achieve much better performance, providing at the same time a more flexible framework for designing and testing different models. They, however, require large training sets, which makes them unsuitable for text normalization, where training sets, unlike in machine translation, are small and created by experts specifically for the task. Several attempts have been made to train neural normalization models, but the resulting systems could not reach the performance of CSMT.[2]

In this paper, we tackle the issue of introducing neural methods to text normalization. We work with the neural framework that proved most successful in machine translation: a combination of two recurrent neural networks known as the encoder-decoder (ED) architecture. Inspired by similar approaches to other tasks (Gulcehre et al., 2016; Ruzsics and Samardžić, 2017), we enrich the basic ED architecture with a mechanism that allows us to overcome the limitation of having a small training set. This modification concerns including two kinds of language models at the decoding stage: word-level and character-level. We compare our approach to a strong baseline and the current state-of-the-art CSMT methods.

2 Related Work

Text normalization is primarily performed in processing historical texts, where several automatic approaches have been developed, including a rule-based method that learns rules from training data (Bollmann, 2012), edit distance methods (Baron and Rayson, 2008; Pettersson et al., 2013) and CSMT. Sánchez-Martínez et al. (2013) use CSMT to normalize old Spanish; Pettersson et al. (2014) apply it to old English, German, Hungarian, Icelandic, Swedish; and Scherrer and Erjavec (2016) to historical Slovene.

Outside of historical texts, normalization is mostly performed with CSMT, which has been applied to Dutch user-generated content (De Clercq et al., 2013), Slovene tweets (Ljubešić et al., 2014) and Swiss German dialects (Samardžić et al., 2015; Scherrer and Ljubešić, 2016). CSMT proves particularly suitable for text normalization because it captures well intra-word transformations. One further advantage of CSMT is that it can be highly effective when little training data is available, thanks to a small vocabulary (the set of characters). Once a transformation pattern has been learned for a string of characters, it can be applied to translate unknown words that would be considered out of vocabulary (OOV) in the usual word-level formulation of the task.

CSMT was initially applied to translating between closely related languages, such as Spanish and Catalan (Vilar et al., 2007). Although it did not produce better results compared to word-level SMT, it did help improve overall translation quality when the two levels were combined, taking the output of

[2]See Koehn and Knowles (2017), among other sources, for an analysis of the poor performance of neural systems when training data is limited.

CSMT only for unknown words. Tiedemann (2009) used CSMT for Norwegian and Swedish, concluding that, although it makes more errors than word-level SMT, many errors are of small entity in that the translated word is very similar to the reference. Moreover, he found that CSMT can also learn mappings between words that are not formally similar. Applied to the task of normalization, however, CSMT barely outperforms a simple baseline that consists in selecting, for each source word in the test set, its most common normalization in the training set and copying the source word if it is not found in the training set (Samardžić et al., 2015). The improvement, here too, comes from the relatively good performance on unknown words.

Since the introduction of neural methods to machine translation (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014), some attempts have been made to apply the new framework to the task of normalization. A recent shared task (Tjong Kim Sang et al., 2017) allowed a direct comparison of CSMT with some neural methods, with CSMT still outperforming neural systems. Honnet et al. (2017) apply a neural method embedded in other techniques, but without direct comparison to CSMT. Bollmann and Søgaard (2016) report experiments with deep, long short-term memory (LSTM) networks, but again without a direct comparison to CSMT.

The neural methods applied to text normalization so far employ mostly convolutional neural networks (with the exception of Bollmann and Søgaard (2016)), whereas our approach draws on the line of work known as the encoder-decoder framework. In this framework, one recurrent neural network (RNN) encodes a sequence of symbols into a fixed-length vector representation, and the other decodes the representation into an output sequence of symbols. We extend this with the soft attention mechanism introduced by Bahdanau et al. (2014), that allows a model to search for parts of a source sequence that are relevant to predicting a target symbol.

