CARU: A Content-Adaptive Recurrent Unit for the Transition of Hidden State in NLP

Ka-Hou Chan[1,2](✉), Wei Ke[1,2], and Sio-Kei Im[2]

[1] School of Applied Sciences, Macao Polytechnic Institute, Macao, China
{chankahou,wke}@ipm.edu.mo

[2] Macao Polytechnic Institute, Macao, China
marcusim@ipm.edu.mo

Abstract. This article introduces a novel RNN unit inspired by GRU, namely the Content-Adaptive Recurrent Unit (CARU). The design of CARU contains all the features of GRU but requires fewer training parameters. We make use of the concept of weights in our design to analyze the transition of hidden states. At the same time, we also describe how the content adaptive gate handles the received words and alleviates the long-term dependence problem. As a result, the unit can improve the accuracy of the experiments, and the results show that CARU not only has better performance than GRU, but also produces faster training. Moreover, the proposed unit is general and can be applied to all RNN related neural network models.

Keywords: Recurrent neural network · Gate recurrent unit · Long-Short Term Memory · Content-adaptive · Natural Language Processing

1 Introduction

With the rapid development of the Internet and social networks, people can easily view and share information with each others to express their thoughts across the digital networks. Unlike face-to-face conversations, the presence of intermediary media provides a way to collect those typed sentences and text which can be further analyzed in the Natural Language Processing (NLP) tasks and human behavior research. However, the structure of social network data is complex and diverse, and usually contains some cultural information related to the living environment. How to query and confirm relevant and meaningful information from the perceptual data (within image and language) has important research significance.

For the natural language in sentences analysis, there is unstructured characteristics of sequence data. A language expression has diversity and irregularity, and the complexity of semantics makes it difficult for computers to extract the real intention and implication of the expression. NLP researchers aim to gather

H. Yang et al. (Eds.): ICONIP 2020, LNCS 12532, pp. 693–703, 2020.
https://doi.org/10.1007/978-3-030-63830-6_58

knowledge about how human beings understand and use languages in order to develop appropriate methods and techniques to allow computer systems to understand and interpret natural languages to perform further tasks. However, traditional NLP methods require a lot of feature extraction and expression in a sentence [13,28]. How to learn the meaning of perceptual data to determine the morphological structure and the grammar of an entire sentence, to express the natural information of text, and to improve the performance of the emotional classification are problems to be solved in text analysis. However, the development of cross-domain (linguistics and computer science) methods will always lag behind the evolution of modern languages.

In recent years, with the development of machine learning for NLP [7], through the neural network models, expressions can be studied through a large number of training parameters without manual design. This is convenient in practice with theory and evidence [1,23]. The neural networks can approximate any unknown expressions through a combination of nonlinear functions and specific layers [21], and these NLP problems can be well studied in the context of recurrent neural networks (RNNs) [14,16,26]. Exploring the RNN models, it can be found that there are some layers being used repeatedly, with each layer consisting of various RNN units. The Elman network [6] is also known as a simple recurrent networks model whose units perform linear operations. Specifically, the sequence of tensor $\left\{v_{m\times1}^{(1)}, v_{m\times1}^{(2)}, \ldots, v_{m\times1}^{(l)}\right\}$, each of size $(m \times 1)$ as features, is input to the RNN, then produces the last hidden state $h_{n\times1}^{(l)}$, and the hidden state results:

$$h_{n\times1}^{(t+1)} = \boldsymbol{w_{n\times n}} h_{n\times1}^{(t)} + \boldsymbol{w_{n\times m}} v_{m\times1}^{(t)} \tag{1}$$

where $h_{n\times1}^{(t)}$ is the t-th hidden state of size $(n \times 1)$. $\boldsymbol{w} = [\boldsymbol{w_{n\times n}} | \boldsymbol{w_{n\times m}}]$ is the weight for training use[1]. The input tensors can be projected onto

$$\left\{x_{n\times1}^{(1)}, x_{n\times1}^{(2)}, \ldots, x_{n\times1}^{(l)}\right\},$$

when $x_{n\times1} = \boldsymbol{w_{n\times m}} v_{m\times1}$. Therefore, the expansion form of (1) becomes

