LSTM Networks for Edit Distance Calculation with Exchangeable Dictionaries

Martin Schall*, Haiyan P. Buehrig†, Marc-Peter Schambach‡ and Matthias O. Franz§
*†§Institute for Optical Systems, University of Applied Sciences Konstanz, Germany
*‡Siemens Postal, Parcel & Airport Logistics GmbH, Konstanz, Germany
*Email: martin.schall@htwg-konstanz.de
†Email: haiyan.buehrig@gmail.com
‡Email: marc-peter.schambach@siemens.com
§Email: mfranz@htwg-konstanz.de

***Abstract*—Algorithms for calculating the string edit distance are used in e.g. information retrieval and document analysis systems or for evaluation of text recognizers. Text recognition based on CTC-trained LSTM networks includes a decoding step to produce a string, possibly using a language model, and evaluation using the string edit distance. The decoded string can further be used as a query for database search, e.g. in document retrieval. We propose to closely integrate dictionary search with text recognition to train both combined in a continuous fashion. This work shows that LSTM networks are capable of calculating the string edit distance while allowing for an exchangeable dictionary to separate learned algorithm from data. This could be a step towards integrating text recognition and dictionary search in one deep network.**

I. Introduction

The string edit distance [1] [2] defines a metric of similarity of two strings. It is the minimum number of character insertion, deletion or replacement operations to transform one string into the other. Information retrieval and document analysis systems use the edit distance for e.g. document retrieval or dictionary search. It is also used for evaluating text recognizers by using it as a measure of the character error rate. Use cases are e.g. the search for address elements in postal and parcel processing, the localization of genome sub-sequences or keyword search in web search engines. Optimized index structures can be used when no two arbitrary strings are compared but a query string with a dictionary of reference strings.

Long Short Term Memory *(LSTM)* networks [3] [4] trained with Connectionist Temporal Classification *(CTC)* [5] [6] produce a sequence of character probabilities while transcribing text from images. This probabilistic output is further decoded to one or more strings. Decoded strings are used for evaluation of the network or in following application steps. A language model can be used to improve decoding of the network output.

Transcription, decoding and dictionary search are often seen as separate steps. We propose to integrate these three steps into one deep LSTM network. This work is a step in this direction by showing that LSTM network can learn to calculate the string edit distance of a one-hot coded string and a dictionary of strings. A one-hot coding of strings is very similar to the probabilistic output of a CTC-trained LSTM network, but values are boolean instead of continuous probabilities. Integration of transcription, decoding and dictionary search in one network could reduce the overall error rate by allowing the network to learn domain specific statistics in all three steps. Also moving decoding and dictionary search into a LSTM network could allow speed improvements by moving the execution to a GPU accelerator.

This work uses an English word corpus [7] derived from the Google Trillion Word Corpus [8] in its experiments.

II. Methodology

Strings used in this work are in English language and between 3 and 10 characters in length. The alphabet is 26 characters in size. Each string is represented as a matrix of size 10×26 with individual characters encoded by a one-hot coding, setting one of the 26 coefficients to one and all others to zero. For example the character *A* is encoded as $[1, 0, \ldots, 0]$, *B* as $[0, 1, 0, \ldots, 0]$ and so on. Strings shorter than 10 characters in length are padded with zero coefficients. Strings are processed by the RNN as sequences of 10 length with 26 features per step.

The network takes two separate inputs. One is the encoded representation of the dictionary strings with the strings concatenated along the feature-dimension. This results in an input of size $|batch| \times 10 \times (26 \times |dictionary|)$ for mini-batch training. Dictionaries are 100 strings each in this work and thus the encoded dictionary is $|batch| \times 10 \times 2600$ in size. Second input is the representation of the query strings with $|batch| \times 10 \times 26$ in size.

The RNN consists of multiple bidirectional [9] LSTM layers with the same number of neurons per layer. The networks task is to process the query string and predict the string edit distances to the dictionary strings as a regression problem. The encoded dictionary is provided as input by concatenating it with the BLSTM input along the feature-dimension. This topology is shown in Figure 1.

Output layer of the RNN is fully connected with ReLU [10] non-linearity. This layer consists of one neuron per string of the dictionary, in our case 100 neurons. These neurons predict the string edit distances between the query string and the dictionary strings. String edit distance is zero or positive

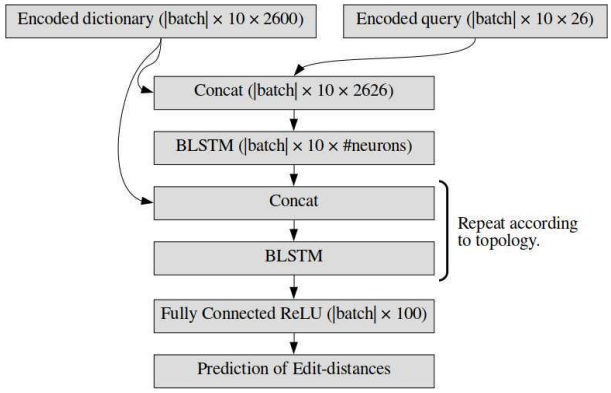

Fig. 1. Network topology for comparing query strings of up to 10 characters length with an alphabet of 26 characters to a dictionary of 100 such strings.

and as such the ReLU non-linearity is capable of predicting it without re-scaling. Loss function for training is the Mean Squared Error *(MSE)* of the predicted and correct string edit distances.

