# Normalising Non-standardised Orthography in Algerian Code-switched User-generated Data

**Wafia Adouane, Jean-Philippe Bernardy and Simon Dobnik**
Department of Philosophy, Linguistics and Theory of Science (FLoV),
Centre for Linguistic Theory and Studies in Probability (CLASP), University of Gothenburg
firstname.lastname@gu.se

## Abstract

We work with Algerian, an under-resourced non-standardised Arabic variety, for which we compile a new parallel corpus consisting of user-generated textual data matched with normalised and corrected human annotations following data-driven and our linguistically motivated standard. We use an end-to-end deep neural model designed to deal with context-dependent spelling correction and normalisation. Results indicate that a model with two CNN sub-network encoders and an LSTM decoder performs the best, and that word context matters. Additionally, pre-processing data token-by-token with an edit-distance based aligner significantly improves the performance. We get promising results for the spelling correction and normalisation, as a pre-processing step for downstream tasks, on detecting binary Semantic Textual Similarity.

## 1 Introduction

Natural language processing (NLP) research has achieved impressive results, notably thanks to the use of deep neural networks (DNNs) which has pushed the field forward, achieving unprecedented performance for various tasks. However, research is often focused on large, standardised, monolingual and well-edited corpora that exist for a few well-resourced languages. We believe that such corpora will not generalise to all languages and domains, particularly regarding the colloquial varieties used in new communication channels. In fact, the large unstructured data coming from such channels is not only unedited, it also poses serious challenges to the current NLP processing pipelines and approaches as a whole.

Traditionally, the *standard language ideology* has dominated linguistic studies: it has been frequently assumed that languages are naturally uniform and monolingual. Nevertheless, the new online data reveals that standardisation is neither natural nor universal, it is rather a *human invention* (Milroy, 2001), and variation is the norm. This variation presents several challenges to studying and processing dialects in social media (Jørgensen et al., 2015). These challenges are even more pronounced in multilingual societies where people use more than one language or language variety at the same time. We consider the case of the colloquial language used in Algeria (hereafter referred to as ALG) which combines both linguistic challenges mentioned above: (i) it is non-standardised, and (ii) it is a mixture of languages which involves code-switching between Modern Standard Arabic (MSA) and local Arabic, French, Berber, and English. (We refer the interested reader to the work of Adouane et al. (2018), who provides an overview of the linguistic landscape in Algeria.)

In interactive scenarios, people usually use spoken-like language and spontaneous orthography which reflects local variations. Our observations confirm those of Eisenstein (2013), namely that speakers have some-kind of tacit knowledge of spelling which is not completely arbitrary. However, it is hard to distinguish between local varieties and draw a clear borderline between them due to the free mobility of people, their ability to interact online, and the fact that these varieties are closely related and therefore hard to describe formally. Therefore, we find that using location to map dialectal variation (Doyle, 2014) is not useful. In many cases, the spelling is not consistent even by a single person within the same conversation. There is nothing intrinsically wrong with this inconsistency for there is no standard form to take as a reference. Besides, spelling variation does not hinder mutual understanding.

Current NLP approaches based on learning underlying regularities from data is not suitable to



*Proceedings of the 2019 EMNLP Workshop W-NUT: The 5th Workshop on Noisy User-generated Text*, pages 131–140
Hong Kong, Nov 4, 2019. 

sparse noisy data. Furthermore, the data written in Arabic script is already rich in orthographic ambiguity because vowels are not written, except in very specific settings. Our focus is to process such user-generated textual data, reflecting the real use of a language. Therefore, for computational purposes, we want to automatically reduce the data sparsity caused by spelling inconsistency by normalising it based on spelling decisions that we designed, and build a tool that can be used for preprocessing such texts for other NLP tasks.

This paper is an attempt to take advantage of DNNs to reduce spelling inconsistency by performing several transformations (normalisation, disambiguation, etc.) detailed in Section 3 as a single machine-learning task. It is significantly different from the well-established spelling error correction mainly because we have to deal with a non-standardised code-switched language. In addition to the fact that ALG is an under-resourced language with respect to the size, quality and the diversity of the available labelled data, it suffers from the absence of other tools and linguistic resources required by current NLP techniques such as tokenisers, syntactic parsers, morphological taggers, lexicons, etc.

