Question Classification with Untrained Recurrent Embeddings

Daniele Di Sarli[0000−0003−4129−0131], Claudio Gallicchio[0000−0002−6692−2564], and Alessio Micheli[0000−0001−5764−5238]

Department of Computer Science
University of Pisa, Pisa, Italy
`daniele.disarli@gmail.com, {gallicch,micheli}@di.unipi.it`

Abstract. Recurrent Neural Networks (RNNs) are at the foundation of many state-of-the-art results in text classification. However, to be effective in practical applications, they often require the use of sophisticated architectures and training techniques, such as gating mechanisms and pre-training by autoencoders or language modeling, with typically high computational cost. In this work, we show that such techniques could actually be not always necessary. In fact, our experimental results on a Question Classification task indicate that using state-of-the-art Reservoir Computing approaches for RNN design, it is possible to achieve competitive or comparable accuracy with a considerable advantage in terms of required training times.

Keywords: Text classification · Recurrent Neural Networks · Echo State Networks.

1 Introduction

Recurrent Neural Networks (RNNs) have long been the de-facto standard neural architectures for many Natural Language Processing tasks [1,8,23,29], mainly because they allow to model the input and output text as a sequence of words or characters. Unfortunately, during training vanilla implementations of RNNs suffer from the well-known problems of gradient vanishing and gradient explosion, which make these networks difficult to train in the presence of long-term dependencies within the input [2].

Some approaches have gained popularity for their ability to avoid or alleviate the problems associated with the gradient propagation during training. For example, gated architectures like Long Short-Term Memory (LSTM) [15] and Gated Recurrent Unit (GRU) [8] are based on the idea of *gating mechanisms* that selectively remember and forget by regulating the flow of information through each time step, helping to alleviate the vanishing of the gradient. Recently, the development of the Transformer architecture [34] made it possible to more easily perform training by not using any recurrent network within the model and employing self-attention mechanisms instead. Increasingly often, transfer learning is used to train a task-independent language model on a large variety of text corpora (for example by employing an autoencoder or a classifier with a next-step prediction task) and then fine-tune it to the task at hand.

These techniques can lead to a significant increase of cost in terms of training time due to their considerable use of computational resources, with different kinds of repercussions such as economic availability, financial costs, and environmental impact. Currently, training a single Transformer model has been estimated to produce about 87 kg of CO_2 on commonly used hardware and cloud computing services, with a financial cost in US dollars between \$289 and \$981 [33]. Do these high cost techniques provide an equally significant improvement in predictive performance? In this paper, we try to address this question by proposing an approach based on RNNs from the class of Reservoir Computing (RC) [26,35], and comparing it with current state-of-the-art results in the literature. In particular we propose the use of Echo State Networks (ESNs) [16,17], a recurrent RC model, to produce by means of randomly initialized and untrained weights an embedding for the input text, which can then be used for classification tasks. While the network is largely untrained, we use advances in the architectural setup of ESNs and we explore the impact of an attention mechanism in this context. Unlike the commonly used approaches, thanks to the fact that the recurrent part of our model is completely untrained, we are able to achieve a strikingly fast training process. We then experimentally assess the feasibility and the performance of our approach with a focused analysis on a Question Classification task.

We briefly introduce the characteristics and advantages of the ESN model in Section 2, where we also address advances on recurrent connections shaping. In Section 3 we present the proposed models, which we then validate on a Question Classification dataset, described in Section 4. Our experiments and methodology are reported in Section 5, while a discussion of the results is presented in Section 6. Finally, in Section 7 we draw the conclusions of this study.

2 Echo State Networks

The framework of RNNs offers a useful and effective method for modeling sequences. In what follows, we use T to denote the length of a generic input sequence. Whenever a specific sequence i is considered, its length is denoted by T_i. Moreover, we use N_U, N_R and N_Y respectively to denote the size of the input layer, the number of hidden recurrent units (i.e. the size of the state embedding), and the number of output units in the RNN model. Given an input sequence composed of vectors $\boldsymbol{u}(1), \dots, \boldsymbol{u}(T) \in \mathbb{R}^{N_U}$, a generic RNN scans the input left-to-right and computes a sequence of states $\boldsymbol{x}(1), \dots, \boldsymbol{x}(T) \in \mathbb{R}^{N_R}$ having the same length T. From these states, an output (in the form of a sequence or of a single element) is then computed. RNNs are usually trained by gradient descent algorithms, which are subject to the problems associated with the gradient, as briefly discussed in Section 1, and can be costly to run.

On the other hand, radically different approaches like ESNs [16,17], from the RC paradigm, are based on the stable initialization of the recurrent dynamics so that the training of the parameters in the recurrent part of the network can be avoided altogether. The state of the network at each time step is computed by an untrained dynamical system with randomly initialized parameters called "reservoir", and the output is typically extracted from the state of the reservoir by means of simple linear regression techniques (even though more complex approaches can be used) [26]: ESNs are thus an efficient approach to modeling and training RNNs.

