Echo State vs. LSTM Networks for Word Sense Disambiguation

Alexander Popov, Petia Koprinkova-Hristova(✉), Kiril Simov, and Petya Osenova

IICT, Bulgarian Academy of Sciences, Sofia, Bulgaria
{alex.popov,kivs,petya}@bultreebank.org, pkoprinkova@bas.bg

Abstract. Inspired by bidirectional long short-term memory (LSTM) recurrent neural network (RNN) architectures, commonly applied in natural language processing (NLP) tasks, we have investigated an alternative bidirectional RNN structure consisting of two Echo state networks (ESN). Like the widely applied BiLSTM architectures, the BiESN structure accumulates information from both the left and right contexts of target word, thus accounting for all available information within the text. The main advantages of BiESN over BiLSTM are the smaller number of trainable parameters and a simpler training algorithm. The two modelling approaches have been compared on the word sense disambiguation task (WSD) in NLP. The accuracy of several BiESN architectures is compared with that of similar BiLSTM models trained and evaluated on the same data sets.

Keywords: Recurrent neural networks · Echo state network · Long short-term memory · Natural language processing · Word sense disambiguation

1 Introduction

An important recent development in natural language processing (NLP) has been the adoption of recurrent neural networks (RNNs) as a viable tool for language modeling. Word sense disambiguation (WSD) is an NLP task aimed at assigning proper categories of meaning to words that are ambiguous (i.e. they can have several related or unrelated meanings depending on the context). For instance, the word "speaker" can refer to a person who speaks ("The next speaker will talk about new scientific achievements.") or a device reproducing speech ("The speaker sound was of very poor quality."). In order to do WSD we need to account for all words (the context) not only before but also after the target word ("speaker" in our example). In some cases the task might require examining even the context preceding a sentence boundary, and on rare occasions it might even depend on looking forward into and beyond the current sentence. Thus the ability of RNNs to keep a memory trace of past "events" at theoretically

I. V. Tetko et al. (Eds.): ICANN 2019, LNCS 11731, pp. 94–109, 2019.
https://doi.org/10.1007/978-3-030-30493-5_10

arbitrary distances from the present step is an obvious advantage over algorithms that collect information from a fixed window around the target word.

For a long time RNNs were considered difficult to train, as their memory capabilities are often thwarted in practice by the *exploding/vanishing gradients problem*. While the exploding gradients can be capped, a more elaborate solution was needed to combat the vanishing part. As an attempt to solve these problems *long short-term memory* cells [12] were designed to selectively forget information about old states and pay attention to new inputs (a good introduction to LSTMs can be found in [8]; another similar and newer development are *gated recurrent units* [2]). A further enhancement to such an architecture is making it *bidirectional* [9], i.e. the input sequence is fed into two recurrent context layers – one running from the beginning to the end of the sequence and the other one running in reverse. Bidirectional LSTMs (BiLSTMs) have been successfully applied to a number of sequence-to-sequence tasks in NLP [13,31–33].

Although LSTM architectures cope well with the *exploding/vanishing gradients problem*, their training via the gradient descent algorithm is a computationally demanding task, especially in the case of very deep networks. Aimed at the development of fast trainable RNNs, an alternative approach was proposed in 2002 independently by [18] under the name *liquid state machines* (LSM) and by [14] under the name *echo state networks* (ESN). Nowadays these are collectively referred to as *reservoir computing* (RC) [17] approaches. The idea consists of generating a random and sparsely connected recurrent reservoir of neurons whose mutual connection weights are not subject to training and a linear readout layer that can be tuned in one shot (presenting each training sample only once and solving the least squares problem). Since their emergence, RC approaches have been widely used for the modeling of a variety of dynamical systems [1]. Gallicchio et al. [7] compared deep ESN architectures with popular deep gated RNNs (like LSTM and GRU architectures) for time series prediction and demonstrated that in most benchmark problems deep ESNs outperform the fully trained RNNs.

Application of ESNs in NLP have started to appear only recently, so there are only a few works in this area. The possibility of language modelling via ESN was investigated in [5,11]. In [29,30], ESNs are applied for semantic role labeling in a multimodal robotic architecture. Other NLP applications are in the area of speech processing—[26,27], and in language modeling—[28]. To the best of our knowledge, there are yet no examples of ESNs being used for WSD. Our preliminary attempt to solve the WSD task using a single ESN reservoir [16] has shown that although the training and testing errors on predicted sense embedding vectors are quite small, the vector representations of the possible senses per word are very close to one another in the embedding space, resulting in low accuracy scores.

That is why, in an attempt to increase WSD accuracy, in present work we have adopted a bidirectional reservoir approach proposed initially in [6,24]. Similar to BiLSTMs, bidirectional reservoir structures (BiESN) were applied for feature extraction from time series – from forward and backward contexts (left and right ESN reservoir state vectors); the features are further used for classification purposes. In our model the readout is a linear combination of the current input and both reservoir states, and the training of the output connections is done via the *recursive least squares* (RLS) algorithm. Here we also compare BiESN models with similar BiLSTM models [22] evaluated on the same task of WSD.

