Neurocomputing 502 (2022) 161–176

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Memory degradation induced by attention in recurrent neural architectures

Mykola Harvat *, José D. Martín-Guerrero

IDAL, Departament d'Enginyeria Electrònica - ETSE, Universitat de València (UV), Av. Universitat, SN, 46100 Burjassot, València, Spain

ARTICLE INFO

Article history:
Received 22 July 2020
Revised 22 February 2022
Accepted 18 June 2022
Available online 27 June 2022

Keywords:
Long short-term memory networks
Attention mechanisms
Recurrence
Gate activations
Forget gate

ABSTRACT

This paper studies the memory mechanisms in recurrent neural architectures when attention models are included. Pure-attention models like Transformers are more and more popular as they tend to outperform models with recurrent connections in many different tasks. Our conjecture is that attention prevents the recurrent connections from transferring information properly between consecutive next steps. This conjecture is empirically tested using five different models, namely, a model without attention, a standard Loung attention model, a standard Bahdanau attention model, and our proposal to add attention to the inputs in order to fill the gap between recurrent and parallel architectures (for both Luong and Bahdanau attention models). Eight different problems are considered to assess the five models: a sequence-reverse copy problem, a sequence-reverse copy problem with repetitions, a filter sequence problem, a sequence-reverse copy problem with bigrams and four translation problems (English to Spanish, English to French, English to German and English to Italian). The achieved results reinforce our conjecture on the interaction between attention and recurrence.

1. Introduction

Recurrent neural networks (RNN) and more specifically systems based on Long Short-Term Memory (LSTM) [8] and Gated Recurrent Unit (GRU) [4] were widely used in many state-of-the-art algorithms in the past decade. One specific application of these systems arises in a special type of sequence-to-sequence learning problems where the alignment between the input and the output sequences is not known. Machine translation [13] is a common example of this kind of systems in Natural Language Processing (NLP). This type of sequence-to-sequence learning is commonly solved with an encoder-decoder architecture where two different RNN-based networks are built for the input and output sequences. Even with good learning algorithms and big amounts of data, naive encoder-decoder architectures perform poorly in these problems as the sequence length grows [4]. Therefore, these architectures are combined with attention-based models in order to learn the alignments between the encoder and decoder outputs thus avoiding the need of memorizing an excessive amount of information [10,3].

In the last years, self-attention Transformer-based neural networks [14] have been stablished as the preferred approach for many sequence-based tasks because of their excellent performance [12,14,5]. Although this architecture introduces a wide range of different features which makes it different from the RNN-based models, like multi-head attention or residual connections, the most important difference is the removal of the recurrent architecture and the adoption of a fully parallel one. Transformer-based models are deeper and many times trained in larger amounts of data due to the speed-up achieved with the parallel architecture; current LSTM-based models do not reach the same performance as Transformers in many sequence-based tasks despite being more natural architectures for them [12,14,5].

In the present work we conjecture that one of the possible reasons for the worse performance of the RNN-based architectures is their bad synergy with the attention mechanism. We will empirically explore the hypothesis that the attention connections prevent the recurrent connections from working properly and, in the limit case, they do not allow the recurrent connection to transfer any information whatsoever from one time step to the next one. This will, ultimately, not allow the memory to work properly and hence converts the RNN-based architecture in a simple feedforward network making it very similar to a Transformer-like architecture but without all the extra features that defines it (multi-heading, multi-layers, etc.). For the empirical analysis we propose a novel

* Corresponding author.
E-mail addresses: Mykola.Harvat@uv.es (M. Harvat), jose.d.martin@uv.es (J.D. Martín-Guerrero).

https://doi.org/10.1016/j.neucom.2022.06.056
0925-2312/

architecture of direct input attentions which closes the gap between parallel attention architectures and recurrent attention architectures.

The motivation behind our work is that one of the main reasons to use the Transformer architecture is the lack of parallelizability in recurrent-based architectures [14]. However, this is a pure computational motivation which apparently does not discard recurrent architectures. Although the computational issue to train recurrent networks efficiently may be solved with future hardware and software improvements, according to the results presented in this paper, their natural sequence-based architecture is not enough to improve the performance in sequence-to-sequence tasks because the attention part does not fit properly with the recurrent architecture.