As contributions, (i) we introduce a new user-generated corpus for ALG with its parallel spelling normalised and corrected version produced by human annotators. (ii) We describe our spelling decisions aiming to reduce orthographic ambiguity and inconsistency for NLP tasks. These decisions are not the only possible ones, and can be debated and further refined. (iii) We propose a general end-to-end model for context sensitive text normalisation of non-standardised languages. We opt for end-to-end deep learning approach (with only a simple automatic pre-processing) because it is not only expensive and time consuming to build equivalent rule-based tools from bottom up, but it is also hard to exhaustively define spelling norms given the high linguistic variation.

The paper is organised as follows. In Section 2 we survey related work. In Section 3, we present our newly compiled parallel corpus and explain our data processing decisions. In Section 4, we give information about data statistics and data alignment. In Section 5, we describe our models. In Section 6, we describe our experiments and discuss the results. We conclude in Section 7 with potential future improvements.

## 2 Related Work

The task of normalising user-generated non-standardised data is closely related to the one of historical text normalisation (Pettersson, 2016), namely they present similar challenges for the current NLP – little sparse data. While the latter has a standardised spelling as a reference, the former does not because many colloquial languages have not undergone the standardisation process. Bollmann (2019) surveys the approaches used for historical text normalization for a set of languages. Both tasks are mainly framed as (statistical/neural) machine translation mostly at a token level where the source and the target language are the same or a standardised version of one another.

Similarly to the previous work, we formulate our task as a sequence-to-sequence (seq2seq) learning problem, but in contrast we take word context into account. A large body of work has been done to address the problem of seq2seq prediction and has achieved impressive results for diverse NLP tasks. Encoder-decoder models are most frequently used for seq2seq prediction with varying the architectures of the encoder like Recurrent Neural Network (RNN) in (Cho et al., 2014; Sutskever et al., 2014), bidirectional Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) in (Bahdanau et al., 2014), Convolutional Neural Networks (CNN) in (Vinyals et al., 2015).

Our CNN-based architecture (see Section 5) is reminiscent of what has been proposed for machine translation by Gehring et al. (2017) but instead they use CNN for both encoder and decoder with multi-step attention. A difference with our model is that we use two sub-networks (LSTM/CNN and CNN/CNN) as an encoder, jointly trained to learn contextual representations of words. Then we use an LSTM as decoder instead of a CNN. Compared to the model of Bahdanau et al. (2014), an important difference is that we do not jointly train alignment and seq2seq prediction. Instead we perform alignment separately as a pre-processing step using edit-distance.

None of the mentioned models have been tested on the same prediction task as ours or on a related language. As the most closely related work for spell checking, Ghosh and Kristensson (2017) propose a seq2seq neural attention network system for automatic text correction and completion. They combine a character-based CNN and

a Gated Recurrent Unit (GRU) (Cho et al., 2014) as encoder and a word-based GRU as decoder using a 12 million word English corpus. Recently, Sooraj et al. (2018) employed a character-based LSTM language model to detect and correct spelling errors for Malayalam. In the same line of research, Etoori et al. (2018) propose an attention model with a bidirectional character-based LSTM encoder-decoder trained end-to-end for Hindi and Telugu spelling correction using synthetic datasets.

Contrary to the task we are trying to address in this paper, the mentioned work deals either with spelling correction for monolingual standardised languages or historical text normalisation for standardised languages. This makes our task linguistically more challenging because our data includes more languages hence the model has to find the correct spelling of a word not only based on its context but also based on its language.

There has been work done for Arabic automatic error correction mainly for MSA including the work of Shaalan et al. (2012) and others included in the Arabic shared task (Mohit et al., 2014). Still they are inadequate to process non-standardised Arabic varieties given the significant phonological, morphological and lexicon differences between MSA and Arabic dialects (Watson, 2007). To the best of our knowledge, this is the first effort to process user-generated non-standardised dialectal Arabic textual data end-to-end.