The structure of the paper is as follows: the next section introduces ESN and BiESN architectures and compares them with similar LSTM models; Sect. 3 briefly describes the available research on BiLSTMs for WSD; the results obtained on WSD with BiESNs are presented next and compared with the accuracy of BiLSTM architectures; a section dedicated to conclusions and future work closes the paper.

2 Echo State Networks vs LSTM Architectures

2.1 ESN Basics

The structure of an ESN is shown on Fig. 1. It incorporates a dynamic reservoir of neurons with a sigmoid activation function (usually the hyperbolic tangent) and randomly generated recurrent connections W^{res}. The reservoir state $R(k)$ depends both on its previous state $R(k-1)$ and on the current input $in(k)$—Eq. 1. The network output is generated as a linear combination of the reservoir and input states—Eq. 2:

$$R(k) = (1-a)R(k-1) + a\tanh(W^{in}in(k) + W^{res}R(k-1)) \tag{1}$$

$$out(k) = W^{out}[in(k), R(k)] \tag{2}$$

Here W^{in} and W^{res} are $n_{in} \times n_R$ and $n_R \times n_R$ matrices that are randomly generated and are not trainable; n_{out}, n_{in} and n_R are the sizes of the corresponding vectors out, in and R; the parameter a, called leaking rate, influences the reservoir short-term memory and in many applications is omitted, i.e. a is set to 1; W^{out} is a $n_{out} \times (n_{in} + n_R)$ trainable matrix.

According to recipes in [14], the reservoir connection matrix W^{res} should be generated so as to guarantee the "echo state property" of the ESN, i.e. the changes in the input vector must be reflected like an "echo" in the output vector (meaning that the response effect should vanish gradually over time). This is achieved by proper normalization of the matrix W^{res} so that its spectral radius becomes smaller than 1 (although for some applications it has been shown that a spectral radius above 1 could also work). The output weights can be tuned by the least squares (LS) approach (or ridge regression) in one shot after the single presentation of all input/output training samples, or iteratively using its recursive version (RLS) – presenting training input/output samples one by one [14].

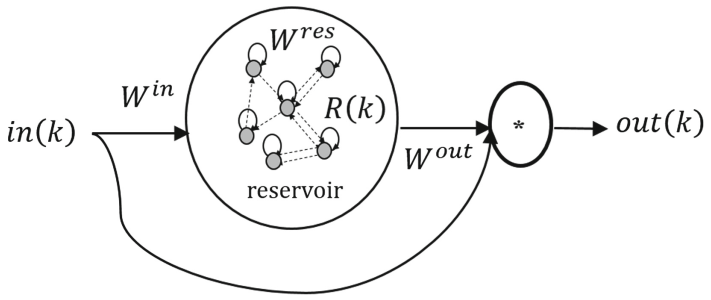

Fig. 1. Echo state network structure.

2.2 Echo State vs. Long Short-Term Memory Networks

A similar recurrent structure widely used in the area of NLP is the long short-term memory (LSTM) network [10], shown in Fig. 2. The LSTM cell model is described by the following set of equations:

$$m(k) = f(k) \circ m(k-1) + i(k) \circ \tanh(W_m^{in} in(k) + W_m^{res} mo(k-1) + b_m) \quad (3)$$

$$mo(k) = o(k) \circ \sigma(m(k)) \quad (4)$$

$$out(k) = W_m^{out} mo(k) \quad (5)$$

Here $\circ$ denotes Hadamard product; $m(k)$ is an internal state (memory) vector, $mo(k)$ is a memory units output vector and $out(k)$ is readout from the structure (following the analogy with ESNs) at the current time instant k; $f(k)$, $i(k)$ and $o(k)$ denote the states of the forget, input and output gates respectively. The dynamics of the gates is governed by the following recurrent equations:

$$i(k) = \sigma(W_i^{in} in(k) + W_i^{res} mo(k-1) + b_i) \quad (6)$$

$$f(k) = \sigma(W_f^{in} in(k) + W_f^{res} mo(k-1) + b_f) \quad (7)$$

$$o(k) = \sigma(W_o^{in} in(k) + W_o^{res} mo(k-1) + b_o) \quad (8)$$

Here σ denotes the sigmoid function. In order to maintain the analogy with ESNs, the recurrent connection weights are denoted by the superscript res, while the input connection weights – by the superscript in. The dimension of the weight matrices depends on the number of memory units n_m and the input vector size n_{in} as follows: W_*^{in}, where $*$ is for m, i, f or o respectively, are matrices of size $n_{in} \times n_m$; W_*^{res} are matrices of size $n_m \times n_m$; b_* are vectors of size n_m.

