The 2019 Conference on Computational Linguistics and Speech Processing
ROCLING 2019, pp. 102-111
©The Association for Computational Linguistics and Chinese Language Processing

Sequence to Sequence Convolutional Neural Network for Automatic Spelling Correction

Daniel Hládek, Matúš Pleva, Ján Staš,
Department of Electronics and Multimedia Communications
Technical University of Košice, Slovakia
daniel.hladek@tuke.sk,matus.pleva@tuke.sk,jan.stas@tuke.sk

Yuan-Fu Liao
Department of Electronic Engineering
National Taipei University of Technology
yfliao@mail.ntut.edu.tw

Abstract

The paper proposes a system that compensates most of the noise in a text in natural language caused by technical imperfection of the input device such as keyboard or scanner with optical character recognition, quick typing, or writer incompetence. Correcting the spelling errors in the text improves the performance of the following natural language processing. The incorrect sequence of characters is transcribed into another sequence of correct characters by a neural network with encoder-decoder architecture. Our approach to automatic spelling correction considers characters in an erroneous sentence as words of the source languages. The neural network searches for the best sequence of output characters for the given input. The proposed approach for spelling correction does not require any or minimal amount of training data. Instead, the error model is expressed by a simple component that distorts unannotated data and creates any necessary quantity of training examples for a neural network. The experimental results show that the presented approach significantly improves the distorted data (from 50% WER to 0.09% WER) with distortion lower than 1.5% WER.

Keywords: automatic spelling correction, sequence to sequence, encoder-decoder, deep learning.

1. Introduction

Written or scanned text is often not in the intended form. Writer or the input device often generate deviations that make it less understandable. The errors are usually not a problem in casual communication but make machine processing more complicated.

Removal of spelling errors helps with the following processing of the text in natural language. Automatic spelling correction (ASC) is an essential part of the processing of the documents with noisy data in natural language. ASC helps to recover the intended canonical form of the text and improves the quality of the input data for the following natural language processing (NLP) components. It supports processing of digitized documents, automated proofreaders, or information retrieval systems (e.g TREC-5 confusion track [1]). The main motivation for this work is an improvement of the training data for language model for speech recognition [2].

2. State of the art

The task of automatic error-correction is to generate the most likely correct word-for ms given a misspelled word-form [3].

Previous approaches to ASC, such as correcting spelling errors in the Chinese language [4] use classical statistical methods, such as the hidden Markov model, n-gram language model, log-linear regression, or forward-backward algorithm [5].

The usual form of mathematical formalism is a noisy channel proposed by Shannon [6]. Shannon defines the ASC of a possibly incorrect word s as finding the best correction candidate w_b from a list of possible correction candidates $w_i \in W$ (W is a valid word dictionary) with the best unnormalized probability [7]:

$$w_b = \max_{w_i \in C(s)} P(s|w_i P(w_i),$$

The error model $P(s|w_i)$ estimates the probability of unknown string s instead of real word w_i. The error model characterizes the spelling correction problem.

The context model $P(w_i)$ calculated the probability of the correction candidate according to the surrounding words. A finite-state based system, such as Hunspell[1] proposes a list of correction candidates, and a language model helps to choose the best spelling correction candidate.

The task of spelling correction is similar to machine translation (MT). An ASC trans lates input sentence containing spelling errors into another sentence in a "correct" lang uage. Machine translation converts a sequence of words in the source language into another sequence of words in the target language. Formally, MT is the search for the best target seq uence T given source sequence S using model P [8]:

$$T_b = \max_T P(T|S)$$

There are a couple of approaches that used statistical MT for ASC before, such as machine translation spelling for historical texts [9]. [10] attempts to character-level spelling correction.

3. Sequence to sequence spelling correction

Statistical machine translation uses classical methods, such as hidden Markov models, n-gram language models, and sentence alignment [2]. SMT systems have weaknesses that prevent to reach better results. The statistical approaches can calculate only with relatively short contexts (three items in the input sequence maximum).