Our work is closely related to those which implement a modification of the ED framework that allows to incorporate additional language model scores at the decoding stage. Gulcehre et al. (2016) integrated a language model into an ED framework to augment the parallel training data with additional monolingual corpora on the target side. Adapting this framework to the task of morphological segmentation, Ruzsics and Samardžić (2017) introduced a "synchronization mechanism" that allows to integrate language model scores at different levels: the basic ED component is trained on character sequences and the target-side language model component is trained on the sequences of morphemes. We adapt this approach by integrating word-level scores of a language model on top of the character-level neural normalization framework. We compare our method to both the simple memory baseline of Samardžić et al. (2015) and to the standard CSMT, which still represents the state-of-the-art on this task.

3 Data

The data for our experiments comes from manually normalized Swiss German corpora:[3]

- **WUS** set is a corpus of WhatsApp messages (Stark et al., 2014; Ueberwasser and Stark, 2017). The entire collection contains 763,650 messages in different languages spoken in Switzerland. A portion of the data, 5,345 messages in Swiss German, was selected for manual normalization in order to provide a gold standard for automatic normalization. We use this manually annotated portion (a total of 54,229 alignment units) as our main dataset. Table 1 shows examples of alignment units in the corpus.

- **SMS** set is a corpus of SMS messages, again in different languages spoken in Switzerland (Stark et al., 2009 2015). This is a smaller corpus entirely manually normalized. The Swiss German portion contains 10,674 messages. We use this set (a total of 262,494 alignment units) as additional training data, as described in more detail below.

All the messages in our dataset are manually normalized using the same web annotation tool and following the same guidelines (Ruef and Ueberwasser, 2013). This normalization process implies a monotonic alignment between the source tokens and the normalized ones. Table 1 shows the different

[3]The data set used in our experiments can be provided on request. Please contact the authors.

alignment type	source form	normalized form	English gloss
one-to-one	viu	viel	much
one-to-many	hämmers	haben wir es	have we it
	ü ber	über	on; above
many-to-one	aweg riise	wegreissen	tear away; rip off
	morge sport	Morgensport	morning gym
many-to-many	über chunts	überkommt es	receives it

Table 1: Examples of aligned token sequences.

types of aligned token sequences. Most of the alignments are pairs of single tokens (one-to-one alignments). There are also many contracted forms corresponding to multiple normalized words (one-to-many alignments). These are typically verb forms merged with subject and object clitics, as shown in the second example in Table 1. The few cases of many-to-one alignments are due to typos (a space instead of a character) and the lack of spelling conventions for Swiss German, most noticeable in arbitrarily split compounds and separable verb particles. Finally, different combinations of the factors listed above can result in many-to-many mappings.

One peculiarity of the WUS corpus is, unsurprisingly given the source of the texts, the frequent use of emojis, which, if untreated, increase significantly the size of the vocabulary. We address this issue by processing two versions of the corpus.

- **Original** is the version of the corpus as provided by its authors, where emojis are replaced with a sequence of characters describing the symbol. For example, the emoji 😃 is rendered as *emojiQsmilingFaceWithOpenMouth*. While they rely on the same vocabulary as the text (i.e. the alphabet), such long sequences may pose a problem to a character-level normalization system and have a negative impact on training time. Moreover, they might produce normalization errors which could be avoided if the sequence were simply copied from source to target.

- **Modified** is the version of the corpus that we created by representing emojis with their Unicode hexadecimal character codes (U+1F603 for the example above). Also, we have removed hyperlinks found in the original corpus, which are often represented by long character strings and might create confusion for the models we use.

In order to assess the impact of the manipulation of the input data, we perform our experiments on both versions of the corpus.