The rest of the paper is structured as follows; Section 2 describes the theoretical motivation of our proposal; the proposed methodology for the empirical analysis is shown in Section 3, and the achieved results in Section 4. Section 5 describes the conclusions drawn from the work and some suggestions for future research. For the sake of brevity, Section 4 focuses on the results achieved in the English-to-Spanish translation problem; Appendix B shows the results for the other seven problems.

2. Theoretical framework

2.1. Encoder-decoder paradigm

The hypothesis presented here is based on some observations about the currently used RNN architectures. In this paper we worked with the original LSTM architecture presented in [8] with the variations proposed in [6] and the full-gradient method of training them [7]. The LSTM setup usually makes use of a sequence of symbols $\mathscr{X} = \{x_1, x_2, \ldots, x_T\}$ as inputs and $\mathscr{Y} = \{y_1, y_2, \ldots, y_{T'}\}$ as outputs, being each output dependent on all past inputs and outputs. The encoder-decoder LSTM based architecture is commonly used, by fitting two separate LSTM networks; the first one, the encoder, takes the input sequence and computes an output vector h_t and a state vector s_t at each time step:

$$h_t, s_t = \text{encoder}(x_t, h_{t-1}, s_{t-1}) \tag{1}$$

The last states obtained from the input x_t, i.e., s_t and h_t, are then used as s_{t-1} and h_{t-1} on the new time step. Inside the LSTM cell architecture, the h_t vector is obtained using the output gate transformation as:

$$h_t = o_t \odot \tanh(s_t) \tag{2}$$

where o_t is the vector of output gate layer activations and $\tanh(\cdot)$ is the nonlinear function used. The decoder can be expressed as:

$$h\prime_t, s\prime_t = \text{decoder}(x\prime_t, h\prime_{t-1}, s\prime_{t-1}) \tag{3}$$

where $x\prime_t$ stands for the decoder inputs (which usually are the previous decoder outputs). The decoder outputs $h\prime_t$ are then transformed by linear or non-linear layers to the space desired in the output. The attention mechanism is applied after the decoder using the encoder outputs for computing, with a scoring function, soft alignments between the current output and all encoder outputs:

$$z_t = \text{score}(\mathbf{H}, h\prime_t) \tag{4}$$

where H is a $T \times d$ matrix obtained from a concatenation of encoder outputs h_i. Here d is the hidden size of the RNN network, and it is the same as the size of the o_t. There are different ways to choose the scoring functions, but the two most common ones are the Loung attention scores [10]:

$$z_t = \mathbf{H}^T h\prime_t \tag{5}$$

and the Bahdanau attention scores, where each score i for each input symbol is computed as:

$$z_t^{(i)} = v^T \tanh(\mathbf{W}_k^T h_i + \mathbf{W}_q^T h\prime_t) \tag{6}$$

here $\mathbf{W}_k$ and $\mathbf{W}_q$ are trainable weight $d \times d$ matrices, and v is a trainable d-dimensional vector [3]. These scores are normalized to a probability distribution:

$$\alpha_t = \text{softmax}(z_t) \tag{7}$$

and then used as weights for obtaining a context vector c_t from the encoder outputs:

$$c_t = \text{diag}(\alpha_t)\mathbf{H} \tag{8}$$

This vector together with the current $h\prime_t$ is used to compute the current output in the decoder network:

$$y_t = \phi([c_t; h\prime_t]) \tag{9}$$

where $\phi(\cdot)$ is usually a dense layer for mapping to the decoder output vocabulary space and $[c_t; h\prime_t]$ is the concatenation of both attention and decoder outputs [3]. It is important to note that in order to retrieve the output, the decoder can use $c_t, h\prime_t$ or both of them, so it is always possible to build a parameter configuration where one of the parts of this architecture, either the attention or the recurrent one, is not used for the output computation (setting the aproppiate weights to 0).

From the encoder point of view, the use of attention adds two extra connections for the encoder output h_t. Eqs. (2)–(4) show that the same h_t vector is used to compute both outputs, creating two different paths to the loss function from there. While in the naive encoder-decoder implementation the only connection between the encoder output and the loss function is the encoder recurrent connection, new connections through the attention layer appear in the attention architecture; these connections act like an additional role for h_t which now is in charge not only to carry the memory information but also the information needed to construct the attention alignments.