Neural networks with encoder/decoder architecture brought significant improvement in the performance of SMT. Current deep neural networks [11] can consider a much bro ader context of words or characters. This ability allows us to use an architecture that is based only on neural networks and considers only characters. A neural model can be used to score any given pair of input and output sequences [12].

[1] http://hunspell.github.io/

[2] Moses toolkit

Sequence to sequence neural networks architecture transforms a sequence of symbols from the source language to another sequence of symbols in the target language. Sequences can have different lengths. One symbol is encoded into an n-dimensional binary vector with one dimension for each possible character. The embedding layer reduces the dimension of the input vector. The transformed input matrix has dimension equal to the embedding dimension and size of the sequence. The neural network that transcribes one sequence of symbols into another consists of the encoder and the decoder.

"The encoder maps a variable-length source sentence to a fixed-length vector, and the decoder maps the vector representation back to a variable-length sentence. The two ne tworks are trained jointly to maximize the conditional probability of the target sequen ce given source sequence.". [12]

Knowing the probability of the next symbol enables the decoder to sample probable sequences of symbols. "Sequence to sequence" systems usually use recurrent neural networks (RNN) or convolutional neural networks (CNN) for encoding and decoding.

"The dominant approach to date encodes the input sequence with a series of bi-directional recurrent neural networks (RNN) and generates a variable-length output with another set of decoder RNNs, both of which interface via a soft-attention mechanism." [13]

Recurrent neural networks perform well with tasks with variable-length input and output. Common types of RNN are gated recurrent units (GRU) [14] and long short-term memory (LSTM) [15].

4. The proposed convolutional network architecture

The RNN always maintain a hidden state and updates it with each new item in the input sequence. Compared to RNN, the current state in the input sequence of a convolutional network does not depend on the previous, which makes the computation easier. The processor can compute convolution for the whole sequence at once.

"Convolutions create representations for fixed-size contexts; however, the effective context size of the network can easily be made larger by stacking several layers on top of each other." [13]

The approach uses a convolutional sequence-to-sequence architecture by [13]. The convolutional architecture uses gated linear units (GLU) [16] with residual connections [17]. "The attention mechanism looks at the input sequence and decides at each step which parts are important." The attention mechanism "writes down" quintessential keywords from the sentence. The attention-mechanism considers several other inputs at the same time and decides which ones are important by attributing different weights to those inputs.

The convolutional architecture was selected because recent results [13] show that they offer superior or comparable performance and higher speed of learning when compared to the more-established recurrent networks.

4. Data preparation

Neural networks are sensitive to the amount of training data. Obtaining reasonable precision requires the sufficient size of the training set. Preparation of the data for training of the neural network is difficult, timely, and expensive. Our approach overcomes the problem of data sparsity by rule-based error model that utilizes any unannotated data in the target language and prepares an artificial training set.

A sequence of edit operations describes a spelling error. Usually, the error model considers insertion, deletion, and substitution of characters. A statistical error model is estimated from training data that contain the original and the erroneous strings. An artificial error function randomly modifies some characters in the dataset and creates a distorted string. Example of the training set is in the Table 1.

The training of the neural network uses the distorted string as input and the original string as the output. Training of the network estimates the reverse function and the network can guess the intended form of a distorted string. The error function can generate any amount of training data from a text in natural language.

Table 1Example of the training data

Distorted input	Correct input
faktom vshak oďtava že stavy zamesnancov	faktom však ostáva že stavy zamestnancov

5. Experiments

The proposed neural network needs a sufficiently large text in a natural language for training. We have composed a set of newspaper articles in the Slovak language.

One training sample for the neural network consists of three words. The "clean" text forms the target part of one sample. Table 2 summarizes the size of the text database.

The error model distorts characters in the sample and creates the source part. The distorted and original sequence form a training pair. The neural network learns a function that is inverse to the one that generated the training data.