Looking at Figs. 1 and 2 and the corresponding Eqs. 3–8 and 1–2, the similarity is obvious. Both accumulate temporal information – in the reservoir (Eq. 1) or in the memory (Eq. 3) of the recurrent structures. The only difference is that ESNs have no bias term (although it could be added) and that its parameters analogous to the input, forget and output gates (according to Eq. 1 they are a, $(1-a)$ and 1, respectively) are constant, while the LSTM gates are dynamically changing (Eqs. 6–8, respectively). In contrast to ESNs, whose trainable

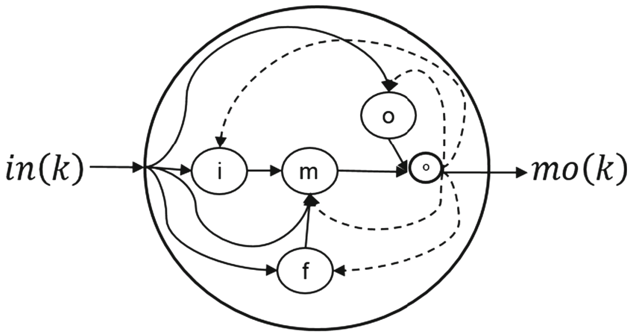

Fig. 2. LSTM cell structure. Dashed lines represent recurrent connections, while solid lines – feedforward connections.

parameters are only in the reservoir readout matrix, all weight matrices from Eqs. 3, 5, as well as 6–8, have to be trained. Thus the total number of trainable parameters of LSTMs and ESNs having the same number of internal units ($n_m = n_R = n_{units}$), as well as the same input and output vector sizes are:

$$n^{LSTM}_{tr.params} = 4n_{units}(n_{in} + n_{units} + 1) + n_{out}n_{units} \tag{9}$$

$$n^{ESN}_{tr.params} = n_{out}n_{in} + n_{out}n_{units} \tag{10}$$

An additional advantage of ESNs is that the matrix containing the recurrent connection weights could be sparse, so that the memory necessary to store its parameters decreases even further. Another difference is found in the training procedure: LSTMs are trained via backpropagation through the time states. The gradient descent algorithm used for the training needs at least several epochs (iterations over all training data) to settle into a possible local error minimum and a lot of memory to keep all intermediate state variables necessary for gradient calculation. ESNs, on the other hand, are trained in one epoch by solving ridge regression equations or using an RLS algorithm, thus finding optimal parameters in one shot. Hence the main advantages of ESNs are the smaller number of parameters and the simpler training algorithm, which can significantly decrease the necessary computational resources. All of this is achieved at the expense of simplifications (constant gates) and the random choice of ESN non-trainable parameters (reservoir connection matrix, its sparsity and leakage rate) that could make its application trickier.

2.3 Bidirectional ESN Architecture

In order to compare ESN performance with that of widely applied LSTM network structures, here we adopt a bidirectional reservoir structure (BiESN) similar to the one proposed in [6] and shown in Fig. 3. The information for the left and right contexts is accumulated in two independent reservoirs – ESN_L and ESN_R.

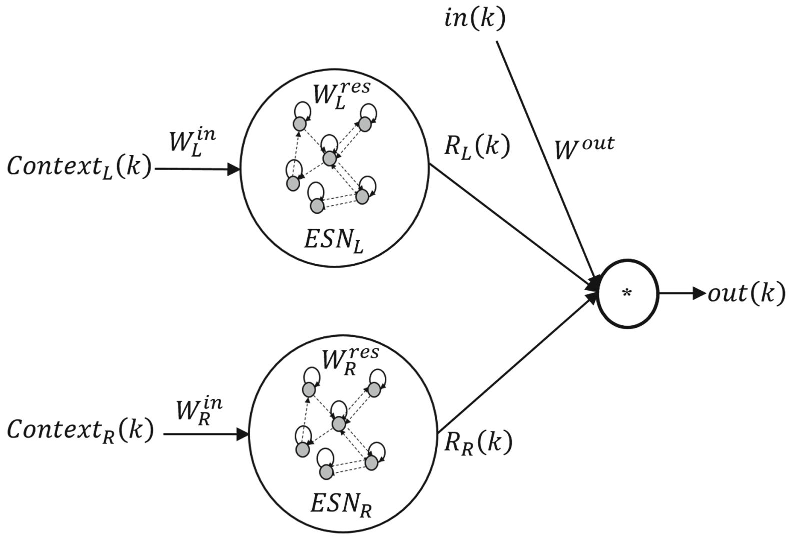

Fig. 3. Bidirectional structure composed of two ESNs for the left and right contexts.

First, for the k-th word in a series of tokens, containing n words in total, we scan the text from left to right (Eq. 11) and from right to left (Eq. 12):

$$Context_L(k) = [in(0), in(1), ...in(k)] \tag{11}$$

$$Context_R(k) = [in(n), in(n-1), ...in(k)] \tag{12}$$

in order to accumulate the left and right contexts respectively. Here $in(t)$ denotes the vector concatenation of the embeddings of the words within the window centered on the t-th word. In the case of a fixed-size context window applied to separate sentences it is necessary to have zero padding at the beginning and end of sentences. However, since the reservoirs themselves are able to accumulate information from context series with indefinite lengths, in our application we scan training/testing text documents from beginning to end (left context) and vice versa (right context). This should allow them to build up a richer contextual representation beyond sentence boundaries. In this sense the architecture behaves differently from the BiLSTM. LSTMs typically read only one sentence at a time, since they are much more difficult to train over such long dependencies.