$$h_{n\times1}^{(t+1)} = \boldsymbol{w_{n\times n}^{t}} x_{n\times1}^{(1)} + \boldsymbol{w_{n\times n}^{t-1}} x_{n\times1}^{(2)} + \cdots + \boldsymbol{w_{n\times n}^{0}} x_{n\times1}^{(t)} \tag{2}$$

where $\boldsymbol{w_{n\times n}^{0}}$ is an identity matrix I_n. (2) can be seen as a system with radix $\boldsymbol{w_{n\times n}}$, which is a non-integer square matrix. The current hidden state $h^{(t)}$ can uniquely correspond to the sequence of the received tensors $v^{(\leq t)}$, implying their feature and order, thus RNNs can overcome the dynamic length data, allowing one to only consider the last hidden state.

[1] For the complete hidden state, (1) should include the bias parameter as $h_{n\times1}^{(t+1)} = \boldsymbol{w_{n\times n}} h_{n\times1}^{(t)} + \boldsymbol{b_{n\times1}} + \boldsymbol{w_{n\times m}} v_{m\times1}^{(t)} + \boldsymbol{b_{n\times1}}$, and followed by a non-linear activation function $\tanh\left(h^{(t+1)}\right)$ that is to prevent divergence during training. To facilitate derivation, we ignore them in this section.

1.1 Related Work

In recent years, RNN has become the mainstream model for NLP tasks. It can be used to represent language expressions with different cultures through unified embedded vectors. Analysis of these word vectors through neural network models can obtain higher-level information. However, the (2) reflects that each input tensor has the same weight, but this feature conflicts with the importance of various words, since a word may have other meanings or different importance in various combinations. Especially the punctuation, preposition and postposition is often less important than other words, and the main information will always be diluted if the above model is used. Fortunately, this problem can be significantly alleviated by the Long-Short-Term-Memory (LSTM) networks [8,9,11] and its variant, known as the Gated Recurrent Unit (GRU) [3]. These are designed to avoid the long-term dependence of less important information. Instead of overwriting previous hidden states, they use various gate units to adjust the ratio of the updated state to alleviate word weighting conflicts. Later, some simplified gated units have been proposed but the performance has not been improved [10]. For the simple NLP problem, LSTM-like method can achieve better accuracy. Their hidden states can be used in classification problems for further purposes, such as parts-of-speech tagging [17,25] and sentiment analysis [20,27].

In addition, the last hidden state can also be regarded as an encoding result, which can be further decoded by another radix. In this case, the coding result must contain all words in the source sentence, but the weights are difference [12]. The dynamic coming along with the ratio of word weighting is well compatible by the Seq2Seq model [22]. Its primary components are one encoder network and one decoder network (both of them are RNNs), and the decoder reverses the encoding process [2]. Furthermore, ideas from RNN architectures that improve the ease of training, such as that the decoder accepts all hidden states which stores the entire context and allows the decoder to look at the input sequence selectively as with the attention mechanisms [24]. In view of these architectures, researchers can propose novel RNN units for various tasks. [4] introduces an Update Gate RNN (UGRNN) that is similar to the Minimal Gated Unit (MGU) [29] which is directly inspired by the GRU. Of course, their variants may work well in some cases, but there is no consensus on the best and comprehensive LSTM-like architecture.

In this work, we propose a new RNN unit inspired by GRU, that contains the linear update unit like (2) and the transition ratio of the hidden state will be completed by interpolation. In particular, we intend to remove the reset gate from the GRU and replace its function with our proposed adaptive gate. In our architecture, the GRU update gate and the proposed adaptive gate form the content adaptive gates, thus we name the proposed unit as the Content-Adaptive Recurrent Unit (CARU). Meanwhile, the proposed method has the smallest possible number of training parameters (equivalent to MGU) in GRU-like units, and has met the requirements of various aspects as much as possible.