4 Methods

In the following sections, we describe the details of our adaptation of the ED framework to the task of normalizing Swiss German WhatsApp messages. We also give a short description of the standard CSMT framework implemented in the off-the-shelf software Moses, that we use for the purpose of comparison with the current state-of-the-art. Each source sequence is automatically normalized in isolation and compared with the manually normalized form (reference) for evaluation. Most source and target sequences consist of one-to-one word alignments (see Table 1).[4]

4.1 Encoder-Decoder Model (ED)

We define two discrete alphabets, Σ consisting of the character symbols that form the source sequences (second column in Table 1) and Σ_n of the character symbols that form the normalized sequences (third column in Table 1). Our task is to learn a mapping from an original character sequence $x \in \Sigma^*$ to its normalized form $y \in \Sigma_n^*$. To learn this transformation we apply an encoder-decoder model with soft

[4]Scherrer and Ljubešić (2016) showed that using longer segments can improve performance by capturing a larger context, which can help resolve ambiguity. We decided to focus on the sequences as shown in Table 1 and to leave the use of longer segments for future work.

attention (Bahdanau et al., 2014). Next, we review the architecture of this model presented by Luong et al. (2015). The model transforms the input sequence into a sequence of hidden states, i.e. a fixed-dimensional vector representation, with a bidirectional encoder which consists of forward and backward RNN. The forward RNN reads the input sequence of embedding vectors $\mathbf{x}_1, \ldots, \mathbf{x}_{n_x}$, in forward direction and encodes them into a sequence of vectors representing forward hidden states:

$$\overrightarrow{\mathbf{h}}_t = f(\overrightarrow{\mathbf{h}}_{t-1}, \mathbf{x}_t), \quad t = 1, \ldots, n_x \tag{1}$$

while the backward RNN reads the sequence in the opposite direction and produces backward hidden states:

$$\overleftarrow{\mathbf{h}}_t = f(\overleftarrow{\mathbf{h}}_{t-1}, \mathbf{x}_t), \quad t = n_x, \ldots, 1 \tag{2}$$

where f stands for LSTM (Hochreiter and Schmidhuber, 1997). The hidden state $\mathbf{h}_t$ for each time step is obtained by concatenating a forward and backward state, so that $\mathbf{h}_t = [\overrightarrow{\mathbf{h}}_t; \overleftarrow{\mathbf{h}}_t]$.

The decoder RNN transforms the internal fixed-length input representation into a variable length output sequence $y = (y_1, \ldots, y_{n_y})$. At each prediction step t, the decoder reads the previous output $\mathbf{y}_{t-1}$ and outputs a hidden state representation $\mathbf{s}_t$:

$$\mathbf{s}_t = f(\mathbf{s}_{t-1}, \mathbf{y}_{t-1}), \quad t = 1, \ldots, n_y \tag{3}$$

The conditional probability over output characters is modeled at each prediction step t as a function of the current decoder hidden state $\mathbf{s}_t$ and the current context vector $\mathbf{c}_t$:

$$p(y_t | y_1, \ldots, y_{t-1}, x) = g(\mathbf{s}_t, \mathbf{c}_t) \tag{4}$$

where g is a concatenation layer followed by a softmax layer (Luong et al., 2015). The context vector $\mathbf{c}_t$ is computed at each step from the encoded input as a weighted sum of the hidden states:

$$\mathbf{c}_t = \sum_{k=1}^{n_x} \alpha_{tk} \mathbf{h}_k \tag{5}$$

The weights are calculated by an alignment model which scores how much attention should be given to the inputs around position k to generate the output at position t:

$$\alpha_{tk} = \phi(\mathbf{s}_t, \mathbf{h}_k) \tag{6}$$

where ϕ is a feed-forward neural network (Bahdanau et al., 2014; Luong et al., 2015). Therefore, the model learns the alignment between input and output jointly with transduction using a deterministic function, whereas the alignment is modeled as latent variable in SMT.

The training objective is to maximize the conditional log-likelihood of the training corpus:

$$L = \frac{1}{N} \sum_{(x,y)} \sum_{t=1}^{n_y} \log p(y_t | y_1, \ldots, y_{t-1}, x) \tag{7}$$

where N is the number of training pairs (x, y).

