Echo State Networks for Named Entity Recognition

Rajkumar Ramamurthy[1,2,3](✉), Robin Stenzel[1,2,3], Rafet Sifa[1,2,3],
Anna Ladi[1,2,3], and Christian Bauckhage[1,2,3]

[1] Fraunhofer Center for Machine Learning, Sankt Augustin, Germany
[2] Fraunhofer IAIS, Sankt Augustin, Germany
{rajkumar.ramamurthy,marc.robin.stenzel,rafet.sifa,
anna.ladi,christian.bauckhage}@iais.fraunhofer.de
[3] University of Bonn, Bonn, Germany

Abstract. This paper explores a simple method for obtaining contextual word representations. Recently, it was shown that random sentence representations obtained from echo state networks (ESNs) were able to achieve near state-of-the-art results in several sequence classification tasks. We explore a similar direction while considering a sequence labeling task specifically named entity recognition (NER). The idea is to simply use reservoir states of an ESN as contextual word embeddings by passing pre-trained word-embeddings as its input. Experimental results show that our approach achieves competitive results in terms of accuracy and faster training times when compared to state-of-the-art methods. In addition, we provide an empirical evaluation of hyper-parameters that influence this performance.

Keywords: Echo state networks · Recurrent Neural Networks · Named entity recognition · Natural language processing

1 Introduction

Natural language processing (NLP) comprises a broad spectrum of tasks such as sentence labeling [2,31], question answering [24], sequence-to-sequence learning [28], natural language interference [3] etc. One of the core components crucial to all these tasks is obtaining an appropriate representation of text. For instance, in a sequence tagging task, each word in a sentence is assigned a linguistic tag (eg. a named entity or a part of speech). In order to be processed by supervised machine learning models, each word is represented by a real-valued vector that encodes both context and semantics. Earlier systems considered word representations that are obtained using latent-semantic analysis methods such as Singular Value Decomposition [6] or GloVe [22] that uses word co-occurrences and context-window based word representations such as Word2Vec [19]. Although effective,

R. Ramamurthy and R. Stenzel—Contributed equally.

I. V. Tetko et al. (Eds.): ICANN 2019, LNCS 11731, pp. 110–120, 2019.
https://doi.org/10.1007/978-3-030-30493-5_11

these methods are not robust against morphological variations or misspelled words. Tackling this and extending on Word2Vec, FastText [13] represents each word with a bag of character n-grams. Following this trend, recent methods such as ELMo [23] and Flair [1] combine character-level language models and deep recurrent architectures to obtain contextual word representations and have become state-of-the-art methods.

In a sequence classification task, too, the task boils down to obtaining compact representations of sentences. While classic methods using bags-of-words (BOW) or term frequency-inverse document frequency (TF-IDF) are simple and effective, they ignore word ordering and suffer from high dimensionality. In recent years, several methods [14, 27] have been developed to learn compositional operators that convert word representations to sentence representations using different neural architectures. One drawback of these approaches is that they are trained for a particular task and require supervised learning. However, it is desired to obtain task agnostic representations that can be shared across a wide range of tasks. An approach tailored to this is SkipThought [15] which is similar to Skip-gram architecture of Word2Vec but encode sentences directly. Similarly, InferSent [5] learns a generic representation which is trained on a standard natural language inference task yields state-of-the-art results on several tasks.

While the majority of representation learning methods nowadays are driven by choosing methods that vary the encoder architecture or the type of network (RNNs, CNNs), there has been an alternative line of work [32] that caught attention recently. These methods do not train any sentence encoders explicitly; rather they use pre-trained word embeddings as inputs to randomized recurrent neural networks such as bi-directional long short term memory (BiLSTMs) and echo state networks (ESNs). They showed that state-of-the-art sentence encoders do not significantly improve performance over random encoders.