2 Content-Adaptive Recurrent Unit

For the context of perceptual data, human beings can capture the meaning or patterns immediately if they have learned it before. Corresponding to NLP, for a standard sentence, its subject, verb and object should be paid more attention before considering other auxiliary information. This process can be seen as a weighted assignment of words in a sentence, which is related to the tagging task in NLP. However, weighting by using tagging is not suitable for non-standardized sequence data, such as phrases or casual chats. In order to overcome this weighting problem, we first analyze the architecture of GRU:

$$r^{(t)} = \sigma\left(\boldsymbol{w_{hr}} h^{(t)} + \boldsymbol{b_{hr}} + \boldsymbol{w_{vr}} v^{(t)} + \boldsymbol{b_{vr}}\right) \tag{3a}$$

$$n^{(t)} = \tanh\left(r^{(t)} \odot \left(\boldsymbol{w_{hn}} h^{(t)} + \boldsymbol{b_{hn}}\right) + \boldsymbol{w_{vn}} v^{(t)} + \boldsymbol{b_{vn}}\right) \tag{3b}$$

$$z^{(t)} = \sigma\left(\boldsymbol{w_{hz}} h^{(t)} + \boldsymbol{b_{hz}} + \boldsymbol{w_{vz}} v^{(t)} + \boldsymbol{b_{vz}}\right) \tag{3c}$$

$$h^{(t+1)} = \left(1 - z^{(t)}\right) \odot n^{(t)} + z^{(t)} \odot h^{(t)} \tag{3d}$$

where $r^{(t)}$ and $z^{(t)}$ is the result of reset gate and update gate, $n^{(t)}$ is a variant of the next hidden state. It presents that the current hidden state $h^{(t)}$ is processed by the Hadamard operation ($\odot$) with $r^{(t)}$ in (3b), that controls the amount of history used to update the candidate states. It would effectively become only dependent on the current word $v^{(t)}$ when $r^{(t)}$ is close to zero, thus the GRU reset gate has achieved the dynamic weighting of previous hidden states. Furthermore, the result of the GRU update gate in (3d) makes the gradual transition of the hidden states instead of overwriting previous states.

Nevertheless in (3a) and (3b), the GRU allows the result of the reset gate to weaken the hidden state, thereby alleviating the dependence on long-term content, but this approach can further dilutes the information in the entire sentence. It is still challenging for compound-complex sentence like relative-clauses. Therefore, our design does not intend to directly interfere with any current hidden state, as shown below briefly:

$$x^{(t)} = \boldsymbol{w_{vn}} v^{(t)} + \boldsymbol{b_{vn}} \tag{4a}$$

$$n^{(t)} = \tanh\left(\left(\boldsymbol{w_{hn}} h^{(t)} + \boldsymbol{b_{hn}}\right) + x^{(t)}\right) \tag{4b}$$

$$z^{(t)} = \sigma\left(\boldsymbol{w_{hz}} h^{(t)} + \boldsymbol{b_{hz}} + \boldsymbol{w_{vz}} v^{(t)} + \boldsymbol{b_{vz}}\right) \tag{4c}$$

$$l^{(t)} = \sigma\left(x^{(t)}\right) \odot z^{(t)} \tag{4d}$$

$$h^{(t+1)} = \left(1 - l^{(t)}\right) \odot h^{(t)} + l^{(t)} \odot n^{(t)} \tag{4e}$$

(4a) It first project the current word $v^{(t)}$ into $x^{(t)}$ as the input feature. This result would be used in the next hidden state and passed to the proposed content-adaptive gate.

(4b) The reset gate has been taken out. It just combines the parameters related to $h^{(t)}$ and $x^{(t)}$ to produce a new hidden state $n^{(t)}$. So far, it is the same as a simple RNN unit like (1).
(4c) It is the same as the update gate in GRU and is used to the transition of the hidden state.
(4d) There is Hadamard operator to combine the update gate with the weight of current feature. We name this gate the content-adaptive gate, which will influence the amount of gradual transition, rather than diluting the current hidden state (See more details below).
(4e) The next hidden state is combined with $h^{(t)}$ and $n^{(t)}$.

According to (4a) to (4e), the complete diagram of our proposed architecture is shown in Fig. 1.