4.1.1 Integrating Language Models

Before the integration, we assume that a plain ED and a language model (LM) are trained separately. The ED model is trained on character sequences in a parallel corpus consisting of aligned source words and their normalized forms (as shown in Table 1). The ED model learns a conditional probability distribution over the normalized character sequences given the source sequences, as shown in Eq. (4). This probability captures context-sensitive individual character mappings, therefore already including the information provided by a usual LM. We augment this model with an additional LM, separately trained only over the target side of the corpus. We propose two ways of augmenting the initial ED. First, following Ruzsics and Samardžić (2017), we train a word-level LM and fuse it with the character-level ED using the "synchronization mechanism". Second, following Gulcehre et al. (2016), we augment the training set with additional target-side data. In the following, we describe how this mechanism allows us to fuse the scores of different models at the decoding stage in a log-linear fashion.

The "synchronized" decoding approach relies on a beam search to find the prediction steps where different scores are combined. The beam search is run at two levels of granularity. First, it produces the output sequence hypotheses (candidates) at the character level using ED scores until the time step s_1, where K best hypotheses $\{(y_1 y_2 \dots y_{s1})^i\}$, $y_t \in \Omega$, $i = 1, \dots, K$ end with a boundary symbol.[5] We consider two boundary symbol types: space, which marks the end of a word in a partial predicted sequence, and a special `eow` symbol, which marks the end of a completed predicted sequence. The step s_1 is the first synchronization step where we re-score the normalization hypotheses with a weighted sum of the ED score and the LM score:

$$\log p(y_{s1}|y_1, \dots, y_{s1-1}, x) \quad = \quad \log p_{ED}(y_{s1}|y_1, \dots, y_{s1-1}, x) \; + \; \alpha_{LM} \log p_{LM}(y_1, \dots, y_{s1}) \quad (8)$$

At this step, $y_1, \dots, y_{s1}$ is considered a sequence of $s1$ characters by the ED system, and one word by the LM. After the first synchronization point we continue to produce the re-scored hypotheses using ED scores until the next synchronization point. The search process ends at the synchronization point where the last symbol of the best scored hypotheses (using the combined ED and LM score) is the end of complete prediction symbol `eow`.

The decoding process scores the hypotheses at two levels: normally working at the character level with ED scores and adding the LM scores only when it hits a boundary symbol. In this way, the LM score helps to evaluate how probable the last generated word is based on the predicted word history, that is the sequence of words generated at the previous synchronization time steps. The described synchronization mechanism is extended in our study to integrate both character-level LM and higher word-level LM.[6] However, in principle, it can be used to add any kind of potentially useful predictors or scores obtained separately from different data sources.

4.2 Character-level Statistical Machine Translation (CSMT)

The core idea behind traditional SMT systems relies on the noisy-channel model, where two basic components are combined: the *translation model* $p(f|e)$,[7] responsible for the adequacy of the translation from source to target sentence, and the *language model* $p(e)$, responsible for the fluency of a sentence in the target language, as shown in Eq. (9), where E is the set of all target sentences.

$$\operatorname*{argmax}_{e \in E} p(e|f) = \operatorname*{argmax}_{e \in E} p(e)p(f|e) \qquad (9)$$

To achieve better context-sensitive source-target mappings, obtained through encoding and memory in the neural approaches, traditional SMT systems rely on phrase-level translation models. These models allow to build a phrase table to store aligned phrase pairs, in the source and target language, that are consistent with the single word alignments established by the IBM models (Brown et al., 1993) with the Expectation-Maximization (EM) algorithm.[8] In phrase-based models, the translation model in Eq. (9) is decomposed as follows:

$$p(\bar{f}_1^I|\bar{e}_1^I) = \prod_{i=1}^{I} \phi(\bar{f}_i|\bar{e}_i) d(\text{start}_i - \text{end}_{i-1} - 1) \qquad (10)$$

[5]Some of the best hypotheses can have length shorter than s_1, but we assume they are of the same length for the ease of notation.

[6]Although it is possible to fuse a character-level LM with the ED system at each prediction step, as in the "shallow fusion" of Gulcehre et al. (2016), in our initial experiments the performance of such model was inferior to our synchronized approach, where we combine the scores for the whole segments at word boundaries.