## 3 Data Preparation

### 3.1 Corpus creation

As a basis we take the extended version of the unlabelled dataset of Adouane et al. (2018). Our extended version of it consists of 408,832 automatically pre-processed user-generated short texts from social media, such as forum discussions, and contains more than 6 million words. The automatic pre-processing involves removal of punctuation, emoticons and reduction of repeated letters to a maximum of two. Indeed, Arabic orthography does not use more than two adjacent occurrences of the same letter, and repeats in social media texts are mainly typos or emphasis. For this work, we further pre-processed this dataset by removing any existing diacritics representing short vowels because they are used rarely and inconsistently, even in the texts generated by the same user. We assume that such idiosyncratic variation will not affect our task in terms of semantics and bring about more robustness to language processing, especially because diacritics are not commonly used outside of the formal register. We also normalised many commonly used (often french-based) Latin script abbreviations to their full versions using the most frequent spelling in Arabic script including psk/because, r7/recipe, bnj/good morning, b1/well, 2m1/see you tomorrow, dsl/sorry, on+/moreover, tj/always, etc.

All texts are written in Arabic script and display spelling variations, typos and misspellings wrt. MSA, diglossic code-switching between MSA and local colloquial Arabic varieties, bilingual code-switching between Arabic varieties; French; Berber and English. From this further pre-processed unlabelled dataset, we created a parallel corpus of manually normalised texts. For this purpose, we randomly selected 185,219 texts and had 5 human annotators, who are native speakers with (computational) linguistics background, to edit and process them. The process took 6 months mainly working on lexical and syntactic ambiguities which require linguistically informative decisions, and all annotators checked the annotations of each other. We give here a few examples of spelling variation, but the corpus contains 50,456 words and 26,199 types to be normalised or corrected. Note that we will use word to refer to lexical words and tokens to refer to lexical words plus digits and interjections.

### 3.2 Annotation standard

In order to guide the annotators in producing parallel normalised text, we designed the following annotation standard which involves (i) spelling correction and (ii) spelling normalisation tasks.

#### 3.2.1 Spelling correction for MSA

Misspelled MSA words are corrected using MSA orthography based on their context. نضيف، غذاءيه، جزاءر، مناقشه (clean, nutritional, Algeria, discussion) are corrected as نظيف، غذائية، جزائر، مناقشة.

#### 3.2.2 Typographical error correction

The texts have been written on different kinds of keyboards resulting in lot of typos which mainly include missing spaces like in ومخلوهاشتخرجو or additional spaces like in لعا يلة which have been respectively corrected as وماخلوهاش تخرج و (and they did not let her to go out and) and لعايلة (the family). There are also keyboard related typos like

reversing the order of letters or substituting one letter by another like in ولبدي where ب should be replaced by ي to get the correct intended word وليدي (my son).

These typos can be detected from their context by manual checking. Usually they are not valid words and tend to be consistently generated by the same user which suggests that they may be related to their typing style and conditions. In خيها فغيها the user used the same wrong letter ى twice instead of ر and the correct form is خيرها فغيرها (the better is in something else).

### 3.2.3 Spelling normalisation

Non-MSA words including local Arabic varieties, French, Berber, English and neologisms are spelled spontaneously in Arabic script where users use improvised phonetically-based orthography.

**A. Local Arabic varieties** To deal with the spelling variation in colloquial varieties, a conventional orthography for dialectal Arabic (CODA) has been proposed for Egyptian (Eskander et al., 2013) and has been extended for Algerian (Saadane and Habash, 2015) and recently for several other Arabic varieties (Habash et al., 2018). We share the overall goals with the authors of CODA that a conventional orthography for developing NLP tools should preserve phonological, morphological and syntactic information of dialectal texts, should not diverge substantially from the existing forms, and should be easy to learn and write by the annotators.

However, CODA is primarily a recommendation of guidelines with several open questions related to how these guidelines could be implemented in new scenarios. In our case the most relevant open question is how to deal with multilingual code-switched data found in ALG. Using the existing recommendations from CODA would be in several cases impractical because several phonological distinctions required by the varieties in ALG could not be encoded and would have to be listed as exceptions. In other cases, the application of CODA would also require a substantial rewriting of the original user-generated text. Instead we use data statistics as heuristics to find the canonical forms.