The state dynamics of an ESN at a particular time step t, in the case of leaky-integrator neurons [18] and hyperbolic tangent activation functions, are ruled by the following equation:

$$\boldsymbol{x}(t) = (1-a)\,\boldsymbol{x}(t-1) + a\tanh\Big(\boldsymbol{W}_{in}\boldsymbol{u}(t) + \hat{\boldsymbol{W}}\boldsymbol{x}(t-1)\Big), \tag{1}$$

where $\boldsymbol{x}(0) = \boldsymbol{0}$, $\boldsymbol{W}_{in} \in \mathbb{R}^{N_R \times N_U}$ is the input-to-reservoir weight matrix, and $\hat{\boldsymbol{W}} \in \mathbb{R}^{N_R \times N_R}$ is the recurrent reservoir-to-reservoir weight matrix. The scalar value $a \in \mathbb{R}$ is the *leaking rate*, under the constraint that $0 < a \leq 1$. For simplicity of notation, here and in the rest of this paper the bias term is omitted.

The key difference between an ESN and a vanilla RNN is in the fact that in the ESN the values in the weight matrices $\boldsymbol{W}_{in}$ and $\hat{\boldsymbol{W}}$ are not trained, instead they are initialized on the basis of stability constraints. These are given by the global asymptotic stability property known as the Echo State Property [26,16,36], and, under a practical perspective, they entail the control of algebraic properties of the recurrent weight matrix of the dynamical reservoir. Specifically, the weight values in $\hat{\boldsymbol{W}}$ are randomly initialized and then re-scaled to control the value of the spectral radius $\rho = \rho(\hat{\boldsymbol{W}})$ (i.e. its largest eigenvalue in absolute value). Similarly, the values in $\boldsymbol{W}_{in}$ are randomly chosen from a uniform distribution on $[-\omega, \omega]$, where $\omega \in \mathbb{R}^+$ acts as input scaling. The values of ρ and ω are hyperparameters that are chosen by model selection. Moreover, both $\boldsymbol{W}_{in}$ and $\hat{\boldsymbol{W}}$ in Equation 1 can be sparse matrices, since this causes a drop in the state transition computational cost, typically without any associated loss in terms of predictive performance [11].

After the input sequence has been fed in, the states produced by the ESN can be used to compute the output. Given the typical high dimensionality of the reservoir, it can be sufficient to use a simple linear layer ("readout") to perform the classification. In that case, the output $\boldsymbol{y}(t) \in \mathbb{R}^{N_Y}$ for a generic state $\boldsymbol{x}(t)$ is simply:

$$\boldsymbol{y}(t) = \boldsymbol{W}_{out}\boldsymbol{x}(t), \tag{2}$$

where $\boldsymbol{W}_{out} \in \mathbb{R}^{N_Y \times N_R}$ is the matrix containing the output weights, which are the only free parameters that are adjusted on a training set. Given the formulation in Equation 2, training reduces to solving the following least squares problem:

$$\min_{\boldsymbol{W}_{out}} \|\boldsymbol{W}_{out}\boldsymbol{X} - \boldsymbol{Y}_{tg}\|_2^2. \tag{3}$$

In Equation 3 we use $\boldsymbol{X} \in \mathbb{R}^{N_R \times N_{train}}$ to denote the state matrix, i.e. the column-wise concatenation of the N_{train} states produced by the ESN that need to be classified, and $\boldsymbol{Y}_{tg} \in \mathbb{R}^{N_Y \times N_{train}}$ to indicate the column-wise concatenation of

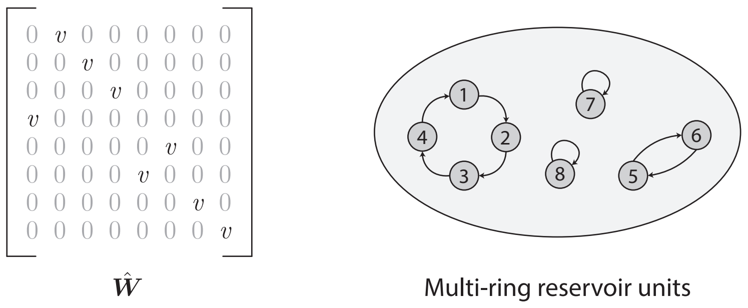


Fig. 1. On the left, an example of a recurrent matrix $\hat{\boldsymbol{W}}$ generated as per Equation 5. On the right, a representation of the corresponding multi-ring reservoir.

the target vectors. The parameters of the linear readout, i.e. the weight values in $\boldsymbol{W}_{out}$, can then be computed in closed-form by exploiting direct methods, such as ridge regression [26], as follows:

$$\boldsymbol{W}_{out} = \boldsymbol{Y}_{tg}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{X}^T + \lambda_r \boldsymbol{I})^{-1}, \tag{4}$$

where $\boldsymbol{I}$ is the identity matrix, and $\lambda_r \in \mathbb{R}^+$ is the regularization parameter.