[7]While, in section 4.1, x refers to the input sequence and y to the output sequence, here we follow SMT conventions and use e to refer to the target sequence and f to the source sequence.

[8]Koehn et al. (2003) improved the mono-directional IBM alignments, which only allow at most one target word to be aligned with a source word, with a heuristic method based on a bidirectional alignment.

In Eq. (10), $\bar{f}_1^I$ and $\bar{e}_1^I$ are sequences of phrases, $\phi(\bar{f}_i|\bar{e}_i)$ is the probability distribution that models phrase translation and $d(\text{start}_i - \text{end}_{i-1} - 1)$ the probability distribution that models the reordering of the target phrases. The various model components (translation, reordering and language model) are weighted in a log-linear model to scale their contribution to the final translation. The reordering model is ignored when reordering is disabled under the assumption of a monotonic translation, which is the case in our experiments.

In CSMT, we simply replace words with characters as the symbols that make up a phrase. CSMT is a suitable approach for those tasks in which many word pairs in the source and target languages are formally similar, such as the Swiss German word *Sunne* normalized as *Sonne* ('sun'), or are characterized by regular transformation patterns that are not captured by word-level systems, such as the pattern *ii* $\rightarrow$ *ei*, which is responsible for the transformations *Ziit* $\rightarrow$ *Zeit* ('time'), *wiiter* $\rightarrow$ *weiter* ('further'), *Priis* $\rightarrow$ *Preis* ('price').

This setting requires to pre-process the parallel corpus by replacing spaces between words with underscores and adding spaces between characters. This converts the corpus alignment unit *hani* $\leftrightarrow$ *habe ich* into *h a n i* $\leftrightarrow$ *h a b e _ i c h* ('I have'). As a result, the characters are now the tokens of the alignment units, phrases are sequences of characters and the language model is based on character n-grams. After running Moses, the predictions made by the system are post-processed, by removing spaces and underscores, before evaluation with a reference.

5 Experiments and Tools

To assess whether our approach provides an improvement over CSMT, we perform a systematic comparison, training and testing both systems on our datasets. In this section we describe the details of the experiments. We run the ED experiments using an extended version of the code from Ruzsics and Samardžić (2017), which offers the possibility to integrate several LM predictors trained on different levels.[9] In a character-level framework, where most alignment units consist of single words, evaluation metrics such as precision, recall and BLEU may provide information on the extent to which a unit normalized by the model, viewed as a sequence of characters, differs from its reference. They thus express the magnitude of the intra-word error. However, we chose to simply assess whether a source sequence has been correctly normalized or not by the system. For this reason, the accuracy score is used to evaluate the baseline and the various models implemented.

5.1 Parameter Settings

ED Hyperparameters and Settings. The character embeddings are shared between input (source) and output (target) vocabulary and set to 100 for the original corpus and 200 for the modified one. The forward and backward RNN of the bidirectional encoder have $H_e = 200$ hidden units each. The decoder also has $H_d = 200$ hidden units. We apply an ensemble of 5 ED models where each model is trained with random start using SGD optimization. The models are trained for a maximum of 30 epochs, possibly stopping earlier if the performance measured on the development set stagnates. The training examples are shuffled before each epoch. We use n-gram order of 7 for the character-level language models. Additionally, the word-level language models used in some of the ED experiments are built on 3-grams.[10] Beam size 3 is used for the final predictions on the test set in all the settings. The weights of the different components of the model are tuned with MERT by maximizing the accuracy score on the development set.

CSMT Settings. We used the Moses toolkit with the following adjustments to the standard settings: i) assuming monotonic character alignment, distortion (reordering) was disabled; ii) in tuning, we used WER[11] instead of BLEU for MERT optimization. We used the KenLM language model toolkit (Heafield,

[9]`https://github.com/tatyana-ruzsics/uzh-corpuslab-normalization`

[10]Kneser-Ney smoothing is used on the modified corpus and modified Kneser-Ney is used on the original one.