We first train word embeddings using FastText (Joulin et al., 2016) on the entire unlabelled data. We collect a list of all words in the corpus and for each word we use FastText to predict the 10 most similar words and their frequencies. This normally returns the spelling variations of that word. A human annotator then decides whether the returned cluster should be considered as a spelling variation and assigns the most frequent spelling as the canonical form for all word occurrences in this cluster.

This is not a trivial task to be performed fully automatically because the model often returns unrelated words for less frequent words (case of the majority of words in the dataset). Hence a human expertise is needed. Contrary to CODA where every word has a single orthographic rendering, if a word has more than one frequently occurring spelling we keep such variations because they reflect local lexical or phonological differences which may be useful for sociolinguistic studies. For example, we keep both spelling variations of question words قداش، قداه and علاش، وعلاه (when and why) because they occurred very frequently and could be mapped to the same form if needed.

In cases where the difference between MSA and local Arabic spelling of a word is based on phonetically close sounds such as the sounds س [s] and ص [$s^{\varsigma}$] as in سمعة، صمعة (reputation) or between ت [t] and ط [$t^{\varsigma}$] as in تريق، طريق (road), and the meaning is preserved, MSA spelling is used. These cases are hard to identify automatically and require human expertise. Making spelling MSA-like as practically as possible will facilitate the reuse of existing MSA resources. Nevertheless, in cases where a word does not exist in MSA and has several different spellings, the most frequent one is used provided that it is not homonymous with another existing word. Such words include frequent local Arabic words like علاجال ، ضك ، امالا (so, now, for) with 27, 59 and 39 spellings respectively, along with the newly created words like نريض (I practise sports) and نرمضن (I fast).

**B. Non-Arabic words** The dataset includes French, Berber and English words, and the limitation of the Arabic script creates more ambiguity regrading how to spell non-existing sounds like /g, p, v/. The most frequent spelling with long vowels is used. For example, the French word “journal”

(newspaper) occurs with 6 spellings all mapped to جورنال which is the most frequent spelling.

#### 3.2.4 Word sense disambiguation

Using various languages with spelling variation at the same time creates considerable ambiguity, especially when all the varieties are written in the Arabic script. One particular frequent source of lexical ambiguity concerns the spelling of the French definite articles (le, la, les) spelled as ، لي ، لو ، لا, either separated or concatenated to the word they associate with. However, the Arabic spelling is ambiguous because each of the above words means something else in MSA or local Arabic. For instance, لي when written as a separate word could either be a prepositional phrase (for me) in MSA or a relative pronoun (who / that / which) in local Arabic. For this reason we decided to spell French definite articles attached as prefixes like the Arabic definite article ال . This allows disambiguation of cases like: لي ماش (hair strand dyeing) in French and (who is not) in local Arabic.

The Berber word for "window" is spelt as طاقة which means energy in MSA. Since Berber does not have a standardised spelling in Arabic script[1], we decided to change the spelling to تاقة which is another spelling found in the dataset. Furthermore, lexical ambiguity is caused by the absence of sounds (and corresponding graphemes) in Arabic like /g,v,p/. "Group" is spelled : ، قرووب غروب ، قغوب ، قروب ، جروب where غروب and قروب mean "sunset" and "closeness" in MSA. To disambiguate these senses قـغـوب is used for "group".

#### 3.2.5 Negation particle

The various spellings of the word ما cause significant lexical and syntactic ambiguity. When written separately, it could be a relative pronoun or an interjection in MSA, a feminine possessive pronoun in French, 'mother', 'water' or a negative particle in local Arabic. We decided to spell this negation particle as a proclitic with a long Alif when used with verbs (ما instead of م). This removes ambiguity for cases like the local Arabic negated verb ماكان (there was not) from the MSA noun مكان (place) and the local Arabic هدا ما كان (that's it). All negated verbs in local Arabic are spelled with ما as proclitic and ش as enclitic. As a result it is easier to get the non-negated form by stripping off the negation clitics. By removing the initial ما and the final ش from ماعيطش (he did not call) we get عيط (he called).