2.1 Multi-ring Reservoir Topology

For the recurrent part, the networks that we are proposing adopt an ESN that follows the same dynamics as in Equation 1. The only difference is that the matrix $\hat{\boldsymbol{W}}$ is constructed in order to implement a constrained topology [32,6]. In particular, we take

$$\hat{\boldsymbol{W}} = v\boldsymbol{P}, \tag{5}$$

where $\boldsymbol{P} \in \{0,1\}^{N_R \times N_R}$ is a randomly generated permutation matrix and $v \in \mathbb{R}^+$ is a scalar that determines the spectral radius of $\hat{\boldsymbol{W}}$, i.e. $\rho(\hat{\boldsymbol{W}}) = v$. This follows from the fact that since $\boldsymbol{P}$ is a permutation matrix it is also orthogonal, i.e. for any vector $\boldsymbol{w}$:

$$\|v\boldsymbol{P}\boldsymbol{w}\| = v\,\|\boldsymbol{w}\|\,. \tag{6}$$

If $\boldsymbol{w}$ is an eigenvector of matrix $v\boldsymbol{P}$ with associated eigenvalue λ, i.e. $v\boldsymbol{P}\boldsymbol{w} = \lambda\boldsymbol{w}$, then it follows that

$$\|v\boldsymbol{P}\boldsymbol{w}\| = |\lambda|\,\|\boldsymbol{w}\|\,. \tag{7}$$

From (6) and (7) we conclude that $|\lambda| = v$ for all eigenvalues, hence $\rho(v\boldsymbol{P}) = v$. For this reason, in the following we will consider $\rho = v$, whose value is to be selected by hyperparameter search.

The "multi-ring" layout that emerges from this configuration (see Figure 1) has many advantages, the most important one being that it allows building large reservoirs with minimal state transition cost. In fact, the matrix-vector multiplication $\hat{\boldsymbol{W}}\boldsymbol{x}(t-1)$ in Equation 1 can be implemented in linear time in the case of a multi-ring reservoir. Moreover, the space requirements for matrix

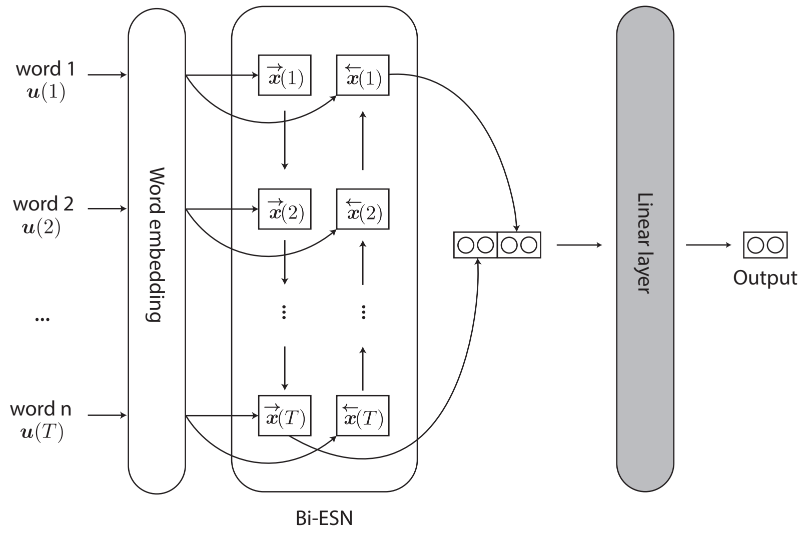

word 1
$\boldsymbol{u}(1)$
word 2
$\boldsymbol{u}(2)$
...
word n
$\boldsymbol{u}(T)$
Word embedding
$\overrightarrow{\boldsymbol{x}}(1)$
$\overleftarrow{\boldsymbol{x}}(1)$
$\overrightarrow{\boldsymbol{x}}(2)$
$\overleftarrow{\boldsymbol{x}}(2)$
$\overrightarrow{\boldsymbol{x}}(T)$
$\overleftarrow{\boldsymbol{x}}(T)$
Bi-ESN
Linear layer
Output

Fig. 2. Representation of the Bi-ESN model. Input words are transformed to vectors via pretrained word embeddings, then they are fed through a bidirectional leaky ESN. The final states are then concatenated for each direction, and fed to a linear classifier. The only parts of the model that undergo training are represented by a shaded background: in this case, only the final linear layer.

$\hat{\boldsymbol{W}}$ shrink from $O(N_R^2)$ to $O(N_R)$. Even further, the time required for initializing the network is reduced since it is not necessary to compute the spectral radius of $\hat{\boldsymbol{W}}$ to rescale it, but it is possible to cheaply initialize the matrix with the desired value of ρ.