Inspired by these works, we explore contextualized word representations using echo state networks. In particular, we propose to solve a sequence labeling task namely named entity recognition (NER). Current approaches train a bi-directional LSTM which takes pre-trained word embeddings as its inputs to train a contextual word representation for a NER classifier. Instead, we propose to use echo state networks which provide a random context for its input *without* training any of the recurrent connections. Therefore, our main contribution is evaluating echo state network-based random contextual encoder and look how close they can match the performance of trained contextual encoders.

Providing random context has been studied under the title "reservoir computing". Echo state networks which follow this paradigm, have been successfully applied to provide context in several sequential tasks; for neural cryptography to memorize sequences [25], to learn policies in reinforcement learning [26] and time series prediction [11, 16]. Extending this to deep architectures, deep ESNs [9] have been applied to detect Parkinson's disease. However, its application in natural language processing is relatively an unexplored area, and is limited to learn grammatical structures [10, 30] and systematicity in natural language [8]. Differing from these methods, our approach focuses on obtaining contextual word representations to be used in NER which, to the best of our knowledge, has not been studied with echo state networks.

2 Preliminaries

2.1 Named Entity Recognition

Named entity recognition (NER) is the task of detecting named entities in text and assigning them an appropriate label such as organization, location or person. For example: "[Jane Green]$_{PER}$ became the youngest director of the world leading sports equipment manufacturer [ActPro Equipment]$_{ORG}$". More generally NER can be typically considered as a sequence labeling problem, where words or characters in a sequence must be classified to one of the predefined classes. Detecting named entities can either be a standalone task (e.g for anonymizing sensitive entities in documents) or part of a text pre-processing pipeline.

Early NER systems relied heavily on domain-specific knowledge in the form of lexicons and simple hand-crafted rules and features; for instance, the morphology of the word, trigger context words and term frequency. Further development on feature-engineering systems, included the replacement of hand-crafted rules with supervised machine learning models. Additionally, more sophisticated features were used, such as part-of-speech tagging, word embedding, context features. For a detailed review of these earlier approaches, the reader is referred to [20].

One of the limitations of feature-engineering models is that they rely on a domain expert, either to hand-craft the rules or to define meaningful features for a specific application. In recent years, research has moved away from this paradigm, towards NN-based feature inferring systems, as showcased in a recent review [33]. Most of the recently developed NER systems consist of the same core, namely a bi-directional long short-term memory (LSTM). The LSTM takes a sequence of word embeddings as its input and returns a sequence of contextual word embeddings by encoding them into its sentence context. This contextual word embedding is obtained by concatenating the LSTM's hidden state of a word for both directions. The LSTM is often coupled with a Conditional Random Field (CRF), a probabilistic method that can predict labels of sequences by taking the neighboring labels into account [29].

2.2 Echo State Networks

Recurrent Neural Networks are a very powerful tool in NLP. However, both training and parameter tuning of an RNN can be cumbersome, due to the size and the connectivity of the network [7,21]. This problem can be addressed by an alternative paradigm, reservoir computing. Reservoir computing is based on the notion of an interconnected randomly generated network that processes sequential data. This concept is the core of the Echo State Network (ESN), introduced by Jäger [12].

An ESN consists of a reservoir of recurrently and sparsely connected nodes, which are then connected to an output layer, responsible for transforming the response of the network to the appropriate target output. The basic benefit of ESNs is that only the output layer of the network needs to be trained, while the reservoir essentially serves as a context-aware, non-linear mapping of the input data to a high dimensional space.

For an ESN with n_{in} input nodes, n_{out} output nodes and n_{res} reservoir nodes, we can define three sets of weights: the input weights $\boldsymbol{W}^{in} \in \mathbb{R}^{n_{res} \times n_{in}}$ that connect the input nodes to the reservoir, the reservoir weights $\boldsymbol{W}^{res} \in \mathbb{R}^{n_{res} \times n_{res}}$ that interconnect the reservoir nodes and the output weights $\boldsymbol{W}^{out} \in \mathbb{R}^{n_{out} \times n_{res}}$ that connect the reservoir to the output nodes.