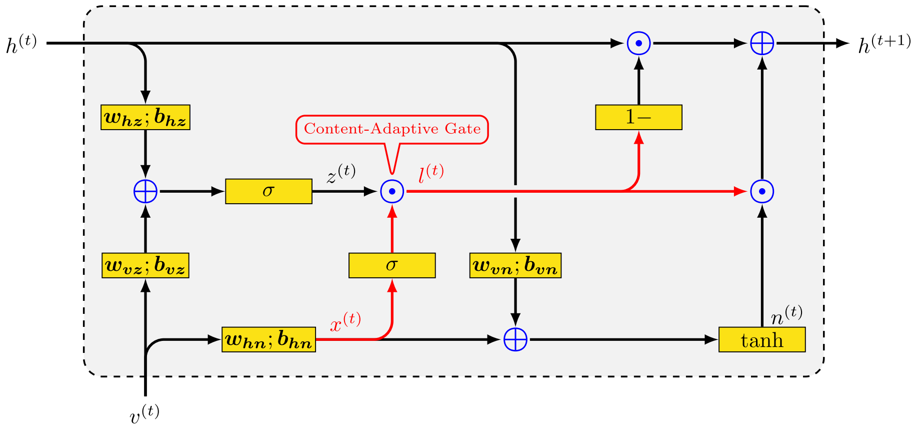
$h^{(t)}$
$h^{(t+1)}$
$\boldsymbol{w_{hz}}; \boldsymbol{b_{hz}}$
Content-Adaptive Gate
$1-$
σ
$z^{(t)}$
$l^{(t)}$
σ
$\boldsymbol{w_{vz}}; \boldsymbol{b_{vz}}$
$\boldsymbol{w_{vn}}; \boldsymbol{b_{vn}}$
$\boldsymbol{w_{hn}}; \boldsymbol{b_{hn}}$
$x^{(t)}$
$n^{(t)}$
tanh
$v^{(t)}$

Fig. 1. The proposed CARU architecture. The direction of data flow is indicated by arrows, involved functions are indicated by yellow rectangles, and various gates (operations) are indicated by blue circles. Our approach had drawn in red color corresponding to (4a) and (4d). (Color figure online)

As illustrated in Fig. 1, functions like $(\boldsymbol{w}; \boldsymbol{b})$ is the linear function like (4a), and there are four sets of linear function training parameters. The tanh and σ are the sigmoid activation function within range $(-1.0, +1.0)$ and $(0.0, 1.0)$ respectively, where tanh prevents divergence during training, and the result of σ is used as the reference amount in the following operation.

2.1 Data Flow of Hidden State and Weight

As mentioned in (3b) from GRU, the hidden state is weighted by $r^{(t)}$ according to the current content. Inspired by this, our method further weights the hidden state according to the current word, and introduces a content-adaptive gate instead of using the reset gate to alleviate the dependence on long-term content. Thus we dispatch these processes into three trunk of data flows:

content-state It produces a new hidden state $n^{(t)}$ achieved by (4b), this part is equivalent to simple recurrent networks.

word-weight It produces the weight $\sigma\left(x^{(t)}\right)$ of the current word, it has the capability like a GRU reset gate but is only based on the current word instead of the entire content. More specifically, it can be considered as the tagging task that connects the relation between the weight and parts-of-speech.

content-weight It produces the weight $z^{(t)}$ of the current content, the form is the same as a GRU update gate but with the purpose to overcome the long-term dependence.

Comparing to GRU, we do not intend to process those data flow like (3b), but dispatch the word-weight to the proposed gate and multiply it by the content-weight. In such a way, the content-adaptive gate takes into account of both the word and the content.

2.2 Transition of Hidden State

As mentioned above, the linear recurrent unit as (2) is an extreme case, which reflects that each input word also has the most important meaning (weight is one), thus the new hidden state $n^{(t)}$ can be considered as this extreme case. In the opposite extreme case, the input word has no meaning (weight is zero) in the entire sentence. Once we clamp the range, the neural network will learn to find a suitable amount between these two extreme cases and then use it for the transition. This can be achieved by using the last step (linear interpolation) in GRU, but in order to meet the condition of the extreme cases, we switch the $n^{(t)}$ and $h^{(t)}$ from (3d) to (4e). For instance, the word-weight $\sigma\left(x^{(t)}\right)$ should be close to zero if the current word is a "full stop", implied as:

$$\begin{aligned}\sigma\left(x^{(t)}\right) \to 0 &\implies \sigma\left(x^{(t)}\right) \odot z^{(t)} = l^{(t)} \to 0 \\ &\implies \left(1 - l^{(t)}\right) \odot h^{(t)} + l^{(t)} \odot n^{(t)} = h^{(t+1)} \to h^{(t)}\end{aligned}$$

where no matter how the content-weight $z^{(t)}$ is, the result of the content-adaptive gate $l^{(t)}$ will be close to zero, meaning that the next hidden state $h^{(t+1)}$ will also tend to the current one $h^{(t)}$. It can be seen that low-weight words have less influence to the hidden state. Similarly, the long-term dependence problem can also be alleviated by the implication of the content-weight $z^{(t)}$.

Overall speaking, the proposed CARU can be summarized in two parts: The first part involves all the training parameters (used for linear function $(\boldsymbol{w}; \boldsymbol{b})$), and focuses on the calculation of the state and weight, which can be done independently. The second part focuses on the transition of the hidden state, starting from the content-adaptive gate, it combines all the obtained weights and the current state then produces the next hidden state directly. Since CARU contains the factor and concept of word tagging, it can perform well in the work of NLP tasks.

3 Experimental Results

In this section, we evaluate the performance of CARU for NLP tasks. There are IMDB [15] dataset and Multi30k [5] dataset for sentiment analysis and machine translation problems, respectively. Because the proposed unit has fewer parameters than GRU, the training period requires less time. We use a simple model as the baseline architecture (no additional layers or activation functions). In order to compare the proposed CARU with GRU and MGU, we have implemented our design and re-implemented the MGU unit by PyTorch, which is a scientific deep learning framework [18]. All our experiments were conducted on an NVIDIA GeForce GTX 1080 with 8.0 GB of video memory. We evaluated the performance on the same environment with and without the use of the proposed unit, and with the same configuration and the same number of neural nodes. We only changed the RNN unit without applying other optimization techniques.

3.1 Sentiment Analysis

The first experiment is about sentiment analysis of IMDB movie reviews, that contains 25,000 movie reviews for training, another 25,000 for testing. Our task is to determine the positive and negative comments according to the content of the paragraph. Since the sequence length provided is not fixed and the maximum length is of thousands, it can be handled by a simple recurrent model. In this way, we apply various RNN units to compare their accuracy on the test set. Moreover, we also make use of a pre-trained word embedding [19] in order to convert words to vectors as input feature of the RNN unit, and all units process a size of 100 input and 256 output. The batch size is 100 (with shuffling), and we use the Adam optimizer and initialize the learning rate to 0.001. Similar to other classification problems, we apply fully connect layer to the last hidden state to classify movie reviews as positive or negative according to the probability calculated from a *Softmax* function, also we cooperate with cross-entropy function during training.

As indicated in Fig. 2, we find out the test accuracy after 100 epochs. Our proposed unit has improved the performance. In addition, we compare the training time of these units. We know that three sets of linear function parameters are required in GRU, but CARU and MGU requires two sets. Therefore we can roughly estimate that the time required for GRU is 1.5 times that of ours. Meanwhile, the convergence of our proposed unit is faster than the other units. It also performed with better stability in all experiments, and the difference of results is very small. This advantage in the training time and parameters is most obvious for the machine translation tasks.

3.2 Neural Machine Translation

The second experiment is the neural machine translation of Multi30k, that contains 30,000 images originally described in English, a collection of professionally translated German sentences, and another of crowdsourced German descriptions. Beyond other NLP tasks, there is no established answer in translation