3 Proposed models

Our proposed models implement a bidirectional recurrent architecture [5,30]: we use two separate networks to scan the input from left to right and from right to left. In the following sections we present the variants of the models that we designed, all of which adopt an ESN for the recurrent module but use different implementations for the readout. Specifically, the first model uses a simpler readout component and is described in Section 3.1. The second model, which includes a self-attention mechanism, is presented in Section 3.2.

3.1 Bi-ESN

With our simplest model, Bi-ESN, we introduce in the literature the use of a bidirectional orthogonal (multi-ring) architecture for the reservoir of a leaky ESN, in order to produce a fixed size untrained embedding of the input text as illustrated in Figure 2. The embedding is created by running the input through

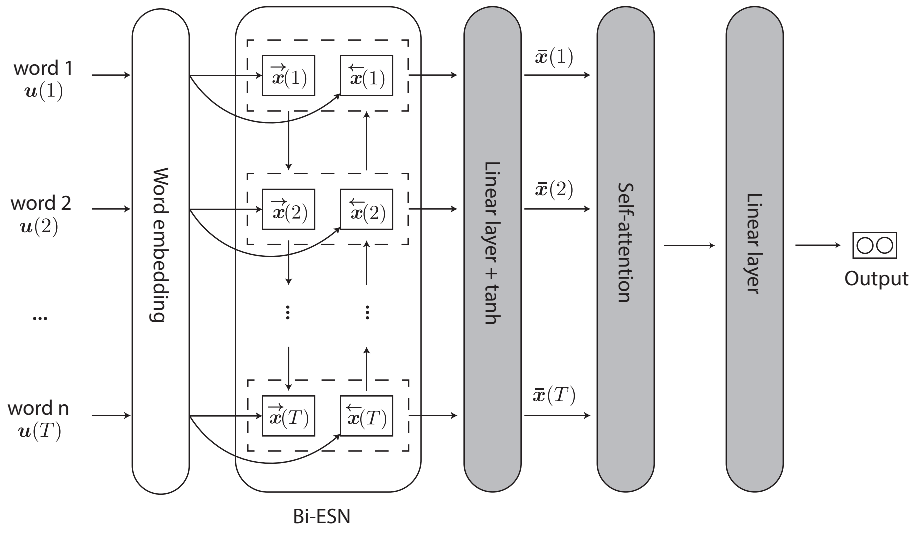


Fig. 3. Representation of the Bi-ESN-Att model. Input words are transformed to vectors via pretrained word embeddings, then they are fed through a bidirectional leaky ESN. All states from each direction are then concatenated (dashed rectangles) and fed one by one to a linear layer that performs dimensionality reduction. After that, the self-attention mechanism selects the most important states, which are summed together and fed to a linear classifier. The only parts of the model that undergo training are represented by a shaded background.

the bidirectional ESN and then taking the concatenation of the last forward and backward states, resulting in a single vector of size $2N_R$. This vector is then processed by a simple linear layer.

Let us denote with $\overrightarrow{\boldsymbol{x}}(t), \overleftarrow{\boldsymbol{x}}(t) \in \mathbb{R}^{N_R}$ respectively the forward and backward state associated to $\boldsymbol{u}(t)$. If $\overrightarrow{\boldsymbol{x}}_n(t)$ is a forward state for the n-th training example (and similarly for $\overleftarrow{\boldsymbol{x}}_n(t)$), then in order to train the Bi-ESN with ridge regression we apply the same formulation as in Equation 4, where in this case the state matrix contains the concatenation of forward and backward states, i.e. $\boldsymbol{X} \in \mathbb{R}^{2N_R \times N_{train}}$, given by:

$$\boldsymbol{X} = \begin{bmatrix} \overrightarrow{\boldsymbol{x}}_1(T_1) & \overrightarrow{\boldsymbol{x}}_2(T_2) & \dots & \overrightarrow{\boldsymbol{x}}_{N_{train}}(T_{N_{train}}) \\ \overleftarrow{\boldsymbol{x}}_1(1) & \overleftarrow{\boldsymbol{x}}_2(1) & \dots & \overleftarrow{\boldsymbol{x}}_{N_{train}}(1) \end{bmatrix} \tag{8}$$

with $T_1, T_2, \dots, T_{N_{train}}$ representing the lengths of the training input sequences.

3.2 Bi-ESN-Att

We compare Bi-ESN with a more advanced model that is still based on a multi-ring leaky Bi-ESN, but uses a more sophisticated readout implementation. The model that we are proposing is a novel application of a self-attention mechanism to a bidirectional multi-ring ESN. As shown in Figure 3, unlike Bi-ESN this model makes use of all the states produced by the ESN, both in the forward and backward direction. In fact, the forward and backward sequences of states are concatenated to produce a single sequence of vectors of size $2N_R$, each of which goes through the same linear layer with the purpose of reducing the vector dimensionality to N_D. If $\overrightarrow{\boldsymbol{x}}(t), \overleftarrow{\boldsymbol{x}}(t) \in \mathbb{R}^{N_R}$ are respectively the forward and backward states associated to $\boldsymbol{u}(t)$, and $\boldsymbol{W}_{dr} \in \mathbb{R}^{N_D \times 2N_R}$ is a weight matrix, then the state vector after dimensionality reduction, $\bar{\boldsymbol{x}}(t) \in \mathbb{R}^{N_D}$, is computed as:

$$\bar{\boldsymbol{x}}(t) = \tanh\left(\boldsymbol{W}_{dr}\left[\overrightarrow{\boldsymbol{x}}(t), \overleftarrow{\boldsymbol{x}}(t)\right]\right). \tag{9}$$