At time step t, let $\boldsymbol{x}_t \in \mathbb{R}^{n_{in}}$, $\boldsymbol{r}_t \in \mathbb{R}^{n_{res}}$ and $\boldsymbol{y}_t \in \mathbb{R}^{n_{out}}$ be the state of the input, reservoir and output neurons respectively. The update of the network is then as follows:

$$\boldsymbol{r}_t = (1-\alpha)\boldsymbol{r}_{t-1} + \alpha f_{res}(\boldsymbol{W}^{res}\boldsymbol{r}_{t-1} + \boldsymbol{W}^{in}\boldsymbol{x}_t) \tag{1}$$

$$\boldsymbol{y}_t = f_{out}(\boldsymbol{W}^{out}\boldsymbol{r}_t), \tag{2}$$

where $f_{res}(.)$ and $f_{out}(.)$ are component-wise activation functions and $\alpha \in [0,1]$ is the leaking rate. For $f_{res}(.)$ typically a sigmoidal function is chosen for the reservoir neurons, while for the output layer the selection is task dependent; usually, a linear or softmax function is used. The leaking rate has to be tuned for the given task, as it relates to the required memory of the network [17].

As discussed earlier, the generation of the reservoir and input weight matrices is random. However, it is controlled by certain parameters, such as the distribution from which the weights are chosen, the spectral radius (maximum eigenvalue of $\boldsymbol{W}^{res}$), or the input scaling. Similarly to the leaking rate, these parameters are task dependent and need to be carefully set, in order to ensure reasonable performance. For a detailed guideline on how such hyper-parameters can be set the reader is referred to the review in [17]. Training a network to perform a certain task, comes then to training the output weights, $\boldsymbol{W}^{out}$ which is explained in Sect. 3 with respect to named entity recognition.

3 Using ESN for Contextual Word Representations

We are concerned with obtaining contextual-word representations using echo state networks; In particular, we are given pre-trained word embeddings of sentences. To that end, let us consider a sentence s consisting of L tokens which can be regarded as a sequence of its pre-trained token embeddings ie. $\boldsymbol{s} = (\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots \boldsymbol{x_L})$. Given this input sequence and an echo state network with randomly initialized input and reservoir weights, we pass the given input sentence one token at a time and collect its reservoir states at every time step as $\boldsymbol{c} = (\boldsymbol{r_1}, \boldsymbol{r_2}, \ldots \boldsymbol{r_L})$ by following Eq. (1).

By computing context in this manner, the representation of a token includes the previous context. Whereas for named entity recognition, it is observed in [5,32] that it is beneficial to have bi-directional context. To do this, we pass the sequence also in reverse order once and collect its corresponding reservoir states.

Therefore, contextual word-representation for each token is the concatenated reservoir states obtained from both directions ie. $\boldsymbol{c} = (\boldsymbol{r_1}, \boldsymbol{r_2}, \ldots \boldsymbol{r_L})$ where each $\boldsymbol{r_i}$ is $[\overrightarrow{\boldsymbol{r_i}}, \overleftarrow{\boldsymbol{r_i}}]$. Note, we use the same echo state network for both directions, unlike LSTM-based methods which use two separate hidden layers parsing in each direction. This is primarily because $\boldsymbol{W_{res}}$ is randomly initialized.

To solve a sequence labeling task such as NER, we pass this sequence $\boldsymbol{c}$ of contextual-word representations to the readout layer with weights $\boldsymbol{W_{out}}$ and its output can be collected in a sequence $\boldsymbol{y} = (\boldsymbol{y_1}, \boldsymbol{y_2}, \ldots \boldsymbol{y_L})$ following Eq. (2) with f_{out} as softmax activation function. Given a ground truth sequence $\boldsymbol{g} = (\boldsymbol{g_1}, \boldsymbol{g_2}, \ldots \boldsymbol{g_L})$ of NER tags where each $\boldsymbol{g_i}$ is a one-hot encoded vector of possible tags, then it is sufficient to optimize $\boldsymbol{W_{out}}$ by taking a gradient descent at the rate of η minimizing cross-entropy loss $\mathcal{L}$ between predicted sequence $\boldsymbol{y}$ and ground truth sequence $\boldsymbol{g}$.