#### 3.2.6 Word segmentation and tokenisation

Users tend to spell prepositions, reduced question words and conjunctions as proclitics. This creates an unnecessary sparse and large vocabulary. To reduce the size of the vocabulary, we write such proclitics as separate full forms, among others: وش ، ح ، م ، في ، كي ، او ، ف ، واش ، شا ، غ ، ك، بلا، تع ، ع ، عل ، ه ، ولا. We split لي and ما when they occur as relative pronouns attached to a verb. وشيكون (who is him) is tokenised to واش يكون, مالازمة (from the crisis) as من الازمة , حنديرلو (I will make him) راح نديرلو, and split relative pronouns in ماتتمناي (what you wish) as ما تتمناي and ليكان (who was) as لي كان. Other ambiguous cases include ورانا which could be either وين رانا (where are we) or و رانا (and we are) or وراءنا (behind us) depending on the context.

#### 3.2.7 Abbreviations and acronyms

We collapse acronyms written as several tokens to a single token and extend abbreviated words to their full form based on their context. For instance, لاس أم اس is collapsed to لاساماس (SMS), and م ك is extended to مغرف كبيرة (tablespoon).

## 4 Data Statistics and Alignment

### 4.1 Data statistics

The final processed parallel corpus, described in Section 3 consists of 185,219 unique (input, output) text pairs where the input is from the automatically pre-processed data and the output is from the manually corrected and normalised data. The input corpus has 3,175,788 words, and 272,421 types (unique words) where 90.20% of them occurred less than 10 times and 59.60% occurred only once in the entire corpus. These figures serve to give an idea about how sparse the data is. The

[1]Berber has its own script called Tifinagh and a standardised Latin spelling.

longest text has 112 words. The output corpus has 3,125,332 words and 246,222 types (unique words). The longest text has 112 words. The difference in the vocabulary size between the two corpora (50,456 words and 26,199 types) is primarily because of the introduced transformations.

### 4.2 Data alignment

Another difference between the two corpora is that the lengths of the input and the output may vary as a result of different tokenisation. This is not a problem in terms of machine learning, because the models described in Section 5 are designed to deal with variable length input and output sequences. However, because our two sequences are from the same language with the same meaning (the only difference is in spelling) we expect that alignment at the token level will lead to improved performance (see Section 6.1).

To this end, we have developed an aligner whose task is to make sure that every single unit (token) in the input (with potential misspelling) matches a unit (token) in the output. This may seem trivial until one remembers that misspellings may include added or deleted spaces. Our aligner works by computing the minimal edits to transform the input into the output (using the standard Levenshtein distance algorithm).

These minimal edits are not the basis for training (they will be discarded) *unless* they concern spaces. If a space is added, then to preserve word alignment we replace the corresponding space in the output by a special symbol (#). In inference mode (see Section 5.4), this symbol will be replaced by a space. If on the contrary a space is deleted, then it is added back (and words are aligned again). A special extra symbol ($) is added to mark that a spurious space was added and should be eventually deleted again when the model is used in inference mode. This alignment algorithm provides correct results whenever the Levenshtein distance at the sequence level is the sum of the Levenshtein distances for each unit (token) that is misspellings are not so large as to make deleting/inserting whole words a shorter operation than changing characters within words; and this condition is satisfied in our corpus.

## 5 Models

We frame the task of spelling correction and normalisation as a sequence-to-sequence (seq2seq) prediction problem, i.e., given an input sequence what is the best predicted output sequence. Note that sequence refers to user texts of any length including one token or more. We use an encoder-decoder architecture which consists of two neural networks where the first one reads an input sequence and transforms it into a vector representation, and the second one, representing a language model, conditions the generation of output symbols on the input sequence representation and generates an output sequence (Cho et al., 2014).

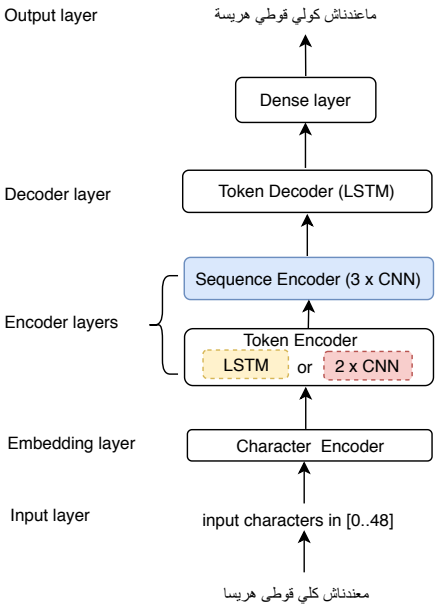


Figure 1: Model architecture.