After that, an attention mechanism selects the most important states from the whole sequence. The particular kind of attention that we use is the "self-attention" [25], which unlike other techniques (see for instance [1]) does not require any additional information other than the sequence itself. Intuitively, in its simplest form the attention mechanism works by assigning a score to each of the states produced by the ESN, based on the relevance that they have in relation to the task. These scores are then used to compute a weighted sum of the state vectors, which leads to a fixed size representation for the whole sentence focused on the most important features. Let T be the length of the input sequence, let $r \in \mathbb{R}$ be the number of parts in the sentence on which the attention mechanism is allowed to focus, and let $d_a \in \mathbb{R}$ represent the number of hidden units for computing the scores. Then, if $\boldsymbol{W}_{s1} \in \mathbb{R}^{d_a \times N_D}$ and $\boldsymbol{W}_{s2} \in \mathbb{R}^{r \times d_a}$ are weight matrices, the self-attention scores $\boldsymbol{A} \in \mathbb{R}^{r \times T}$ are computed as:

$$\begin{aligned}
\bar{\boldsymbol{X}} &= \begin{bmatrix} \bar{\boldsymbol{x}}(1)^T \\ \bar{\boldsymbol{x}}(2)^T \\ \vdots \\ \bar{\boldsymbol{x}}(T)^T \end{bmatrix} \in \mathbb{R}^{T \times N_D} \\
\boldsymbol{A} &= \operatorname{softmax}\left(\boldsymbol{W}_{\mathrm{s2}} \tanh\left(\boldsymbol{W}_{\mathrm{s1}} \bar{\boldsymbol{X}}^{\mathrm{T}}\right)\right).
\end{aligned} \tag{10}$$

As can be noticed from Equation 10, none of the weight matrices depend on the length of the sequence. The attention scores are then used to extract a fixed-size weighted sum of the most important states into a matrix $\boldsymbol{M} \in \mathbb{R}^{r \times N_D}$:

$$\boldsymbol{M} = \boldsymbol{A}\bar{\boldsymbol{X}}. \tag{11}$$

As for hyperparameters r and d_a, we simply take $r = 1$ and $d_a = N_D$. In this case, $\boldsymbol{M}$ reduces to a vector of size N_D that we then classify using a linear layer.

Note that all free parameters of the model can be trained end-to-end by gradient descent. Since unlike what happens in standard RNNs here the gradient only flows through a short path, we do not incur in the issue of gradient vanishing.

4 TREC Dataset

The TREC dataset for Question Classification[1] [24] is a commonly used benchmark for Natural Language Processing which deals with the classification of a number of sentences, written in English, into one of 6 classes about their topic (i.e. whether they ask about a person, a location, a number, a human being, a description or an entity).

The dataset has been split in three folds: training, validation and test. The test fold is directly provided by the authors of the dataset [24] and is composed of 500 labeled questions. We divided the training data, composed of 5452 labeled questions, by the commonly used "80/20 rule", where 80% of the instances (chosen at random) are used for training and the other 20% for validation. This yields a training set of 4362 questions and a validation set of 1090 questions, with similar class distributions between the two sets (we did not perform an explicit stratification).

The questions are tokenized and each word is then represented by a pretrained FastText embedding vector for the English language, with 300 dimensions [14]. In case of words without a corresponding embedding, a random vector of the same shape is used. This vector is different for each missing word. While the NLP community is pushing towards context-sensitive word embeddings, in the current setting we chose FastText for its relative efficiency.

5 Experiments

We performed all our experiments[2] on a single NVIDIA Tesla V100 with 16 GB of memory, and we developed our models by using the PyTorch framework [27] which provides automatic differentiation. In addition to the Bi-ESN and Bi-ESN-Att that we have described in Section 3, we also implemented a standard bidirectional GRU (Bi-GRU) that we use for comparison purposes on the analysis of accuracy and efficiency.

After hyperparameter tuning on the validation set, our models have been retrained on the whole training and validation data to get a final estimate of the performance. In addition, all measurements of the test performance have been performed by repeating the process 10 times, with different random initializations each time, and averaging the results.