$$\mathcal{L}(\boldsymbol{y}, \boldsymbol{g}) = -\sum_{i=1}^{L} \sum_{j=1}^{C} \boldsymbol{g}_i(j) \log(\boldsymbol{y}_i(j)) \tag{3}$$

$$W_{out} = W_{out} - \eta \nabla_{W_{out}} \mathcal{L}(\boldsymbol{y}, \boldsymbol{g}) \tag{4}$$

As input and reservoir weight matrices $\boldsymbol{W_{in}}$ and $\boldsymbol{W_{res}}$ are kept constant throughout this training process, we can break this whole procedure naturally into (1) generating contextual-word representations for all sentences (2) fitting a readout layer. Step (1) can be done just once for each echo state network setting and contextual-representations can be stored. In Step (2), any classifier of our choice (shallow or deep) can be trained offline with a batch gradient descent.

4 Experimental Results

In our experiments, we evaluate the contextual-word representations computed by echo state networks on the task of Named Entity Recognition. Specifically, we test our approach on GermEval Dataset from 2014 [2] in which the task is to identify named entities in sentences. In total, there are 12 classes comprising of 4 main classes: PERson, LOCation, ORGanisation, OTHer and 2 subclasses for each of them: -deriv and -part. The subclasses complicate the task by introducing nested named entities and derived entities. One example of a nested entity is “University of Bonn” which is an ORG but at the same time contains a location entity “Bonn” (LOC-part). Similarly, the second subclass includes word derivation, for example, “das Bonner Theater” (translated as “the theater of Bonn”) contains the word “Bonner” (LOC-deriv) derived from the entity “Bonn”. The dataset has a predefined splitting into a training, a development and a test set with 24 000, 2 200 and 5 100 sentences respectively.

4.1 Hyper-parameter Tuning

As echo state networks have several hyper-parameters which must be set beforehand, we first tune them by cross-validating on the development set. To that end,

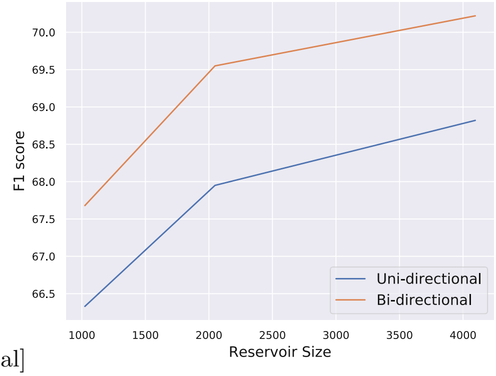
Table 1. Hyper-parameters tuning

Parameter	Value	Performance
Spectral radius	0.6	61.18
	0.7	**61.34**
	0.8	60.92
	0.9	60.60
	1.0	60.71
	1.1	60.50
	1.3	60.14
Leaking rate	0.3	60.92
	0.5	64.74
	0.6	65.11
	0.7	66.27
	0.75	**66.28**
	0.8	60.71
Input scaling	0.125	60.76
	0.25	**62.23**
	0.375	61.10
	0.5	60.92
	0.75	59.45

we use pre-trained word embeddings from FastText [13] as the inputs. For this purpose, we use the Flair implementation[1], which offers an easy to use framework for training and evaluating NER models along with access to different types of embeddings. To keep this tuning process tractable, as discussed before, we first generate contextual-word embeddings for different settings of ESN. Later, we fit a logistic regression model as our read out layer to train it to predict NER tags. For training this readout layer, we use Adadelta optimizer trained for 150 epochs with a learning rate of 0.1 and weight decay of 10^{-5}.