[11]WER: Word Error Rate. This metric becomes Character Error Rate in CSMT.

2011) with character 7-grams.[12]

5.2 Baseline and Comparison

The baseline for our task is the one defined by Samardžić et al. (2015), where each original word in the test set is normalized as follows: if the word is found in the training set, use the most frequent normalization, otherwise, copy the source word as its normalization (leave unchanged).

To assess how our modifications influence the performance, we run different versions of our ED system on different datasets. Our plain ED is a soft-attention model described in Section 4.1, with the selected hyperparameters set as described above. In further ED experiments, we integrate language models trained at different levels, character and word, and combine their scores over words in a synchronized decoding approach. The word-level language models are built on the target sides of the two datasets: the train part of the WUS corpus only and its concatenation with the SMS corpus (WUS+SMS). The character-level language model is only trained using the WUS+SMS corpus, since the decoder of the ED system already acts as a (neural) character-level language model over the target side of WUS. In addition, we try a combination of the ED system with both types of language models.

For the purpose of a systematic comparison, we consider two settings for CSMT. First, we train the model on the WUS corpus only. Second, we add an additional language model trained over the target side of the SMS corpus. Note that the CSMT language models operate only at the character level.[13]

5.3 Train/Test Split

We compute the accuracy of the normalized test set word tokens (token sequences in the case of many-to-one or many-to-many alignments), by comparison with the manual normalization. We split the randomly shuffled WUS corpus in 80% training, 10% development and 10% test set, and use these same splits for all our experiments. The original training set contains 43,385 parallel items; the modified training set is slightly smaller with 43,370 parallel items, since we removed hyperlinks from the original. Both test sets contain 5,422 items, the original development set also has 5,422 items, and the modified development set 5,418. For the experiments where we use additional target data, we add 262,494 target token sequences of the SMS corpus. This results in a total of 305,864 items for the extended target WUS+SMS data.

6 Results

The results of our experiments are shown in Table 2. Both the CSMT and the ED models outperform the baseline in all settings. With respect to the ED models, the integration of the additional word-level language model, the first trained on the WUS corpus, the second on the WUS+SMS corpus, results in better performance. Adding a character-level language model trained on the WUS+SMS corpus produces a higher accuracy too, when applied in isolation. Further improvements are observed from combining language models trained on different levels only for the modified corpus. The CSMT method benefits more than the ED method from the additional character-level LM trained on the SMS corpus. However, the capability of ED models to overcome certain limitations of the CSMT approach becomes evident when we exploit the possibility of augmenting them with word-level language models. These produce, for example, improvements in the normalization of foreign words (e.g. source *cream*, where CSMT erroneously forces normalization and gives *kream*), single source words that are normalized as two or more target words (e.g. source *söuis* $\rightarrow$ reference *soll ich es* ('should I [...] it')), and source words whose reference normalization is formally very different (e.g. source *wg* $\rightarrow$ reference *wohngemeinschaft* 'shared apartment'). The best accuracy overall (87.61% for the original corpus) is obtained by the ED model augmented with the word-level LM trained over the extended target data. We observe a slight drop in the performance for the modified corpus which is due to choices of the systems on the ambiguous source items.

[12]We have observed improvements with order 12, but we chose to use a general setting for normalization and to leave this investigation for future work.

[13]It is not a trivial task to incorporate a synchronized LM over words into the CSMT framework and to the best of our knowledge such work has not been done before.

System	Corpus	
	Original	Modified
ED + LMwus+sms:char + LMwus+sms:word	87.57	87.22
ED + LMwus+sms:char + LMwus:word	87.38	87.09
ED + LMwus+sms:word	**87.61**	87.05
ED + LMwus+sms:char	87.38	87.07
ED + LMwus:words	87.15	86.55
ED ensemble 5	87.03	86.50
ED average 5	85.03	84.70
CSMT LMwus+sms:char	86.35	86.43
CSMT LMwus:char	85.30	85.85
Baseline	84.45	84.45

Table 2: Text normalization accuracy scores. Original: corpus with hyperlinks and emojis as description of the symbol. Modified: corpus without hyperlinks and with emojis as symbols. ED: character based encoder-decoder model. CSMT: character-level statistical machine translation. LM: language models on words or characters. ED average 5: average over five encoder-decoder models. ED ensemble 5: ensemble of five encoder-decoder models (all other ED models, except the average, are extensions of this ensemble). wus: corpus of WhatsApp messages. sms: corpus of sms messages.