III. Results

Data for training and evaluation was derived from the 20k most frequent English words [7] [8] with a length between 3 and 10 characters, which results in a set of 16968 strings. 1000 of these were used as 10 dictionaries of 100 strings each. 9 dictionaries were for training, the other only for evaluation. A random one of the 9 dictionaries was chosen for each mini-batch during training. 80% of the remaining strings were used as query strings for training and 10% each for validation and evaluation.

Optimization of the network was done using Adam [11] with a learning rate of 0.001 and a mini-batch size of 16. Training was limited to a maximum of 200 epochs. Multiple optimization strategies were evaluated but Adam and mini-batch training produced good and reliable results.

TABLE I
RMSE FOR DIFFERENT NETWORK SIZES WITH UNSHUFFLED DICTIONARIES.

#layers × #neurons	2 × 30	2 × 60	3 × 60	5 × 200
Test set, unkn. dict.	1.78	1.78	1.56	2.13
Validation set, unkn. dict.	1.78	1.80	1.57	2.14
Training set, unkn. dict.	1.78	1.79	1.57	2.12
Test set, known dict.	0.37	0.30	0.29	0.36
Validation set, known dict.	0.37	0.29	0.29	0.36
Training set, known dict.	0.37	0.29	0.28	0.34

Table I shows the Root Mean Squared Error *(RMSE)* for the described network and experiment. The 10 dictionaries were not shuffled in this experiment and thus the strings remained in the same order within each dictionary for the whole training and evaluation. Much lower RMSE values were achieved for the 9 known dictionaries in comparison to the unknown dictionary.

Table II contains RMSE values for the same experimental set-up, but the dictionaries were randomly shuffled and thus the strings were in random order within their dictionary. Shuffling was done for each mini-batch to reduce the risk of repeating the same dictionary order. Results show a much smaller gap in RMSE between the known and unknown dictionaries.

TABLE II
RMSE FOR DIFFERENT NETWORK SIZES WITH SHUFFLED DICTIONARIES.

#layers × #neurons	2 × 30	2 × 60	3 × 60	5 × 200
Test set, unkn. dict.	0.86	0.84	0.84	0.84
Validation set, unkn. dict.	0.86	0.84	0.84	0.84
Training set, unkn. dict.	0.86	0.84	0.84	0.84
Test set, known dict.	0.85	0.82	0.82	0.81
Validation set, known dict.	0.85	0.82	0.82	0.81
Training set, known dict.	0.85	0.82	0.82	0.81

IV. Discussion

The conducted experiments are promising and show that LSTM networks are capable of learning to predict the string edit distance while separating dictionary data from the actual algorithm. The achieved RMSE of ≈ 0.8 is not enough to retrieve the correct distance by rounding. It may still enable the use of such networks for applications like decoding of CTC-trained text recognizers. Further studies are necessary to validate the assumptions made about a close integration of CTC-based text recognition and string edit distance calculation in one network.

Acknowledgment

The authors would like to thank the Siemens Postal, Parcel & Airport Logistics GmbH for funding this work.

References

[1] R. A. Wagner and M. J. Fischer, "The String-to-String Correction Problem," *Journal of the ACM*, vol. 21, no. 1, pp. 168–173, 1974.
[2] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Physics Doklady*, vol. 10, no. 8, pp. 707–710, 1966.
[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory." *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[4] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: continual prediction with LSTM." *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.
[5] A. Graves, S. Fernandez, F. Gomez, and J. Schmidhuber, "Connectionist Temporal Classification : Labelling Unsegmented Sequence Data with Recurrent Neural Networks," in *Proceedings of the 23rd international conference on Machine Learning*. ACM Press, 2006, pp. 369–376.
[6] A. Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer, 2012.
[7] "GitHub: Josh Kaufman," https://github.com/first20hours/google-10000-english, accessed: 2017-09-29.
[8] J.-B. Michel, Y. K. Shen, A. P. Aiden, A. Veres, M. K. Gray, J. P. Pickett, D. Hoiberg, D. Clancy, P. Norvig, J. Orwant *et al.*, "Quantitative analysis of culture using millions of digitized books," *science*, vol. 331, no. 6014, pp. 176–182, 2011.
[9] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *Signal Processing, IEEE Transactions on*, vol. 45, no. 11, pp. 2673–2681, 1997.
[10] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML*, vol. 30, no. 1, 2013.
[11] D. P. Kingma and J. L. Ba, "Adam: a Method for Stochastic Optimization," *International Conference on Learning Representations*, pp. 1–13, 2015.