As shown in Figure 1, the encoder consists of two sub-neural networks, namely token encoder and sequence encoder.

### 5.1 Token encoder

It reads the input sequence character by character and outputs a vector representation for each token in the sequence. Two configurations are used: either an LSTM encoder or a CNN encoder.

- **LSTM encoder:** represented in yellow and takes as input character embeddings with vocabulary size of 49, 100 dimensions, token representation size of 50 and a dropout rate of 30%.
- **CNN encoder:** represented in red and takes as input character embeddings. It is composed of 2 CNN layers with 50 filters of size 5, a RELU activation, a dropout rate of 20% followed by max pooling in the temporal dimension.

### 5.2 Sequence encoder

Represented in blue and consists of 3 CNN layers with 200 filters for the two first layers and 100 for the third layer, all filters have size 3, a RELU activation and dropout rate of 5%.

### 5.3 Token decoder

It is composed of one character-based LSTM layer with the same hyper-parameters as the LSTM encoder, followed by a dense layer.

### 5.4 Training and inference

All models are trained end-to-end to maximise the likelihood of the output sequence conditioned on the input sequence for 150 epochs using a batch size of 64 and Adam optimiser. Gradients with a norm greater than 5 are clipped.

For inference (generating an output character sequence), we use beam-search with a size of 20. Note that beam-search is used only to generate an output sequence and does not influence neither model training nor validation. The models generate characters starting from the start symbol (<) and stop at the end symbol (>) or at a predefined sequence length given as a hyper-parameter, whichever comes first.

## 6 Experiments and Results

In order to test our models and the gain from the aligner (see Section 4.2), we experiment with both versions of data: the non-aligned and the aligned data. It is worth mentioning that the only difference between them is that the aligned one contains extra symbols (# and $) marking missing or extra spaces. An extra space – thus word– is also added for every dollar sign. Moreover, to measure the effect of the context, we feed the data either token-by-token or sentence by sentence.

We split both the datasets into 75% (138,917 samples) for training, 5% (9,261 samples) for development, and 20% (37,041 samples) for validation. The reported hyper-parameters in Section 5 were fine-tuned on the development set.

We conduct two evaluations: (i) how well the suggested models perform on the seq2seq task, and (ii) how good is the best performing model for spelling correction and normalisation task, and what is its effect as a pre-processing step on downstream tasks like Semantic Textual Similarity (STS). We evaluate (i) using character-level accuracy, and we evaluate (ii) by calculating Precision, Recall and the F-score for the class of tokens that should be changed. Hence, Recall is the ratio of the correctly changed tokens to the number of tokens that should be changed, and Precision is the ratio of the correctly changed tokens to the number of tokens that are changed. F-score is the harmonic average of both.

### 6.1 Comparing models on Seq2seq task

In Table 1 we report the overall character level accuracy of the 4 best performing models for each configuration and experiment: (1) LSTM-Token-seq: the model with the Token LSTM + Sequence encoder (yellow and blue parts of Figure 1) and Token decoder, (2) CNN-Token-seq: the model with the Token CNN + Sequence encoder (red and blue parts of Figure 1) and Token decoder. Both (1) and (2) are trained and evaluated on non-aligned data with a sequence of tokens as input. (3) CNN-Token-seq-alig the same as model (2) but trained and evaluated on aligned data. (4) CNN-Token-token-alig: the same as (3) but with one token as input (token-by-token).

Results indicate that the LSTM encoder in (1) does not suit our task / data and fails to learn the sequential representations with an overall character accuracy of only 23.90%. This could be because of the high sparsity of the data which makes it hard to learn regularities. In contrast, the CNN encoder in (2) performs much better, with an overall character accuracy of 89.20%, suggesting that learning sequences of patterns through convolutions suits better our task / data than sequence modelling with LSTM. This is in line with what has been reported for machine translation in (Gehring et al., 2017).