2.2. Attention influence on the encoder network

The encoder state vector s_t receives updates at each time step from the current input x_t and past h_{t-1} which decides its change. If the current input is relevant, the update in the s_t will be large; if some information is no longer useful for the future predictions then the forget gate will delete it from the state. The state update in the LSTM architecture is defined by:

$$s_t = f_t \odot s_{t-1} + i_t \odot c_t \tag{10}$$

where f_t is the current forget gate activation, i_t is the current input gate activation and c_t is the candidate state. After this, the state is transformed into h_t through the output gate. Due to the split in h_t, the exploration of the local error flow between successive s_t and s_{t+1} shows that the state vector receives error signals from the next time steps through its input and through the recurrent connection (Fig. 1). Besides, if the attention is used, an additional signal from the attention split also appears. The expression for the error signal becomes then:

$$\delta s_t = (\delta y_t + \delta \xi_t) \odot o_t \odot \tanh\prime(s_t) + \delta s_{t+1} \odot f_{t+1} \tag{11}$$

where the δy_t is referred to the gradient accumulation until that point of the network and $\delta \xi_t$ is the delta referred to the recurrent input gradient. Our conjecture works around the interaction of the recurrent gradient $\delta s_{t+1} \odot f_{t+1}$ and the output gradient $\delta y_t \odot o_t \odot \tanh\prime(s_t)$. Note that in a non-attention setup the state gra-

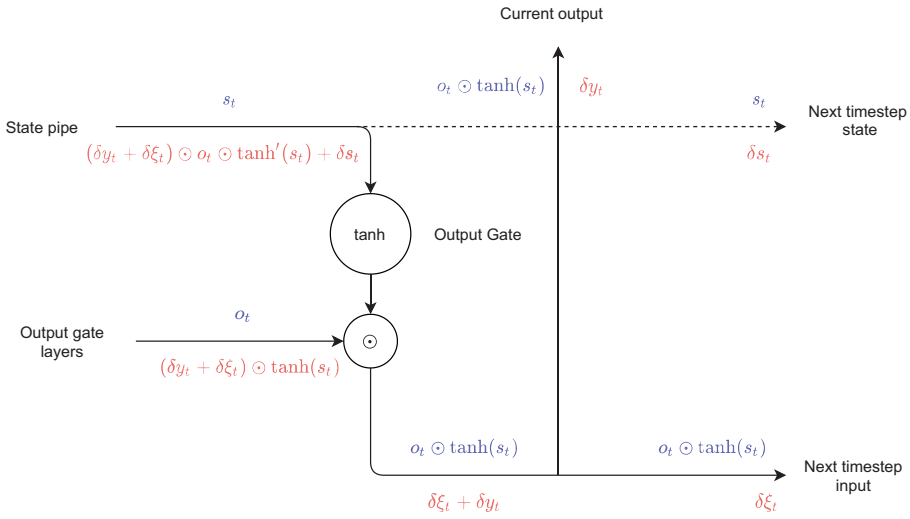

Fig. 1. Output gate and the connections to the next time step, corresponding with the forward pass (blue) and backward pass (red) of the LSTM cell. The full gradient of the current time step depends on three different gradients: the next input gradient $\delta \xi_t$, the next state gradient δs_t and the current output gradient δy_t.

dient only receives the $\delta s_{t+1} \odot f_{t+1}$ update. If we expand the latter expression for several time steps, and being $\delta c_t = (\delta y_t + \delta \xi_t) \odot o_t \odot \tanh\prime(s_t)$:

$$\delta s_t = \delta c_t + \delta s_{t+1} \odot f_{t+1} \tag{12}$$

$$= \delta c_t + (\delta c_{t+1} + \delta s_{t+2} \odot f_{t+2}) \odot f_{t+1} \tag{13}$$

$$= \delta c_t + (\delta c_{t+1} + (\delta c_{t+2} + \cdots) \odot f_{t+2}) \odot f_{t+1} \tag{14}$$

$$= \delta c_t + \delta c_{t+1} \odot f_{t+1} + \cdots + \delta c_T \odot \prod_{i=t+1}^{T} f_i + \delta s'_1 \odot \prod_{i=t+1}^{T} f_i \odot f'_1 \tag{15}$$

Therefore, all terms of this expansion (except the first one) will tend to vanish due to the succesive multiplication by the forget activation which is bounded in $[0, 1]$. In a naive LSTM model only the term $\delta s\prime_1 \odot \prod_{i=t+1}^{T} f_i$ is present and it transfers the information from the output to update the encoder gates. In the attention model, δc_t is unaltered by the forget gate, being the term that mostly contributes to the upgrade in the gate layers. A direct consequence of this, is that the gate activation will be corrected based on the attention gradients more than the recurrent gradients. Our conjecture here is that the attention state management (which depends on the gates) and the recurrent state management differ.