Table 2 Experimental dataset size

Set	Samples	Words	Characters
train	12 000 000	36 000 000	206 711 896
test	50 000	150 000	873 358

The error model uses the following rules and probabilities:

- Insertion of arbitrary character 0.02
- Deletion of arbitrary character 0.02
- Replacement of arbitrary character 0.08
- Keeping the character 0.9

A forward-backward algorithm by Ristad and Yanilos [18] can estimate parameters of the error model for a set of training examples, which is left for the further research. The ASC system uses Fairseq toolkit [19]. The table 3 summarizes the parameters of the neural network (named fconv_iwslt_de_en in Fairseq toolkit).

Table 3 Neural network architecture

Dropout	0.1
Encoder Embedding Dimension	256
Encoder Layers	256,3 * 4
Decoder Embed Dimension	256
Decoder Layers	256,3 * 3
Decoder Out Embed Dimension	256
Decoder Attention	True

Word error rate is a usual form of evaluation of spelling correction models. Its advantage is that the size of the target and source sequence does not have to be the same. The metric first aligns the sequences with the hypothesis and with the golden truth. The WER is defined as a ratio of the counts of the inserted, deleted, and replaced words:

$$WER = \frac{C_i + C_d + C_s}{C}$$

C_i is number of insertions, C_d number of deletions, C_s is number of substitutions.

Character error rate (CER) is a similar metric but considers characters instead words. Sentence error rate (SER) is ratio of incorrect samples to all samples in the testing set.

The first experiment measures CER, WER, SER of correcting randomly distorted testing text omitted from the training. The Table 4 displays performance of the system after selected iterations (1,5,10,15) of the training of the neural network. The first row (0 - no correction) shows measure of preliminary distortion of the testing set by the error model without any processing. Figure 1 displays the complete learning curve in CER for each training iteration.

Performance of the system is improved only slightly after tenth round of training.

Table 4 Performance of the proposed system

Iteration	CER	WER	SER
0 (no correctoin)	0.1096	0.5173	0.8463
1	0.0386	0.1447	0.3396
5	0.0307	0.1108	0.2677
10	0.0279	0.0998	0.2443
15	0.0273	0.0971	0.2387

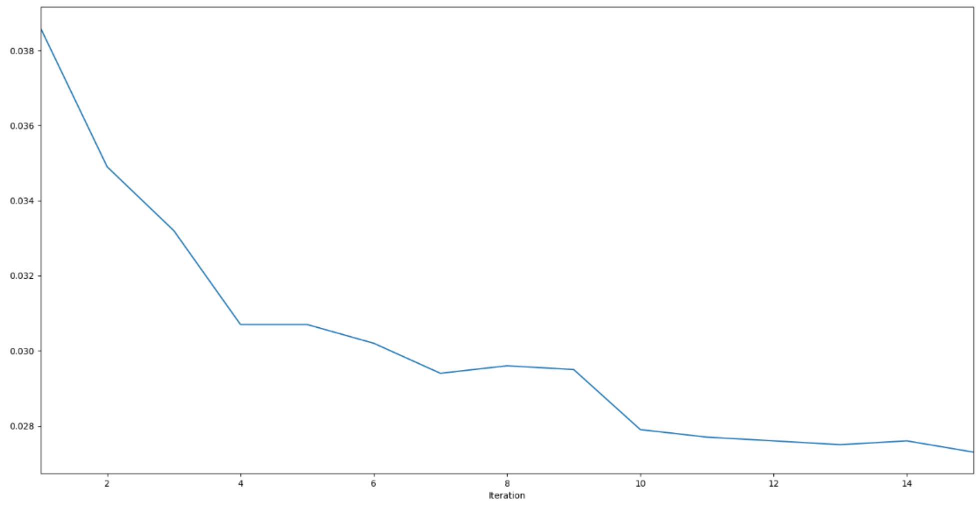

Figure 1 CER Learning Curve

The second experiment measures how the trained neural network damages useful data. Input of the ASC system is a clean text. The Table 4 shows how the neural network distorts the clean data. The distortion of the clean data is very low (0,0022 CER) and decreases with number of training iterations. Distortion CER is marked in the Table 5 for each training round. It shows clear correlation with the learning curve in Figure 1.