The CNN encoder performs even better with the aligned data in (3). The difference can be attributed to the positive effect of the aligner which boosts the accuracy by 7%. The 9.1% drop in the accuracy in (4) compared to (3) is due to the lack of word context. This indicates that word context is essential, especially for word sense disambiguation in such highly varied data.

### 6.2 Quality and effect

- **Quality** We use the best performing model (3) and run the inference mode, (see Section 5.4), on the validation set which contains 567,308 words of which 507,429 words are already correctly spelled and 59,880 words must be changed, either corrected or normalised. We

| | Models | Input | Data | Validation |
|---|---|---|---|---|
| 1 | LSTM-Token-seq | sequence of tokens | non-aligned | 23.90 |
| 2 | CNN-Token-seq | sequence of tokens | non-aligned | 89.20 |
| 3 | CNN-Token-seq-alig | sequence of tokens | aligned | **96.20** |
| 4 | CNN-Token-token-alig | one token | aligned | 87.10 |

Table 1: Accuracy of models (%) on Seq2seq task.

perform quantitative and qualitative analysis of the generated sequences in terms of the changed spellings at a word level. Model (3) achieves an overall F-score of 64.74%, Recall of 88.58% and Precision of 51.02% on the words to change. It correctly spells 53,041 words from the total words to change and fails to correctly change 6,839 words. However, it introduces 50,914 incorrect changes (newly misspelled words or infelicitous corrections).

- **Error analysis** Examining the generated sequences shows that most errors are at the level of one character (duplicating or substituting one character) and the generated words are very similar to the reference. This is similar to the conclusion of Tiedemann (2009) that many errors of a character level phrase-based statistical machine translation for Norwegian and Swedish are of small length. Furthermore, we find that most of the not properly corrected words actually do not have enough representative instances, i.e., most of them occurred only once in the validation data and were not seen during the training. The high sparsity of the data is an interesting challenge for the current neural networks for which more research is needed.

  With the settings of our experiments, the high Recall of the model at a word level indicates that it can be used for detecting errors and words to normalise but not for *automatically* fixing them because of its low Precision. Actually the reported low Precision is not that dramatic as it might seem because it is aggressive, i.e., a single wrong character means the entire word is wrong. Besides improving our inference settings, a better metric for evaluating such cases is needed.

- **Effect** We evaluate the effect of spelling correction and normalisation, as a pre-processing step for downstream tasks, on detecting binary Semantic Textual Similarity. We chose this task because it is one of the few available tasks for ALG we are aware of. We apply our spelling correction and normalisation on the ALG data reported by (Adouane et al., 2019). We replicate the best performing model for which the authors report an accuracy of 92.76%, and we get an accuracy of 94.40% with the same settings. The gain indicates that the spelling correction and normalisation is potentially a useful pre-processing step for downstream tasks.

## 7 Conclusion and Future Work

We compiled a new parallel corpus for ALG with linguistically motivated decisions for spelling correction and normalisation. Considerations such as being practical to implement and suitability for our goals are taken into account. We designed, implemented and tested 2 deep neural network architectures trained end-to-end to capture the knowledge encoded in the corrected and normalised corpus. The results showed that a CNN token-sequence encoder and an LSTM decoder performed the best when including context information. Additionally, applying a token aligner on the input data yielded better performance compared to the non-aligned data. Even though, with the current inference settings, the model generated some errors at a character level mainly due to the data sparsity, it is general and does not require extra resources except a parallel corpus. Hence it could be applied to other languages with the same settings.

In future work, we plan to improve the current inference mode by investigating other settings, improve the decoder by pre-training on the corrected and normalised data and a large MSA corpus to avoid generating incorrect character sequences. Moreover, we will evaluate the model extrinsically by using it to pre-process data for tasks such as code-switch detection, and topic detection to see how much it helps or hinders attempts to tackle these tasks.

## Acknowledgement

The research reported in this paper was supported by a grant from the Swedish Research Council (VR project 2014-39) for the establishment of the Centre for Linguistic Theory and Studies in Probability (CLASP) at the University of Gothenburg. We are grateful to the annotators and all people who helped collecting the data. We also thank the anonymous reviewers for their useful comments.