We carried out a comparison of the predictions made by the best model of each approach with the reference (5,422 test set normalization units), when the original corpus is used. The analysis reveals 229 cases in which only CSMT is wrong, and 161 in which only ED is wrong. A wrong prediction is made by both models in 511 cases. In particular, in 354 cases they make the same error, whereas in 157 cases they make different errors. Many errors common to both models are related to ambiguity in the source text, that arises when one source word has more than one normalization form in the training set. For example, the source word *di* is manually normalized 83 times as *dich* ('you' as object), and 68 times as *die* (feminine definite article). Moreover, both systems have difficulty normalizing source words characterized by irregularities such as vowel reduplication, e.g. *bitteeeee* instead of the more plausible *bitte* ('please'), and by spelling which is not due to an arbitrary choice of the writer, but rather to a typo (e.g., *ado* instead of the more plausible *aso* ('so')).

Of a total of 166 emojis, all of them are correctly normalized by both models in the original version of the corpus. This means that the models are able to effectively process them, thus avoiding the need for solutions that could be cumbersome in terms of framework engineering, such as copying emojis at decoding time.

7 Conclusion

We have shown in this paper that integrating different-level language models into a neural encoder-decoder framework allows a neural method to reach and even improve the performance of character-level statistical machine translation methods, previously considered superior to neural methods in the task of text normalization. The method that we propose is an adaptation of mechanisms introduced in machine translation and morphological segmentation. While the experiments conducted in this paper show the advantage of integrating different-level language models, the adaptation that we propose can be extended to integrating other potential scores into a single encoder-decoder framework. This possibility can be exploited for further improvements of text normalization methods.

Acknowledgements

This research is funded by the Swiss National Science Foundation, project "What's Up, Switzerland? Language, Individuals and Ideologies in mobile messaging" (Sinergia: CRSII1_160714).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Alistair Baron and Paul Rayson. 2008. VARD 2: A tool for dealing with spelling variation in historical corpora. In *Proceedings of the Postgraduate Conference in Corpus Linguistics*, Aston University.

Marcel Bollmann and Anders Søgaard. 2016. Improving historical spelling normalization with bi-directional LSTMs and multi-task learning. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 131–139. The COLING 2016 Organizing Committee.

Marcel Bollmann. 2012. (Semi-)automatic normalization of historical texts using distance measures and the Norma tool. In *Proceedings of the Second Workshop on Annotation of Corpora for Research in the Humanities (ACRH-2)*, pages 3–14, Lisbon, Portugal.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October. Association for Computational Linguistics.

Orphée De Clercq, Bart Desmet, Sarah Schulz, Els Lefever, and Véronique Hoste. 2013. Normalization of Dutch user-generated content. In *Proceedings of RANLP 2013*, pages 179–188, Hissar, Bulgaria.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, and Yoshua Bengio. 2016. On integrating a language model into neural machine translation. *Computer Speech and Language*, 5.

Kenneth Heafield. 2011. KenLM: faster and smaller language model queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, United Kingdom, July.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November.

Pierre-Edouard Honnet, Andrei Popescu-Belis, Claudiu Musat, and Michael Baeriswyl. 2017. Machine Translation of Low-Resource Spoken Dialects: Strategies for Normalizing Swiss German. *ArXiv e-prints*, 1710.11035, October.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA, October. Association for Computational Linguistics.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. *CoRR*, abs/1706.03872.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (Volume 1)*, pages 48–54, Edmonton, Canada.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the ACL 2007 demonstration session*, pages 177–180, Prague, Czech Republic.