Spectral Radius: The spectral radius $\rho(\boldsymbol{W_{res}})$ which is the maximal absolute eigenvalue of the reservoir weight matrix. As $\boldsymbol{W_{res}}$ is initialized randomly, it is essential to set the spectral radius to a value less than 1.0 to satisfy the echo state property [12]. However, it was shown empirically that sometimes even higher values satisfy this condition and deliver better performance [17]. We varied the spectral radius between 0.6 and 1.3 and measured the resulting F1 score. Table 1 shows the F1 scores for different spectral radii; the performance peaks around 0.7 and then decreases continuously apart from a small local maximum around the unit spectral radius.

[1] https://github.com/zalandoresearch/flair.



[Uni vs Bi-directional]

Fig. 1. Influence of reservoir size on performance; the larger the reservoir, the better the performance. The fact that the bi-directional variant performs better than the uni-directional suggests that task depends on context from either sides

Leaking Rate: Next, we examine the leaking rate α which determines the speed of reservoir updates for its inputs. A high leaking rate indicates that a lot of old information vanishes through the "leak" in favor of new inputs. Table 1 shows that the best values are somewhere found between 0.7 and 0.75. This indicates that for this task, a higher leaking rate is preferred indicating that a faster update is necessary. Optimal settings of both leaking rate and spectral radius suggest that it is sufficient to have a shorter temporal context.

Input Scaling: Another key parameter of echo state networks is the scaling of input weight matrix $\boldsymbol{W_{in}}$. These weights can be controlled to influence the non-linearity of the reservoir responses. In our setting, we use tanh($\cdot$) activations; for linear tasks, it is thus beneficial to have small weights around 0.0, where the activation is almost linear. On the other side, for a more complex task, it might be better to choose a high scaling in order to make use of the non-linearity of the activation function. However, it was shown in previous research [17] that large weights can make the ESN unstable. Also, there exists a trade-off between non-linear mapping and memory capacity of echo state networks, it was suggested to use Extreme Learning Machines (ELMs) to tackle that subject [4]. In Table 1, the effect of the input scaling is presented. The highest f1-score corresponds to input scaling of 0.25.

Reservoir Size: Next, we investigate the influence of reservoir size for both uni- and bi-directional ESNs. Since a bi-directional ESN produces an embedding of twice the reservoir size, we instead look at the final embedding sizes to obtain a fair comparison. Figure 1 shows that the bi-directional embedding leads to a continuously better f1-score of around 1.5% indicating the task benefits from having bi-directional temporal context.

Table 2. NER on word embeddings

Model	Embedding dimension	Run time	Performance
Logistic regression (LR)	300	33 min	52.75 ± 0.19
Bi-ESN + LR	4096	37 min	69.04 ± 0.22
Bi-ESN + NN	4096	47 min	70.54 ± 0.10
Bi-LSTM	4096	10 h 12 min	75.45 ± 0.10
Bi-LSTM	256	3 h 3 min	76.87 ± 0.21

4.2 Evaluation on Different Embeddings

Next, we investigate the performance of contextual-word representations obtained from ESN on the test set by considering two types of word embeddings: FastText and Stacked embedding of FastText (dimension of 300) and Flair (dimension of 4096) [1]. We also consider two variants of ESN, first with a logistic regression (Bi-ESN + LR) read out layer and second with a neural network (Bi-ESN + NN) with a hidden layer consisting of 1000 neurons and 0.5 dropout. In either case, we fix our size of reservoir to have 2048 neurons amounting to 4096 dimensional embedding (bi-directional). The optimal setting of other hyper-parameters is obtained from the analysis presented in the previous section. We compare our approach to several baseline models: (i) logistic regression (LR) which does not use any context (ii) bi-directional LSTM (Bi-LSTM) which learns contextual word-representations in an end-to-end fashion for NER task with hidden size 256 as chosen in [1]. For fair comparison, we also train a variant with hidden size that matches the size of ESN reservoir.