## References

Wafia Adouane, Jean-Philippe Bernardy, and Simon Dobnik. 2018. Improving Neural Network Performance by Injecting Background Knowledge: Detecting Code-switching and Borrowing in Algerian Texts. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pages 20–28. Association for Computational Linguistics (ACL).

Wafia Adouane, Jean-Philippe Bernardy, and Simon Dobnik. 2019. Neural Models for Detecting Binary Semantic Textual Similarity for Algerian and MSA. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, pages 78–87, Florence, Italy. Association for Computational Linguistics (ACL).

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv*, arXiv:1409.0473.

Marcel Bollmann. 2019. A Large-scale Comparison of Historical Text Normalization Systems. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3885–3898. Association for Computational Linguistics (ACL).

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. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics (ACL).

Gabriel Doyle. 2014. Mapping Dialectal Variation by Querying Social Media. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 98–106. Association for Computational Linguistics (ACL).

Jacob Eisenstein. 2013. Phonological Factors in Social Media Writing. In *Proceedings of the Workshop on Language Analysis in Social Media*, pages 11–19, Atlanta.

Ramy Eskander, Nizar Habash, Owen Rambow, and Nadi Tomeh. 2013. Processing Spontaneous Orthography. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 585–595. Association for Computational Linguistics (ACL).

Pravallika Etoori, Manoj Chinnakotla, and Radhika Mamidi. 2018. Automatic Spelling Correction for Resource-scarce Languages Using Deep Learning. In *Proceedings of Association for Computational Linguistics 2018, Student Research Workshop*, pages 146–152. Association for Computational Linguistics (ACL).

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional Sequence to Sequence Learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252, International Convention Centre, Sydney, Australia. PMLR.

Shaona Ghosh and Per Ola Kristensson. 2017. Neural Networks for Text Correction and Completion in Keyboard Decoding. *arXiv*, arXiv: 1709.06429.

Nizar Habash, Fadhl Eryani, Salam Khalifa, Owen Rambow, Dana Abdulrahim, Alexander Erdmann, Reem Faraj, Wajdi Zaghouani, Houda Bouamor, Nasser Zalmout, Sara Hassan, Faisal Al shargi, Sakhar Alkhereyf, Basma Abdulkareem, Ramy Eskander, Mohammad Salameh, and Hind Saddiki. 2018. Unified Guidelines and Resources for Arabic Dialect Orthography. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Anna Jørgensen, Dirk Hovy, and Anders Søgaard. 2015. Challenges of Studying and Processing Dialects in Social Media. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 9–18. Association for Computational Linguistics (ACL).

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of Tricks for Efficient Text Classification. *arXiv*, arXiv:1607.01759.

James Milroy. 2001. Language Ideologies and the Consequence of Standardization. *Journal of Sociolinguistics*, 5:530 – 555.

Behrang Mohit, Alla Rozovskaya, Nizar Habash, Wajdi Zaghouani, and Ossama Obeid. 2014. The First QALB Shared Task on Automatic Text Correction for Arabic. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Langauge Processing (ANLP)*, pages 39–47, Doha, Qatar.

Eva Pettersson. 2016. Spelling Normalisation and Linguistic Analysis of Historical Text for Information Extraction. *PhD thesis*, Studia Liguistica Upsaliensia 17, Uppsala: Acta Universitatis Upsaliensis.

Houda Saadane and Nizar Habash. 2015. A Conventional Orthography for Algerian Arabic. In *Proceedings of the Second Workshop on Arabic Natural Language Processing, ANLP, ACL 2015, Beijing, China, July 30, 2015*, pages 69–79. Association for Computational Linguistics (ACL).

Khaled Shaalan, Mohammed Attia, Pavel Pecina, Younes Samih, and Josef van Genabith. 2012. Arabic Word Generation and Modelling for Spell Checking. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

S. Sooraj, K. Manjusha, M. Kumar, and K.P. Soman. 2018. Deep Learning Based Spell Checker for Malayalam Language. *Journal of Intelligent & Fuzzy Systems*, 34:1427–1434.

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.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and Tell: A Neural Image Caption Generator. In *Computer Vision and Pattern Recognition*.

Janet C. E. Watson. 2007. *The Phonology and Morphology of Arabic*. The Phonology of the World's Languages. OUP Oxford.