The two separate reservoir states calculated upon the presentation of the left and right context sequences are obtained as follows:

$$R_L(k) = ESN_L(Context_L(k)) \tag{13}$$

$$R_R(k) = ESN_R(Context_R(k)) \tag{14}$$

Here $ESN_*(Context_*(k))$ stands for the iterative calculation of Eq. (1), as applied to input sequences (11) and (12) respectively. Notice that the parameters for the two ESNs – leaking rates a_L and a_R and the input and reservoir weight matrices W_L^{in}, W_L^{res}, W_R^{in}, W_R^{res} – are initialized independently. The main difference with [6,24] is that in our BiESN structure the output is a linear combination of both reservoirs states and the target word context window, as follows:

$$out(k) = W^{out}[in(k), R_L(k), R_R(k)] \tag{15}$$

The output weights W^{out} are tuned using RLS, so that the BiESN predicts the sense (word sense embedding) of the middle word in the window.

In the case of BiESNs and the corresponding BiLSTM models the number of trainable parameters becomes:

$$n_{tr.params}^{BiLSTM} = 8n_{units}(n_{in} + n_{units} + 1) + n_{out}(n_{in} + 2n_{units}) \tag{16}$$

$$n_{tr.params}^{BiESN} = n_{out}(n_{in} + 2n_{units}) \tag{17}$$

3 BiLSTM for WSD

RNNs have been used in several ways for the task of WSD. One such work is [15], which uses a BiLSTM to solve a *lexical sample task* – that is, the model is disambiguating one word per sentence only. The model is on par (or slightly better) with state-of-the-art systems, but uses no other features apart from input word embeddings. Popov [22] presents two BiLSTM architectures for solving the *all-words task* (i.e. disambiguating all open-class words in a context). Figure 4 is a combined representation of the two architectures that share the same embedding and BiLSTM layers design; the main difference is in the output layers.

The recurrent BiLSTM layer consists of two LSTM cells processing the incoming embeddings in forward and reverse order; the outputs of the forward and backward LSTM cells are then concatenated and fed into the linear output layer to be re-sized according to the training data dimension (the size of the **vocabulary** n_v for *Architecture A* and the size of the *embedding vectors* n_{emb} for *Architecture B* respectively). *Architecture A* has an additional **softmax** layer calculating the probability distribution of the output layer over the vocabulary of synonym sets (**synsets**).

The training data for *Architecture A* consists of one-hot vectors **labels**. The training data for *Architecture B* consists of real-valued *embedding vectors* of word senses obtained from the gold labels (for a description of how these vectors are generated, see [25] and [22]). The embedding model used for the word senses is the same as the model used for the input tokens, as this should facilitate training.

Thus the loss functions subject to minimization during training via the gradient descent procedure are: **cross entropy** in the case of *Architecture A* and **mean square error** between predicted and target word synset embeddings in the case of *Arcitechture B*. In the case of *Arcitechture B* the classification itself is done by choosing the closest possible synset label, as measured in terms of cosine similarity between synset embeddings. Thus, *Architecture A* is a classifier which learns to distinguish directly between synset categories (hence the use of cross entropy – the network is trained to predict probability distributions), whereas *Architecture B* learns to embed the context of usage of each open-class word – in terms of the semantico-syntactic features in the embedding models (hence the mean square error loss function is used – it allows the network to learn to predict each dimension of meaning).

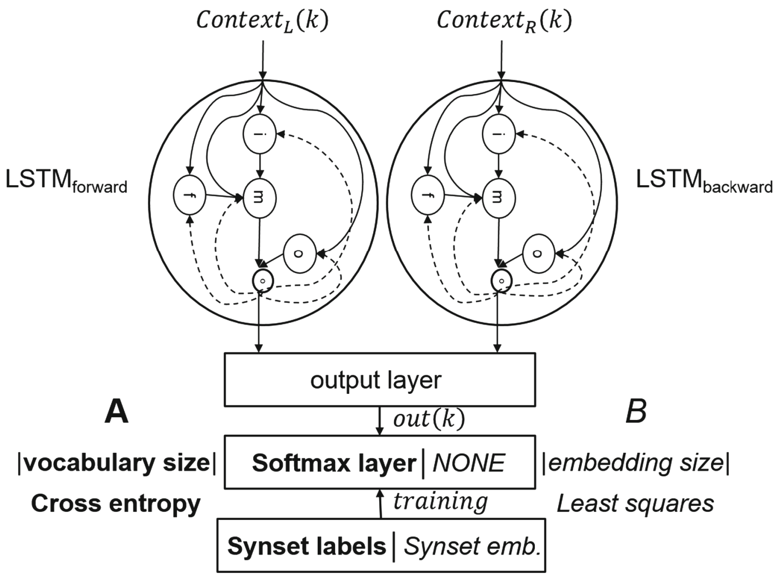

Fig. 4. Recurrent neural networks for WSD: where diverging, the two pathways (*Architectures A and B*) are marked by a separator (/); the left pathway (marked in **bold**) corresponds to elements from *Architecture A*; the right pathway (marked in *italic*) corresponds to elements from *Architecture B*. The *NONE* symbol signals that no corresponding element is present for the particular architecture.