The simple linear readout allowed us to train Bi-ESN by ridge regression, while all other models were trained by mini-batched gradient descent using the Adam algorithm [21] and cross entropy as loss function. This led to a very short training time for Bi-ESN, which allowed us to cheaply compute also an ensemble out of the predictions of 10 identical networks with different random initializations (we simply average the output scores and then take as final prediction the class corresponding to the highest averaged score). As before, also for the ensemble we

[1] http://cogcomp.org/Data/QA/QC/

[2] Source code for reproducing the experiments is available at https://github.com/danieleds/qc_with_untrained_recurrent_embeddings.

repeat the training process 10 times in order to compute a mean accuracy and standard deviation, for a total of $10 \times 10 = 100$ repetitions.

For model selection of Bi-ESN and Bi-ESN-Att, we chose the number of recurrent units N_R within $[500, 10000]$. The values for ω and ρ have been selected in $[e^{-7}, e^{4}]$, while the connectivity ratio of the input-to-reservoir matrix and leaking rate have been chosen in $(0, 1)$. The ESN hyperparameters have been chosen separately for the forward and backward direction. For Bi-ESN-Att, the number of units N_D has been selected in $\{128, 256, 512\}$. Regarding the optimization algorithm, we chose a learning rate in $[e^{-9}, e^{-3}]$ and an early stopping strategy with a maximum of 500 epochs, while for regularization we used dropout and a weight decay strength in $[e^{-9}, 1]$. In the case of Bi-ESN, which is instead trained by ridge regression, we simply choose the regularization parameter λ_r within $[10^{-6}, 10^{6}]$. For searching within the hyperparameter space we used a combination of random search [4], simulated annealing [22] and tree-structured Parzen estimator [3]: at each iteration, we randomly choose one of these three algorithms to select the next point in the hyperparameter space.

6 Results

The results of our experiments are reported in Table 1. For comparison, we also report the performance achieved by state-of-the-art models in the literature.

The first important observation that can be drawn from Table 1 is that all our proposed models which are based on an ESN, and that are thus exploiting a completely untrained recurrent dynamics, are able to compete against a fully trained Bi-GRU. In the case of the ensemble model, the accuracy is even matched. The remarkable fact is that this comes with an extremely lower training cost for the Bi-ESN which has turned out to be at least 70 times more efficient than Bi-GRU. In fact even the ensemble model, which requires the training of 10 differently initialized classifiers, is still highly competitive against the Bi-GRU in terms of training time (and could trivially be further improved by applying parallelization between the different instances).

Adding an attention mechanism on top of the ESN as we did with Bi-ESN-Att led to a gain in predictive performance with respect to Bi-ESN, but this gain was rather limited. This may be due to the relative simplicity of the TREC dataset, which exhibits short sentences with a relatively simple structure. In fact, many sentences start with “Who is”, “How many”, “Where is”, “When did”, and so on. The bidirectional architecture (and in particular the backward direction), then, seems sufficient to capture these important features in the data, as illustrated in Figure 4. Still, Bi-ESN-Att is more efficient than a simple Bi-GRU, requiring just one-seventh of the time to get trained.

Regarding the reported literature results it is worth noticing that, regarding the SVM [31] and KDA [9], the authors do not specify how model selection was performed, so it is difficult to provide a uniform comparison of the generalization capability of these models when compared to our own. Also, among the different results shown for different configurations of the approaches, we have reported

Table 1. Results on the TREC dataset. Asterisks indicate those models for which the methodology for model selection was not specified (see the text for details).

Our implementations

Model	**Accuracy**	**Training time**
Bi-GRU	93.8 ± 0.4	450s ± 40
Bi-ESN	93.3 ± 0.6	6s ± 1
Bi-ESN, ensemble	93.8 ± 0.2	62s ± 17
Bi-ESN-Att	93.5 ± 0.9	65s ± 8

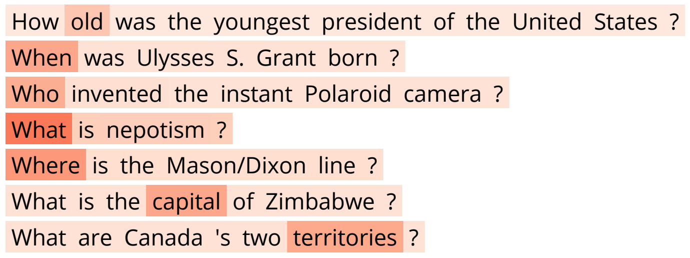
Previous literature

Model	**Accuracy**
SVM [31]	95.0 *
Paragraph Vector [37]	91.8
Ada-CNN [37]	92.4
CNN-non-static [20]	93.6
CNN-multichannel [20]	92.2
DCNN [19]	93.0
KDA [9]	94.3 *
LSTM [38]	93.2
Bi-LSTM [38]	93.0
C-LSTM [38]	94.6
U_T [7]	93.2
CNN$_{rnd}$ [7]	97.9
U_T+CNN$_{w2v}$ [7]	98.7

the best on the test set as highlighted by the authors. Moreover, the SVM uses as features 60 highly engineered hand-coded rules, which could directly harm generalization when applied to other datasets. The CNN$_{rnd}$ from [7] should have an architecture identical to the one previously introduced in [20], but the authors do not provide an explanation for the extremely high increase in accuracy with respect to the original paper. Finally, models U_T and U_T+CNN$_{w2v}$ make use of sentence embedding transfer learning, with weights trained on unrelated tasks on large text corpora, while we only make use of the examples within the TREC dataset and limit our use of transfer learning just to pre-trained word embeddings.