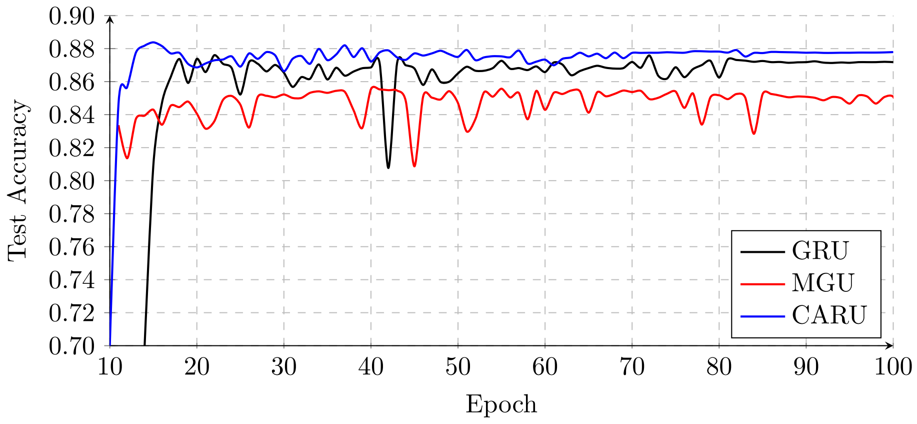

Fig. 2. Test accuracy rates for IMDB test set.

problem. It requires comprehension and representation of the source and target languages, respectively. For this task, we use the Seq2Seq model that can meet these requirements well, which is a type of Encoder-Decoder model using RNN. In fact, its Encoder and Decoder are two independent simple recursive models, which can directly correspond to comprehension and representation. Therefore, we can compare the performance of translation by using various RNN units (English to German, and reversed). Because there are two different languages, a huge vocabulary is required for training, all RNN units must increase the size to 256 input and 2048 output, other settings are the same as previous.

Table 1. BLEU-4 scores of translation results on the Multi30k development set.

	GRU	MGU	CARU
German to English	0.2793 ± 0.0074	0.2651 ± 0.0173	0.3019 ± 0.0022
English to German	0.2852 ± 0.0042	0.2892 ± 0.0070	0.3008 ± 0.0020

Table 2. BLEU-4 scores of translation results on the Multi30k test set.

	GRU	MGU	CARU
German to English	0.2921 ± 0.0033	0.2849 ± 0.0184	0.3199 ± 0.0041
English to German	0.3020 ± 0.0033	0.3123 ± 0.0062	0.3257 ± 0.0015

As indicated in Table 1 and Table 2, the scores by the CARU is found to improve the overall translation performance in terms of BLEU-4 scores with multiple runs. The most obvious improvement is German to English, by a gap of 0.03. As expected, both of CARU and MGU took about 20 to 25 min to complete a training epoch, and GRU took more than half an hour for this. Furthermore, no matter which RNN unit we use during training, the convergence often stagnates after completing several epochs. Whenever we encounter this issue, we will halve the learning rate and the convergence will start to improve again.

In all the experiments, our proposed unit performs well in sentence-based NLP tasks, but has no advantage in handling paragraphs or articles. The reason for this problem is that weighted words are active throughout the sequence, but when entering the entire paragraph, the active-weighted words may mislead other unrelated sentences. Fortunately, this issue can be alleviated by the sentence tokenization during pre-processing. In addition, our experiment also requires more pre-processing and post-processing, the reason for that is less relevant and we are not going into the detail. Please find the complete source code on following this link:

`github.com/ChanKaHou/CARU`

4 Conclusion

In this article, we propose a new RNN unit for NLP tasks. The proposed Content-Adaptive Recurrent Unit (CARU) clearly illustrates the content and weighted data flow and the gates used, making the model easier to design and understand. It requires fewer training parameters than GRU, which means it requires less memory and less training time. We compared CARU with MGU and GRU on two datasets that process sequence data in various NLP tasks. CARU achieved better results and faster training time than others.

However, all RNN units so far can only take into account the current and previous contents of sequence data. If we want to consider the following content to achieve complete sentence understanding, it can only be achieved by the network model, which is beyond the capabilities of the RNN unit. For the future work, the proposed unit has great potential for further improvement and analysis, such as answering questions and even being used in other non-NLP tasks or any neural network involving RNN units.

Acknowledgment. The article is part of the research project funded by The Science and Technology Development Fund, Macau SAR (File no. 0001/2018/AFJ).