On FastText Embeddings: Table 2 summarizes the performance of all methods trained with FastText embeddings as inputs. It is evident that the contextual-word representations generated by ESN lead to a strong improvement over a logistic regression method. Comparing the two ESN variants, it is noticed that a readout with neural networks has ≈1.5% improvement over logistic regression read out layer. The most important result is that the difference between the LSTM and the ESN models amounts to a value of 6% only. This suggests that LSTM variants do not improve much over ESN methods but only incur longer training times of ≈3 h. On the other hand, ESN methods can be quickly trained in less than an hour ≈0.5 h. These result suggests that random contextual-word representations obtained from ESN are already competitive and can be used as a baseline while bench-marking LSTM-based NER models.

On Word + Flair Embeddings: Table 3 presents the same set of experiments as before but with the combination of Flair and FastText token embeddings as inputs. In these experiments, we choose the size of the reservoir as 4396 resulting in contextual-word embedding size of 8792 due to bi-direction. One important observation is that all methods have a considerable improvement over the results presented in Table 2 which shows that Flair embeddings are more powerful in NER task. As Flair embeddings already encode contextual information

Table 3. NER on Flair + word embeddings

Model	Embedding dimension	Run time	Performance
Logistic regression	4396	2 h 10 min	73.78 ± 0.18
Bi-ESN + LR	8792	3 h 40 min	76.74 ± 0.03
Bi-ESN + NN	8792	4 h 40 min	78.17 ± 0.19
Bi-LSTM	8792	75 h	81.24 ± 0.13
Bi-LSTM	256	4 h 35 min	83.52 ± 0.21

as opposed to FastText, one might expect no further improvement by applying a further contextual encoding using ESN or LSTM. Nevertheless, both the ESN and the LSTM increase the performance noticeably by 3 to 5% and 8 to 10% respectively. Comparing the LSTM with the ESN models, we observe that the performance gap is just around 5%. These findings concur with our previous analysis that ESNs are capable of achieving competitive performance as LSTMs while requiring only short period of training time.

5 Conclusion

In this paper, we explored a random contextual-word encoder using echo state networks. Although their input and recurrent connections are not adapted to the task, they are still capable of providing context which can be leveraged for named entity recognition. Experiments suggest that they can be trained quickly and achieve competitive accuracy when compared to state-of-the-art methods. As the performance between trained encoders such as LSTMs and our ESN-based random context encoders is not much, our method can be used as a baseline for evaluating other contextual-word learning methods. There are several aspects in echo state networks which can be further explored; these include sparsity of connections, the structure of reservoir (random graphs, scale-free networks). Also, there has been some research into ESNs with multiple layers [18] that could enhance the performance. As future work, we intend to pursue these ideas which would bring the performance gap further down.

References

1. Akbik, A., Blythe, D., Vollgraf, R.: Contextual string embeddings for sequence labeling. In: Proceedings of International Conference on Computational Linguistics (2018)
2. Benikova, D., Biemann, C., Kisselew, M., Pado, S.: Germeval 2014 Named Entity Recognition Shared Task (2014)
3. Bowman, S.R., Angeli, G., Potts, C., Manning, C.D.: A large annotated corpus for learning natural language inference. arXiv preprint arXiv:1508.05326 (2015)