In light of the above considerations we can see how, with no more than 65 seconds of training time, our proposed models are able to approach or match the predictive performance of many of the models in the literature, with a few above-mentioned exceptions which could be attributed to a different model selection strategy or to the heavy use of transfer learning. A notable observation is how our Bi-ESN, with only 6 seconds of training time, is able to match (and slightly surpass) a fully trained Bi-LSTM, which is an architectural superset of our Bi-GRU for which we can thus estimate a supposedly similar (or worse) training time of around 7.5 minutes.

We were not able to reach the high accuracy of U_T+CNN$_{w2v}$ [7], however we want to highlight the fact that U_T and U_T+CNN$_{w2v}$ have more than 200M parameters. In comparison, our largest model (Bi-ESN-Att) has just 1.6M trainable



How old was the youngest president of the United States ?
When was Ulysses S. Grant born ?
Who invented the instant Polaroid camera ?
What is nepotism ?
Where is the Mason/Dixon line ?
What is the capital of Zimbabwe ?
What are Canada 's two territories ?

Fig. 4. Visualization of the intensity of the attention scores assigned by Bi-ESN-Att to some of the sentences in the dataset. As you can see, it is common for the network to focus mainly on the first word of the sentence since it carries the most important information for the task. This specific region of focus is implicitly provided by any bidirectional architecture without the need of self-attention.

parameters and, despite that, all our proposed models are able to compete with U_T, which uses the encoder of a transformer and is pre-trained with data from Wikipedia, web news, web question-answer pages and other sources.

7 Conclusion

Sophisticated architectures requiring high amounts of computational resources are not uncommon in the field of Natural Language Processing. While definitely effective and justified on most complex tasks, they can be overkill in other situations. In order to investigate how a highly efficient model can compete in these situations, for the first time in the literature we have proposed the use of a bidirectional multi-ring ESN, possibly associated to a self-attention mechanism.

To determine the efficacy of the approach, we have selected a Question Classification task which allowed us to compare our method and architecture with those of different kinds of works in the literature, showing how our own is comparable with the state-of-the-art performance of many of the alternatives. In the cases where a direct comparison has been possible, this showed the extreme efficiency of the proposed models.

In particular, we have demonstrated how a Bi-ESN shows basically the same accuracy of another recurrent model, Bi-GRU, while however presenting notable computational advantages, namely 1) not requiring any gating mechanism, and 2) keeping the input and recurrent weights untrained. In other words, the largest percentage of computational time used for training a GRU is actually unnecessary and detrimental. This can only get worse with other gated models, like LSTMs, for which to the same state size corresponds a higher number of parameters that need to be trained.

In addition, we showed how our Bi-ESN model is still able to compete against the more advanced attention mechanism of Bi-ESN-Att, which we showed to determine an improvement in accuracy that however, at least on this dataset, is quite limited. Still, the use of Bi-ESN-Att can be of interest even on this kind of dataset when looking for a more interpretable (and very efficient) model.

While within the limits of an analysis which has been focused on a Question Classification task, our results show the potential of Reservoir Computing methods and of their possible evolution. This potential is especially tangible with respect to the extreme efficiency of these methods, which is increasingly important in Natural Language Processing contexts that are often characterized by considerable amounts of data.

As future works, we plan to extend our analysis to more complex tasks in which an attention mechanism can have a higher impact. Moreover, we would like to assess the role of multiple recurrent layers as in DeepESN [12,13], which could provide richer information at different time scales, and of kernels [9], which could help extract more interesting features from the data. Finally, given the recently proven effectiveness of large language models for transfer learning [28,10], it would be interesting to explore how Reservoir Computing approaches can reduce the huge amount of time required to train these models, both in the case of training the language model itself, and in the case of training the task-dependent network.

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: 3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings (2015), http://arxiv.org/abs/1409.0473
2. Bengio, Y., Simard, P.Y., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE Trans. Neural Networks **5**(2), 157–166 (1994)
3. Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain. pp. 2546–2554 (2011)
4. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of Machine Learning Research **13**, 281–305 (2012)
5. Bianchi, F.M., Scardapane, S., Løkse, S., Jenssen, R.: Bidirectional deep-readout echo state networks. In: 26th European Symposium on Artificial Neural Networks, ESANN 2018 (2018)
6. Boedecker, J., Obst, O., Mayer, N.M., Asada, M.: Studies on reservoir initialization and dynamics shaping in echo state networks. In: Proc. of the 17th European Symposium on Artificial Neural Networks (ESANN). pp. 227–232. d-side publi. (2009)
7. Cer, D., Yang, Y., Kong, S., Hua, N., Limtiaco, N., John, R.S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Strope, B., Kurzweil, R.: Universal sentence encoder for english. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018: System Demonstrations. pp. 169–174. Association for Computational Linguistics (2018)
8. Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: 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. ACL (2014)