References

1. Bianchini, M., Scarselli, F.: On the complexity of neural network classifiers: a comparison between shallow and deep architectures. IEEE Trans. Neural Netw. Learn. Syst. **25**(8), 1553–1565 (2014)
2. Cho, K., van Merrienboer, B., Bahdanau, D., Bengio, Y.: On the properties of neural machine translation: encoder-decoder approaches. In: SSST@EMNLP, pp. 103–111. Association for Computational Linguistics (2014)

3. Cho, K., et al.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: EMNLP, pp. 1724–1734. ACL (2014)
4. Collins, J., Sohl-Dickstein, J., Sussillo, D.: Capacity and trainability in recurrent neural networks. In: ICLR (Poster). OpenReview.net (2017)
5. Elliott, D., Frank, S., Sima'an, K., Specia, L.: Multi30k: multilingual English-German image descriptions. In: VL@ACL. The Association for Computer Linguistics (2016)
6. Elman, J.L.: Finding structure in time. Cogn. Sci. **14**(2), 179–211 (1990)
7. François, T., Miltsakaki, E.: Do NLP and machine learning improve traditional readability formulas? In: PITR@NAACL-HLT, pp. 49–57. Association for Computational Linguistics (2012)
8. Gers, F.A., Schmidhuber, J.: LSTM recurrent networks learn simple context-free and context-sensitive languages. IEEE Trans. Neural Netw. **12**(6), 1333–1340 (2001)
9. Gers, F.A., Schmidhuber, J., Cummins, F.A.: Learning to forget: continual prediction with LSTM. Neural Comput. **12**(10), 2451–2471 (2000)
10. Heck, J.C., Salem, F.M.: Simplified minimal gated unit variations for recurrent neural networks. In: MWSCAS, pp. 1593–1596. IEEE (2017)
11. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
12. Kalchbrenner, N., Blunsom, P.: Recurrent continuous translation models. In: EMNLP, pp. 1700–1709. ACL (2013)
13. Kim, S., Seo, H., Rim, H.: Information retrieval using word senses: root sense tagging approach. In: SIGIR, pp. 258–265. ACM (2004)
14. Lopez, M.M., Kalita, J.: Deep learning applied to NLP. CoRR abs/1703.03091 (2017)
15. Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C.: Learning word vectors for sentiment analysis. In: ACL, pp. 142–150. The Association for Computer Linguistics (2011)
16. Mikolov, T., Kombrink, S., Burget, L., Cernocký, J., Khudanpur, S.: Extensions of recurrent neural network language model. In: ICASSP, pp. 5528–5531. IEEE (2011)
17. Nguyen, D.Q., Dras, M., Johnson, M.: A novel neural network model for joint POS tagging and graph-based dependency parsing. In: CoNLL Shared Task (2), pp. 134–142. Association for Computational Linguistics (2017)
18. Paszke, A., et al.: Automatic differentiation in Pytorch (2017)
19. Pennington, J., Socher, R., Manning, C.D.: Glove: global vectors for word representation. In: EMNLP, pp. 1532–1543. ACL (2014)
20. Poria, S., Cambria, E., Gelbukh, A.F.: Aspect extraction for opinion mining with a deep convolutional neural network. Knowl. Based Syst. **108**, 42–49 (2016)
21. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323**(6088), 533–536 (1986). https://doi.org/10.1038/323533a0
22. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: NIPS, pp. 3104–3112 (2014)
23. Szegedy, C., et al.: Going deeper with convolutions. In: CVPR, pp. 1–9. IEEE Computer Society (2015)
24. Vaswani, A., et al.: Attention is all you need. In: NIPS, pp. 5998–6008 (2017)
25. Wang, P., Qian, Y., Soong, F.K., He, L., Zhao, H.: A unified tagging solution: bidirectional LSTM recurrent neural network with word embedding. CoRR abs/1511.00215 (2015)

26. Werbos, P.J.: Generalization of backpropagation with application to a recurrent gas market model. Neural Netw. **1**(4), 339–356 (1988)
27. Zhang, L., Wang, S., Liu, B.: Deep learning for sentiment analysis: a survey. Wiley Interdiscip. Rev. Data Min. Knowl. Discov. **8**(4) (2018)
28. Zhang, Y., Clark, S.: Joint word segmentation and POS tagging using a single perceptron. In: ACL, pp. 888–896. The Association for Computer Linguistics (2008)
29. Zhou, G.-B., Wu, J., Zhang, C.-L., Zhou, Z.-H.: Minimal gated unit for recurrent neural networks. Int. J. Autom. Comput. **13**(3), 226–234 (2016). https://doi.org/10.1007/s11633-016-1006-2