Both architectures are trained on the SemCor data set [20], which is traditionally used for the WSD task (it is used for the ESN models below as well). For evaluation, the concatenation of a number of Senseval and SemEval data sets is used, as described in [22] and here called *WSDEval-ALL*; it contains 7254 word sense annotations in total. The Senseval-2 data set is used as a development set [3].

The word/synset embeddings are obtained via the Word2Vec package [19], using training data obtained from a *pseudo corpus* of artificial sentences generated from an enriched version of the WordNet knowledge graph [4] (for more details see [22]). The model here is slightly richer, as a Wikipedia dump is concatenated to the pseudo corpus prior to training, so that the model would reflect not just the explicit relational knowledge encoded in the WordNet knowledge graph but knowledge about natural language text as well. The same embeddings are used to train the BiESN models as well, as described in the next section.

It is important to note that for some of these recurrent architectures this embedding model is not optimal. Widely used models such as GloVe [21] have been shown to yield results which are closer to the state of the art (e.g. [22]). However, since some of the architectures presented here need access to both word and word sense (synset) embeddings, the model described above is used in all cases in the interest of making a fair comparison.

Table 1. WSD accuracy for the BiLSTM architectures (on the *WSDEval-ALL* data set).

$Model$	n_{units}	$Accuracy$	$n^{BiLSTM}_{tr.params}$
Architecture A	400	66.2	$320\,800 + 1100n_v$
Architecture B	100	62.6	470 800
Architecture B	400	61.3	2 573 200
Architecture B	1000	61.9	11 098 000
MFS		64.8	
IMS-s+emb		69.6	

Table 1 shows the recorded accuracy scores. The BiLSTM architectures both use a single bidirectional layer of LSTM blocks. We report a result with an *Architecture A* type model which has LSTM blocks of size 400; input and output dropout of 0.2 is applied to the blocks, whose matrices are initialized with a random uniform initializer in the range $[-1; 1]$; training is carried out with a SGD optimizer with learning rate 0.2, on batches of 100 training sentences [22]. As for *Architecture B* type models, we report results with LSTM blocks of sizes 100, 400 and 1000, so as to give a fuller sense of how those compare with their reformulation as ESN networks (the rest of the hyperparameters are the same as with the A-model, only the dropout parameter is set to a different value – 0). Additional improvements to the RNNs are possible, such as adding attention mechanisms and training on auxiliary tasks (see [23]), but since our purpose is to provide a relatively fair comparison between the ESN and LSTM mechanisms, we refrain from exploring them. The results from the *IMS-s+emb* system are used here as representative of the state of the art, as reported in [22]; the model is a trained SVM using tailor-made features, which include word embeddings. The most frequent sense (MFS) baseline, which is a very strong one and is also described in [22], is provided as well.

4 BiESNs for WSD

In this section we evaluate the BiESN architecture with linear readout (described previously). For the sake of comparison, we have also trained BiESN models with an additional softmax layer at the output.

4.1 BiESN with Linear Readout

Since the reservoir, being a recurrent structure, has its own internal memory, we set the window size to one word. The left and right contexts are obtained by starting from the beginning and the end of each text in the training corpus. Hyper-parameters of ESN reservoirs were not subject of optimization in present work. Based on our previous experience on WSD with one reservoir [16], we set hyper-parameters of both reservoirs as follows: random input weight matrices W^{in} in range $[-0.5; 0.5]$; recurrent connection matrices W^{res} with 50% sparsity were initialized randomly and were scaled to achieve spectral radius at the edge of stability (1.25), the leakage rate a was set to 1. The following Table 2 presents accuracy scores for trained BiESN models that have different reservoir sizes and, correspondingly, varying numbers of trainable parameters.

Table 2. Accuracy of WSD by BiESN models with linear readout.

n_R	*Accuracy*	$n^{BiESN}_{tr.params}$
100	61.712	150 000
1000	63.201	690 000
3000	64.387	1 890 000
5000	65.049	3 090 000

Comparison with the reported in Table 1 accuracy for BiLSTM *Architecture B* from Fig. 4 (which is more akin to the BiESN structure) shows that the BiESN architecture outperforms BiLSTM significantly. The total number of trainable parameters of the BiLSTM *Architecture B* that achieved highest accuracy of 62.6% ($n_{units} = 100$) was 470800; this corresponds to a BiESN structure with much more (about 770) units. However, further increase of the BiLSTM size decreased WSD accuracy. In contrast, a BiESN with only 100 units and 150000 trainable parameters achieved accuracy of 61.7%, which corresponds to accuracy of 61.9% for the BiLSTM *Architecture B* with 1000 units and about 74 times bigger number (11098000) of trainable parameters. Further increase of BiESN model size led to increased accuracy so that even a BiESN with 1000 units outperforms a BiLSTM with the same number of units (and many more trainable parameters), achieving 63.2% accuracy. Moreover, our best BiESN model achieved accuracy of more than 65% (for $n_R = 5000$). The BiESN architecture thus offers a simpler and easily trainable alternative that is able to beat the difficult MFS baseline and edge towards the state of the art. Moreover, the same kinds of improvements applicable to LSTMs (more expressive embedding models, attention mechanisms, etc.) can be implemented for ESNs as well.