9. Croce, D., Filice, S., Basili, R.: On the impact of linguistic information in kernel-based deep architectures. In: AI*IA 2017 Advances in Artificial Intelligence - XVIth International Conference of the Italian Association for Artificial Intelligence, Proceedings. Lecture Notes in Computer Science, vol. 10640, pp. 359–371. Springer (2017)
10. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. CoRR **abs/1810.04805** (2018), http://arxiv.org/abs/1810.04805
11. Gallicchio, C., Micheli, A.: Architectural and markovian factors of echo state networks. Neural Networks **24**(5), 440–456 (2011)
12. Gallicchio, C., Micheli, A.: Deep reservoir computing: A critical analysis. In: 24th European Symposium on Artificial Neural Networks, ESANN 2016 (2016)
13. Gallicchio, C., Micheli, A., Pedrelli, L.: Deep reservoir computing: A critical experimental analysis. Neurocomputing **268**, 87–99 (2017)
14. Grave, E., Bojanowski, P., Gupta, P., Joulin, A., Mikolov, T.: Learning word vectors for 157 languages. In: Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018) (2018)
15. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997)
16. Jaeger, H.: The "echo state" approach to analysing and training recurrent neural networks – with an erratum note'. Bonn, Germany: German National Research Center for Information Technology GMD Technical Report (2001)
17. Jaeger, H., Haas, H.: Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. Science **304**(5667), 78–80 (2004)
18. Jaeger, H., Lukosevicius, M., Popovici, D., Siewert, U.: Optimization and applications of echo state networks with leaky-integrator neurons. Neural Networks **20**(3), 335–352 (2007). https://doi.org/10.1016/j.neunet.2007.04.016
19. Kalchbrenner, N., Grefenstette, E., Blunsom, P.: A convolutional neural network for modelling sentences. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, Volume 1: Long Papers. pp. 655–665. The Association for Computer Linguistics (2014)
20. Kim, Y.: Convolutional neural networks for sentence classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014. pp. 1746–1751. ACL (2014)
21. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: 3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings (2015)
22. Kirkpatrick, S., Jr., D.G., Vecchi, M.P.: Optimization by simmulated annealing. Science **220**(4598), 671–680 (1983)
23. Lei, Z., Yang, Y., Yang, M., Liu, Y.: A multi-sentiment-resource enhanced attention network for sentiment classification. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Volume 2: Short Papers. pp. 758–763. Association for Computational Linguistics (2018)
24. Li, X., Roth, D.: Learning question classifiers. In: 19th International Conference on Computational Linguistics, COLING 2002 (2002)
25. Lin, Z., Feng, M., dos Santos, C.N., Yu, M., Xiang, B., Zhou, B., Bengio, Y.: A structured self-attentive sentence embedding. In: 5th International Conference on Learning Representations, ICLR 2017, Conference Track Proceedings (2017)

26. Lukosevicius, M., Jaeger, H.: Reservoir computing approaches to recurrent neural network training. Computer Science Review **3**(3), 127–149 (2009)
27. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in PyTorch (2017), https://openreview.net/forum?id=BJJsrmfCZ
28. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners (2019), https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf
29. Sachan, D.S., Zaheer, M., Salakhutdinov, R.: Revisiting LSTM networks for semi-supervised text classification via mixed objective function. In: AAAI 2019 (2019)
30. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE Trans. Signal Processing **45**(11), 2673–2681 (1997)
31. da Silva, J.P.C.G., Coheur, L., Mendes, A.C., Wichert, A.: From symbolic to sub-symbolic information in question classification. Artif. Intell. Rev. **35**(2), 137–154 (2011)
32. Strauss, T., Wustlich, W., Labahn, R.: Design strategies for weight matrices of echo state networks. Neural computation **24**(12), 3246–3276 (2012)
33. Strubell, E., Ganesh, A., McCallum, A.: Energy and policy considerations for deep learning in NLP. In: ACL (1). pp. 3645–3650. Association for Computational Linguistics (2019)
34. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017. pp. 6000–6010 (2017)
35. Verstraeten, D., Schrauwen, B., D'Haene, M., Stroobandt, D.: An experimental unification of reservoir computing methods. Neural Networks **20**(3), 391–403 (2007)
36. Yildiz, I.B., Jaeger, H., Kiebel, S.J.: Re-visiting the echo state property. Neural networks **35**, 1–9 (2012)
37. Zhao, H., Lu, Z., Poupart, P.: Self-adaptive hierarchical sentence model. In: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015. pp. 4069–4076. AAAI Press (2015)
38. Zhou, C., Sun, C., Liu, Z., Lau, F.C.M.: A C-LSTM neural network for text classification. CoRR **abs/1511.08630** (2015), http://arxiv.org/abs/1511.08630