In order to define the state management by the attention-based networks, attentional distributions in real-world problems can be considered. For instance, in many problems, like translations, the α_t distribution is closer to a discrete distribution centred in some specific time steps than to a uniform one. More formally we can say that the entropy of this distribution tends to be lower than a random uniform distribution on the attention weights [3]. This is even often used as a validation method for the translation problem: if the network aligns the decoder outputs to the encoder inputs properly, then the network is performing the translation in a correct way [3,10]. It has also been employed for the construction of new attention architectures that favour these distributions [11]; however, in order to get these distributions, the score from Eq. (4) must change between some time steps. If, for example, we have two consecutive words where the first must be aligned but not the second, we need the vectors h_{i-1} and h_i to be different so that the final scores are also different. Applying Eq. (4) to two consecutive h_{i-1} and h_i, we get:

$$\alpha_t^{i-1} = \text{softmax}(z_{i-1}) = \text{softmax}(h_{i-1}^T h'_t) \tag{16}$$

$$\alpha_t^{i} = \text{softmax}(z_{i}) = \text{softmax}(h_{i}^T h'_t) \tag{17}$$

where t indicates the output sequence time step and i the input sequence time step. As $h\prime_t$ is the same for both align scores, the information of the change between alignments is carried entirely by the h_{i-1} and h_i. Therefore, if in the final alignment, α_t^{i-1} is different from α_t^i, which usually happens in translation tasks, then this implies that h_{i-1} must be sufficiently different from h_i to enable the attention layer to transform it to the proper final α.

However, this fact is contrary to the philosophy of the encoder as information accumulator. Although theoretically they are not completely incompatible, that is, the state can change but still accumulate information, our hypothesis is that this change will be higher only to fulfil the attention layer needs for building the appropiate alignment. As we will show experimentally in Section 4.2, this situation will lead to a faster vanishing in the recurrent gradient making the encoder to work rather as a feedforward network than a recurrent network. This will make the recurrent connection to carry less weight in the final prediction thus supporting the fact that architectures without any recurrence (like Transformer networks) can achieve similar or even better results than their recurrent counterparts.

An example of this rationale with a hypothetical situation in translation would be a time step where the previous time step had a word like “go” and the next word is “to” with Spanish translations “ir” and “a”, respectively. In the naive (without attention) encoder-decoder architecture the update on the state for the word “to” should not be necessary big as it can be predicted from the past word easily (“go to” is a common bi-gram in the English language), but the decoder must align “ir” to “go” and “a” to “to”; therefore, in the attention architecture, the state will receive an extra update to differentiate both states for the attention layer which would introduce a state change in the forward pass. However, as the state is limited in size, this will only be achieved by erasing information in the state vector through the forget gate. We simulate a similar situation to this example in one of our sequence problems in Section 4.3.

Note that this situation is not dependent on the final model performance as long as the attention distribution premise follows, that is, the network reaches a training state where the distribution of the align vector has low entropy. Although using or not using the recurrent connection may not necessarily lead to a worse performance, the point here is that if that connection is not working as it is expected to be, or even, not working at all then this connection

does not lead to any architectural benefit and architectures without this connection can be taken into account. Our main interest in this study is to analyse the differences between how the state should work in naive recurrent architectures and how it is working in attention ones.

In conclusion, we will follow the next argument: due to the usual alignment distributions, we conjecture that the state vector should change between time steps. Although several factors can influence the difference between h_{t-1} and h_t, the forget activation function is sufficient to justify the alignment distribution. If this activation is closer to 0 than to 1, it is not only changing the state on the forward pass through Eq. (10), but it also prevents the LSTM from learning on the backward, using the last term in Eq. 12). Although the state change must not necessary mean memory loss, in a limited memory setup, as the one in usual RNN architectures where the size of the embedding representation and the memory is usually similar or the same, this will lead to an underuse of this memory. As we will show, this underuse is mainly due to an excessive forget gate activation that we interpret in terms of memory degradation.

Our proposal is that recurrent architectures are not using the recurrent connection as it is expected, what justifies, in the limit situation, the usage of parallel architectures, like Transformer, where this connection is not present.