4.2 BiESNs with Softmax Output Layer

Next, we present a few experiments with a BiESN architecture trained to perform direct classification of word senses. It is to a great extent analogous with the BiLSTM *Architecture A* from Fig. 4. The concatenated vector of both reservoirs states and the input vector is processed by a layer of neurons with ReLU activation functions whose dimension corresponds to the size of the dictionary n_v. Then the gradient procedure minimizing the mean square error is used to train the weight matrices of both ReLU and softmax layers positioned after the reservoirs. Finally, the softmax probability distribution is computed and a cross entropy comparison is performed against the gold labels (one-hot vectors) like in the case of BiLSTM *Architecture A*. Table 3 displays the results for the softmax version of the BiESN.

Table 3. Accuracy of WSD with BiESN + ReLU + softmax.

n_R	*Accuracy*	$n_{tr.params}^{BiESN}$
300	64.856	$900n_v$
1000	65.008	$2300n_v$
3000	64.994	$6300n_v$

We are yet to comprehensively optimize the hyperparameters of the network, but even a preliminary comparison shows that BiESNs are able to achieve results similar to those associated with vanilla BiLSTMs. In all cases the accuracy is around 65%, i.e. slightly below the results obtained with BiESNs trained via RLS method on the linear readout (see Table 2). We have to mention that in our experiments with BiLSTM models training is done only in one iteration over all data with constant learning rate (1.0). Therefore using smaller or variable learning rates in combination with multiple iterations could improve WSD accuracy further at the expense of increased training time.

4.3 Linguistic Error Analysis

The error analysis was performed by comparing the outputs of the best-performing BiESN network with the gold annotations. A table with all error cases was extracted, aligning the gold synset and gold synset gloss (i.e. definition) per error with the selected synset and its gloss. In each case the cosine similarity between the gold and selected synset embeddings was included. The totality of error cases were then ordered in descending order, starting from the most similar pairs and proceeding to the least similar ones.

In the range of pair cosine similarity between 0.97 and 0.90 there are 392 examples. Inside this group examples of all content words have been found: nouns, verbs, adjectives, adverbs. We chose highly similar cases in order to inspect whether such small differences reflect genuine distinctions or are rather misleading.

These pairs containing highly similar but non-identical synsets can be divided into several categories. In the first case the non-match is due to the selection of another meaning of the lemma which is very close to the gold sense (in terms of the synset glosses). For example, the noun “week” has three meanings. The gold one is sense number three: “a period of seven consecutive days starting on Sunday”. The selected one is the first sense: “any period of seven consecutive days; it rained for a week”. Another example is the verb “say”, which has 11 senses. The gold one is number 2: “report or maintain”. The selected sense is number 1: “express in words”. Interestingly enough, in another case the same verb is annotated in the gold corpus with sense number 8: “utter aloud”. The sense selected by the system is again number 1:“express in words”, which could indicate that it is in fact a generalization of the other two. Another example is the adjective “new”. It has 10 senses. The gold one is number 3: “original and of a kind not seen before”. The selected one is number 1: “not of long duration”.

In this group we observe that the gold annotation provides some non-first sense of the lemma while the selected sense is always the first meaning. This observation can be used to put forward two hypotheses. The first one is that apparently the partition of meanings in WordNet includes some overlap between the different senses (or even subsumption in some cases). The other hypothesis is that the algorithm tends to select the first meaning whenever there is insufficient knowledge to distinguish between more fine-grained senses of the same lemma.

The second category of errors includes examples where the gold sense and the selected sense belong to different domains. For example, for the lemma “window” the gold sense is number 8 and belongs to the domain of computer science: “a rectangular part of a computer screen that contains a display different from the rest of the screen”. The selected sense is again number 1 but it belongs to a totally different domain: “a framework of wood or metal that contains a glass windowpane and is built into a wall or roof to admit light or air”. Another example is the noun “trial”, which has 6 senses altogether. In the gold annotation it is used in its sense #3 in the law domain: “(law) the determination of a person’s innocence or guilt by due process of law”. In the test data however sense 1 is selected, which is more general: “the act of testing something”.

Within this set of highly similar pairs one can find also mismatches between highly dissimilar senses, irrespective of any domain misalignments. For example, the verb “crawl” has 5 meanings. The gold one is sense 2: “feel as if crawling with insects”. The selected one is, not unexpectedly, sense 1: “move slowly; in the case of people or animals with the body near the ground”. Here the meanings are neither close, nor overlapping. As a subtype here we can consider also senses that differ not so much in their lexical meaning but rather in the participants involved. For example, the verb “kill”, with 15 senses, is in one case annotated in the gold corpus with meaning number 9: “deprive of life”. The selected sense number 1 is: “cause to die; put to death, usually intentionally or knowingly”. In the former case the causing force is underspecified, it could be due to something like a disease. In the latter case it necessarily involves humans.