Table 5 Distortion on the clean data

Iteration	CER	WER	SER
1	0.00342	0.01594	0.044

5	0.00265	0.01233	0.034
10	0.00254	0.01202	0.033
15	0.00224	0.01029	0.028

6. Conclusion

The experiments confirm that the proposed approach can remove most of the noise from a text in natural language. An expert can design the artificial error model according to the typical error patterns. It is possible to use statistical estimation with relatively small training data, e.g. a letter confusion matrix ([5], [18]). Processing of the clean data has very low distortion and the proposed neural network can be used without damaging the clean data.

Acknowledgements

This work was supported by the Ministry of Education, Science, Research and Sport of the Slovak Republic under research project VEGA 1/0511/17 of the Slovak Scientific Grant Agency, Slovak Research and Development Agency under contract no. APVV SK-TW-2017-0005, APVV-15-0517, APVV-15-0731 and Taiwan Ministry of Science and Technology MOST-SRDA contract no. 108-2911-I-027-501,107-2911-I-027-501,107-2221-E-027-102, 107-3011-F-027-003 and 108-2221-E-027-067.

References

[1] P. B. Kantor and E. M. Voorhees, "The TREC-5 Confusion Track: Comparing Retrieval Methods for Scanned Text," *Inf. Retr. Boston.*, vol. 2, no. 2, pp. 165–76, 2000.

[2] M. Rusko *et al.*, *Advances in the Slovak Judicial domain dictation system*, vol. 9561. 2016.

[3] T. A. Pirinen and K. Lindén, "State-of-the-art in weighted finite-state spell-checking," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014, vol. 8404 LNCS, no. PART 2, pp. 519–532.

[4] Y. Zhang, P. He, W. Xiang, and M. Li, "Discriminative Reranking for Spelling Correction," *Proc. 20th Pacific Asia Conf. Lang. Inf. Comput.*, pp. 64–71, 2007.

[5] D. Hládek, J. Staš, S. Ondáš, J. Juhár, and L. Kovács, "Learning string distance with smoothing for OCR spelling correction," *Multimed. Tools Appl.*, pp. 1–19, 2016.

[6] C. E. Shannon, “A mathematical theory of communication,” *Bell Syst. Tech. J.*, vol. 27, no. 4, pp. 623–56, Oct. 1948.

[7] E. Brill and R. C. Moore, “An improved error model for noisy channel spelling correction,” in *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics ACL 00*, 2000, pp. 286–93.

[8] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to Sequence Learning with Neural Networks,” *Adv. Neural Inf. Process. Syst. 27 (NIPS 2014)*, pp. 3104–3112, 2014.

[9] P. Mitankin, S. Gerdjikov, and S. Mihov, “An Approach to Unsupervised Historical Text Normalisation,” in *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage - DATeCH ’14*, 2014, pp. 29–34.

[10] R. Mihalcea, “Diacritics restoration: learning from letters versus learning from words,” in *CICLing 2002*, vol. 2276, A. Gelbukh, Ed. Springer Berlin Heidelberg, 2002, pp. 96–113.

[11] A. C. Kinaci, “Spelling Correction Using Recurrent Neural Networks and Character Level N-gram,” in *2018 International Conference on Artificial Intelligence and Data Processing, IDAP 2018*, 2019, pp. 1–4.

[12] K. Cho *et al.*, “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)*, 2014, pp. 1724–1734.

[13] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, “Convolutional Sequence to Sequence Learning,” May 2017.

[14] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling,” in *NIPS 2014 Deep Learning and Representation Learning Workshop*, 2014.

[15] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[16] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, “Language Modeling with Gated Convolutional Networks,” Dec. 2016.

[17] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016.

[18] E. S. Ristad and P. N. Yianilos, “Learning string-edit distance,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 5, pp. 522–32, May 1998.

[19] M. Ott *et al.*, “fairseq: A Fast, Extensible Toolkit for Sequence Modeling,” Apr. 2019.