4. Butcher, J.B., Verstraeten, D., Schrauwen, B., Day, C., Haycock, P.: Reservoir computing and extreme learning machines for non-linear time-series data analysis. Neural Netw. **38**, 76–89 (2013)
5. Conneau, A., Kiela, D., Schwenk, H., Barrault, L., Bordes, A.: Supervised learning of universal sentence representations from natural language inference data. arXiv preprint arXiv:1705.02364 (2017)
6. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. J. Am. Soc. Inf. Sci. **41**(6), 391–407 (1990)
7. Doya, K.: Bifurcations in the learning of recurrent neural networks. In: Proceedings IEEE International Symposium on Circuits and Systems (1992)
8. Frank, S.L.: Learn more by training less: systematicity in sentence processing by recurrent networks. Connect. Sci. **18**(3), 287–302 (2006)
9. Gallicchio, C., Micheli, A., Pedrelli, L.: Deep echo state networks for diagnosis of Parkinson's disease. arXiv preprint arXiv:1802.06708 (2018)
10. Hinaut, X., Dominey, P.F.: On-line processing of grammatical structure using reservoir computing. In: Villa, A.E.P., Duch, W., Érdi, P., Masulli, F., Palm, G. (eds.) ICANN 2012. LNCS, vol. 7552, pp. 596–603. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33269-2_75
11. Jaeger, H., Haas, H.: Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science **304**(5667), 78–80 (2004)
12. Jäger, H.: The "Echo State" approach to analysing and training recurrent neural networks. Technical report 148, GMD (2001)
13. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759 (2016)
14. Kim, Y.: Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 (2014)
15. Kiros, R., et al.: Skip-thought vectors. In: Proceedings of Advances in Neural Information Processing Systems, pp. 3294–3302 (2015)
16. Lin, X., Yang, Z., Song, Y.: Short-term stock price prediction based on echo state networks. Expert Syst. Appl. **36**(3), 7313–7317 (2009)
17. Lukoševičius, M.: A practical guide to applying echo state networks. In: Montavon, G., Orr, G.B., Müller, K.-R. (eds.) Neural Networks: Tricks of the Trade. LNCS, vol. 7700, pp. 659–686. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35289-8_36
18. Ma, Q., Shen, L., Cottrell, G.W.: Deep-ESN: a multiple projection-encoding hierarchical reservoir computing framework. arXiv preprint arXiv:1711.05255 (2017)
19. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
20. Nadeau, D., Sekine, S.: A survey of named entity recognition and classification. Lingvisticae Investigationes **30**(1), 3–26 (2007)
21. Pascanu, R., Mikolov, T., Bengio, Y.: On the difficulty of training recurrent neural networks. In: Proceedings of the ICML (2013)
22. Pennington, J., Socher, R., Manning, C.: Glove: global vectors for word representation. In: Proceedings of Conference on Empirical Methods in Natural Language Processing, pp. 1532–1543 (2014)
23. Peters, M.E., et al.: Deep contextualized word representations. arXiv preprint arXiv:1802.05365 (2018)
24. Rajpurkar, P., Zhang, J., Lopyrev, K., Liang, P.: SQuAD: 100,000+ questions for machine comprehension of text. arXiv preprint arXiv:1606.05250 (2016)

25. Ramamurthy, R., Bauckhage, C., Buza, K., Wrobel, S.: Using echo state networks for cryptography. In: Lintas, A., Rovetta, S., Verschure, P.F.M.J., Villa, A.E.P. (eds.) ICANN 2017. LNCS, vol. 10614, pp. 663–671. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68612-7_75
26. Ramamurthy, R., Bauckhage, C., Sifa, R., Wrobel, S.: Policy learning using SPSA. In: Kůrková, V., Manolopoulos, Y., Hammer, B., Iliadis, L., Maglogiannis, I. (eds.) ICANN 2018. LNCS, vol. 11141, pp. 3–12. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01424-7_1
27. Socher, R., et al.: Recursive deep models for semantic compositionality over a sentiment treebank. In: Proceedings of Conference on Empirical Methods in Natural Language Processing (2013)
28. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Proceedings of Advances in Neural Information Processing Systems, pp. 3104–3112 (2014)
29. Sutton, C., McCallum, A., et al.: An introduction to conditional random fields. Found. Trends Mach. Learn. **4**(4), 267–373 (2012)
30. Tong, M.H., Bickett, A.D., Christiansen, E.M., Cottrell, G.W.: Learning grammatical structure with echo state networks. Neural Netw. **20**(3), 424–432 (2007)
31. Toutanova, K., Klein, D., Manning, C.D., Singer, Y.: Feature-rich part-of-speech tagging with a cyclic dependency network. In: Proceedings of Conference on HLT-NAACL (2003)
32. Wieting, J., Kiela, D.: No training required: exploring random encoders for sentence classification. arXiv preprint arXiv:1901.10444 (2019)
33. Yadav, V., Bethard, S.: A survey on recent advances in named entity recognition from deep learning models. In: Proceedings of International Conference on Computational Linguistics (2018)