There are also some grey areas where the domain is not explicitly stated in the definition but it might be presupposed. For example the noun “pound”, with 12 senses, is in the gold corpus given meaning number 2: “the basic unit of money in Great Britain and Northern Ireland; equal to 100 pence". The selected sense is number 1: "16 ounces avoirdupois".

From the point of view of linguistics, and more precisely—that of modeling lexical senses, our survey confirms previous observations—that in some cases the distinct senses per lemma overlap and become virtually indistinguishable for algorithms. Thus, a productive strategy could be merging them in more general senses. From the point of view of computational methods, more knowledge is necessary in order to make these distinctions, especially new training sets which provide good examples of the senses currently used for evaluation. Last but not least, resorting to the first sense (i.e. usually the most frequent one) when no other clues are available remains the preferred strategy for computational models.

5 Conclusions

The results reported in this paper outperform our previous work [16], thereby confirming the ability of the BiESN architecture to capture more context information. Moreover, the results obtained for WSD are comparable with the performance of BiLSTM architectures. One important takeaway is that since the computational time and resources necessary for the training of reservoir architectures are considerably less in comparison with the gradient descent training of LSTM architectures, ESNs could emerge as a more attractive tool in the near future.

We have to point out that since in the present work we do not optimize the ESN hyperparameters (they were set on the basis of some preliminary observations), further work towards their optimal tuning could increase even more the accuracy of BiESN models. In addition to this, since recent reports on deep ESN architectures are also very promising, our next aim will be to apply them as well in the context of WSD. One potential direction for future work is to apply the same approach over different kinds of input constructions: not just over linear text, but also over graphs. In this way we will be able to explore the syntactic and discourse structures of text, in addition to its linear make-up.

Acknowledgments. This research has been funded by the Bulgarian National Science Fund grant number 02/12/2016—*Deep Models of Semantic Knowledge (DemoSem)* and was partially supported by the National Scientific Program "Information and Communication Technologies for a Single Digital Market in Science, Education and Security (ICTinSES)", financed by the Ministry of Education and Science. Alexander Popov was also partially supported by the Bulgarian Ministry of Education and Science under the National Research Programme "Young scientists and postdoctoral students" approved by DCM # 577/17.08.2018.

References

1. Butcher, J.B., Verstraeten, D., Schrauwen, B., Day, C.R., Haycock, P.W.: Reservoir computing and extreme learning machines for non-linear time-series data analysis. Neural Netw. **38**, 76–89 (2013). https://doi.org/10.1016/j.neunet.2012.11.011
2. Cho, K., van Merriënboer, B., Bahdanau, D., Bengio, Y.: On the properties of neural machine translation: encoder-decoder approaches. In: Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, pp. 103–111. Association for Computational Linguistics, Doha, October 2014. https://doi.org/10.3115/v1/W14-4012
3. Edmonds, P., Cotton, S.: SENSEVAL-2: overview. In: The Proceedings of the Second International Workshop on Evaluating Word Sense Disambiguation Systems, SENSEVAL 2001, pp. 1–5. Association for Computational Linguistics, Stroudsburg (2001). http://dl.acm.org/citation.cfm?id=2387364.2387365
4. Fellbaum, C.: WordNet. In: Poli, R., Healy, M., Kameas, A. (eds.) Theory and Applications of Ontology: Computer Applications, pp. 231–243. Springer, Dordrecht (2010). https://doi.org/10.1007/978-90-481-8847-5_10
5. Frank, S.L., Čerňanský, M.P.: Generalization and systematicity in echo state networks. In: The Annual Meeting of the Cognitive Science Society, pp. 733–738 (2008)
6. Gallicchio, C., Micheli, A.: A reservoir computing approach for human gesture recognition from kinect data. In: Proceedings of the AI for Ambient Assisted Living (2016)
7. Gallicchio, C., Micheli, A., Pedrelli, L.: Comparison between DeepESNs and gated RNNs on multivariate time-series prediction. CoRR (2018). http://arxiv.org/abs/1812.11527
8. Graves, A.: Supervised Sequence Labelling with Recurrent Neural Networks. SCI, vol. 385. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-24797-2
9. Graves, A., Schmidhuber, J.: Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Netw. **18**(5–6), 602–610 (2005). https://doi.org/10.1016/j.neunet.2005.06.042
10. Greff, K., Srivastava, R.K., Koutník, J., Steunebrink, B.R., Schmidhuber, J.: LSTM: a search space odyssey. IEEE Trans. Neural Netw. Learn. Syst. **28**(10), 2222–2232 (2017). https://doi.org/10.1109/TNNLS.2016.2582924
11. Hinaut, X., Dominey, P.F.: Real-time parallel processing of grammatical structure in the fronto-striatal system: a recurrent network simulation study using reservoir computing. PLOS ONE **8**(2), 1–18 (2013). https://doi.org/10.1371/journal.pone.0052946
12. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
13. Huang, Z., Xu, W., Yu, K.: Bidirectional LSTM-CRF models for sequence tagging. CoRR (2015). http://arxiv.org/abs/1508.01991
14. Jaeger, H.: Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the echo state network approach. GMD Report 159, German National Research Center for Information Technology (2002)
15. Kågebäck, M., Salomonsson, H.: Word sense disambiguation using a bidirectional LSTM. In: Proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon (CogALex - V), pp. 51–56. The COLING 2016 Organizing Committee, Osaka, December 2016. https://www.aclweb.org/anthology/W16-5307
16. Koprinkova-Hristova, P., Popov, A., Simov, K., Osenova, P.: Echo state network for word sense disambiguation. In: Proceedings of the Artificial Intelligence: Methodology, Systems, and Applications - 18th International Conference, AIMSA 2018,