Nikola Ljubešić, Tomaž Erjavec, and Darja Fišer. 2014. Standardizing tweets with character-level machine translation. In *Proceedings of CICLing 2014*, Lecture notes in computer science, pages 164–175, Kathmandu, Nepal. Springer.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1412–1421, Lisbon, Portugal, September. Association for Computational Linguistics.

Eva Pettersson, Beáta B. Megyesi, and Joakim Nivre. 2013. Normalisation of historical text using context-sensitive weighted Levenshtein distance and compound splitting. In *Proceedings of the 19th Nordic Conference of Computational Linguistics (Nodalida 2013)*, pages 163–79, Oslo, Norway.

Eva Pettersson, Beáta B. Megyesi, and Joakim Nivre. 2014. A multilingual evaluation of three spelling normalisation methods for historical text. In *Proceedings of the 8th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH)*, pages 32–41, Gothenburg, Sweden.

Felicity Rash. 1998. *The German language in Switzerland: multilingualism, diglossia and variation*. Lang, Bern.

Beni Ruef and Simone Ueberwasser. 2013. The taming of a dialect: Interlinear glossing of Swiss German text messages. In Marcos Zampieri and Sascha Diwersy, editors, *Non-standard Data Sources in Corpus-based Research*, pages 61–68, Aachen.

Tatyana Ruzsics and Tanja Samardžić. 2017. Neural sequence-to-sequence learning of internal word structure. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 184–194, Vancouver, Canada. Association for Computational Linguistics.

Tanja Samardžić, Yves Scherrer, and Elvira Glaser. 2015. Normalising orthographic and dialectal variants for the automatic processing of Swiss German. In *Proceedings of The 4th Biennial Workshop on Less-Resourced Languages*. ELRA.

Felipe Sánchez-Martínez, Isabel Martínez-Sempere, Xavier Ivars-Ribes, and Rafael C. Carrasco. 2013. An open diachronic corpus of historical Spanish: annotation criteria and automatic modernisation of spelling. Research report, Departament de Llenguatges i Sistemes Informàtics, Universitat d'Alacant, Alicante.

Yves Scherrer and Tomaž Erjavec. 2016. Modernising historical Slovene words. *Natural Language Engineering*, 22(6):881–905.

Yves Scherrer and Nikola Ljubešić. 2016. Automatic normalisation of the Swiss German ArchiMob corpus using character-level machine translation. In *Proceedings of the 13th Conference on Natural Language Processing (KONVENS 2016)*, pages 248–255.

Elisabeth Stark, Simone Ueberwasser, and Beni Ruef. 2009-2015. Swiss SMS corpus, University of Zurich. `https://sms.linguistik.uzh.ch`.

Elisabeth Stark, Simone Ueberwasser, and Anne Göhring. 2014. Corpus "What's up, Switzerland?". Technical report, University of Zurich.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Jörg Tiedemann. 2009. Character-based PSMT for closely related languages. In *Proceedings of 13th Annual Conference of the European Association for Machine Translation*, pages 12–19, Barcelona, Spain.

Erik Tjong Kim Sang, Marcel Bollmann, Remko Boschker, Francisco Casacuberta, Feike Dietz, Stefanie Dipper, Miguel Domingo, Rob van der Goot, Marjo van Koppen, Nikola Ljubešić, Robert Östling, Florian Petran, Eva Pettersson, Yves Scherrer, Marijn Schraagen, Leen Sevens, Jörg Tiedemann, Tom Vanallemeersch, and Kalliopi Zervanou. 2017. The CLIN27 shared task: Translating historical text to contemporary language for improving automatic linguistic annotation. *Computational Linguistics in the Netherlands Journal*, 7:53–64, 12/2017.

Simone Ueberwasser and Elisabeth Stark. 2017. What's up, Switzerland? A corpus-based research project in a multilingual country. *Linguistik Online*, 84(5).

David Vilar, Jan-Thorsten Peter, and Hermann Ney. 2007. Can we translate letters? In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 33–39, Prague, Czech Republic.