Varna, Bulgaria, 12–14 September 2018, pp. 73–82 (2018). https://doi.org/10.1007/978-3-319-99344-7_7

17. Lukosevicius, M., Jaeger, H.: Reservoir computing approaches to recurrent neural network training. Comput. Sci. Rev. **3**, 127–149 (2009). https://doi.org/10.1016/j.cosrev.2009.03.005
18. Maass, W., Natschläger, T., Markram, H.: Real-time computing without stable states: a new framework for neural computation based on perturbations. Neural Comput. **14**(11), 2531–2560 (2002). https://doi.org/10.1162/089976602760407955
19. Mikolov, T., Chen, K., Corrado, G.S., Dean, J.: Efficient estimation of word representations in vector space. CoRR (2013). https://arxiv.org/abs/1301.3781
20. Miller, G.A., Leacock, C., Tengi, R., Bunker, R.T.: A semantic concordance. In: Proceedings of the Workshop on Human Language Technology, HLT 1993, pp. 303–308. Association for Computational Linguistics, Stroudsburg (1993). https://doi.org/10.3115/1075671.1075742
21. Pennington, J., Socher, R., Manning, C.: Glove: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543. Association for Computational Linguistics, Doha, October 2014. https://doi.org/10.3115/v1/D14-1162
22. Popov, A.: Word sense disambiguation with recurrent neural networks. In: Proceedings of the Student Research Workshop Associated with RANLP 2017, pp. 25–34. INCOMA Ltd., Varna, September 2017. https://doi.org/10.26615/issn.1314-9156.2017_004
23. Raganato, A., Camacho-Collados, J., Navigli, R.: Word sense disambiguation: a unified evaluation framework and empirical comparison. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Long Papers, vol. 1, pp. 99–110. Association for Computational Linguistics, Valencia (2017). https://www.aclweb.org/anthology/E17-1010
24. Rodan, A., Sheta, A.F., Faris, H.: Bidirectional reservoir network strained using SVM+ privileged information for manufacturing process modeling. Soft Comput. **21**(22), 6811–6824 (2017). https://doi.org/10.1007/s00500-016-2232-9
25. Simov, K., Osenova, P., Popov, A.: Comparison of word embeddings from different knowledge graphs. In: Gracia, J., Bond, F., McCrae, J.P., Buitelaar, P., Chiarcos, C., Hellmann, S. (eds.) LDK 2017. LNCS (LNAI), vol. 10318, pp. 213–221. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59888-8_19
26. Skowronski, M., Harris, J.: Minimum mean squared error time series classification using an echo state network prediction model. In: 2006 IEEE International Symposium on Circuits and Systems. IEEE (2006). https://doi.org/10.1109/ISCAS.2006.1693294
27. Squartini, S., Cecchi, S., Rossini, M., Piazza, F.: Echo state networks for real-time audio applications. In: Liu, D., Fei, S., Hou, Z., Zhang, H., Sun, C. (eds.) ISNN 2007. LNCS, vol. 4493, pp. 731–740. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72395-0_90
28. 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). https://doi.org/10.1016/j.neunet.2007.04.013
29. Twiefel, J., Hinaut, X., Soares, M.B., Strahl, E., Wermter, S.: Using natural language feedback in a neuro-inspired integrated multimodal robotic architecture. In: 25th IEEE International Symposium on Robot and Human Interactive Communication, RO-MAN 2016, New York, NY, USA, 26–31 August 2016, pp. 52–57 (2016). https://doi.org/10.1109/ROMAN.2016.7745090

30. Twiefel, J., Hinaut, X., Wermter, S.: Semantic role labelling for robot instructions using echo state networks. In: 24th European Symposium on Artificial Neural Networks, ESANN 2016, Bruges, Belgium, 27–29 April 2016 (2016). http://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2016-168.pdf
31. Wang, P., Qian, Y., Soong, F.K., He, L., Zhao, H.: Part-of-speech tagging with bidirectional long short-term memory recurrent neural network. CoRR (2015). http://arxiv.org/abs/1510.06168
32. Wang, P., Qian, Y., Soong, F.K., He, L., Zhao, H.: A unified tagging solution: bidirectional LSTM recurrent neural network with word embedding. CoRR (2015). http://arxiv.org/abs/1511.00215
33. Wang, W., Chang, B.: Graph-based dependency parsing with bidirectional LSTM. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics: Long Papers, vol. 1, pp. 2306–2315. Association for Computational Linguistics, Berlin, August 2016. https://doi.org/10.18653/v1/P16-1218