2.3. Direct input attentions

Our conjecture will be tested by making use of model configurations that are similar in all but the attention part. The training times and the overall performance are different when using attention or using the naive encoder-decoder architecture. Therefore, in order to compare two almost equivalent models we designed a novel, as far as we know, direct input attention encoder-decoder network. This model uses the standard encoder-decoder architecture with a LSTM cell and attention, but in this case the attention is not connected to the encoder outputs but to its inputs. The expression for the score vector for each time step using a dot-product attention becomes:

$$z_t = \mathbf{X} h\prime_t \tag{18}$$

where $\mathbf{X}$ is a matrix of input vectors set in a row-wise manner. This configuration implies that the encoder output connection and the whole encoder network are not supervised by the attention layer. We observed that this configuration performs similar, or even better in some cases, than the classical version of the attention layer. We consider three situations, namely, with attentions connected to the encoder output, with attention connected to encoder inputs and without attentions. We expect the last two situations to provide similar results in our tests (more state similarity and less gate activations). The three architectures are depicted in Fig. 2.

3. Methods

3.1. Exploration of the encoder LSTM function

The main challenge to test the different architectures and attention scores is to come up with a suitable metric. It is common to perform ablation or sensitivity-to-noise studies when testing the usage of parts of a neural architecture. There are two main reasons why we cannot use such techniques in the present paper:

1. The architectures we test have different performance on our test tasks, which means that any performance measure will not be comparable between them.
2. As noted in Section 2.2 the performance is not a relevant measure for our theoretical setup, being enough for the alignments distribution to have a low entropy.

Therefore, we focus on Eqs. (12) and (16), that were the basis of our conjectures. In particular, Eq. (16) shows that the scores are different if h_t and h_{t+1} are different. And, in turn, the sufficient condition for this (note that not necessary condition because the output gate can also change the state) is that s_t and s_{t+1} are different, too.

The state difference was assessed by means of the cosine metric similarity between each s_i and s_{i+1}. Higher similarity between contiguous states means less average change between state vectors and more unchanged information in that time step:

$$\Delta_i = \frac{s_i^T s_{i+1}}{\|s_i\| \|s_{i+1}\|} \tag{19}$$

where $\|\cdot\|$ stands for the L2-norm. It is possible to have information retention without state similarity if the future layers of the network (e.g., decoder output layers) can map two arbitrarily different states to the same output. However, in our case, according to Eq. (12) and taking into account that the recurrent update tends to be degraded by the reset gate, and most of the influence on the gate gradients come in a feedforward way, together with the fact that both the attention model and the naive model deal with state similarity in a similar way, we can conclude that the metric is actually informative of the state usage. Also, note that the resolution of this metric is dependent on the state size. For a sufficiently large state size, the LSTM will be able to work without using most of its neurons, changing them randomly and keeping the Δ values low. We show this effect in a naive encoder-decoder architecure in our first experiment in Section 4.1. The idea of this metric is to set a specific state size where the problem is learnable for both, the input attention models and the standard attention models, so that it is possible to evaluate the differences between them and the naive model for that specific state size. As the distribution of all Δ does not describe the one-to-one differences between architectures, an alternative metric was taken into account:

$$\Theta = \sum_{s_i \in S} \frac{\mathbb{I}[\Delta_i < \Delta\prime_i]}{N} \tag{20}$$

where S is the set of all sequences of states for the input implementation and the classical implementation, $\mathbb{I}[\cdot]$ is an indicator function, Δ and $\Delta\prime$ are the cosine similarity obtained for both models and N is the total number of timesteps used to estimate the Θ value. This basically measures how many state differences, in a one-to-one manner (over the same samples) are lower in the naive and input models compared to the attention model.

As mentioned earlier, in order to validate this metric we performed an additional experiment with a naive encoder-decoder architecture. We selected one of the test problems and trained encoder-decoder LSTMs for different state size values on this problem. When the state is much larger than the size needed to achieve the maximum accuracy on the problem we can suppose that most of the state values are redundant. In order to characterize this behaviour we represented the progression of Δ_i distributions for different state sizes. This representation characterized the distribution in situations ranging from the full state usage, when state size is not enough to achieve high accuracy to low state usage, when the state size is much higher than the problem complexity. We expect that attention architectures will have a similar behaviour as that of naive encoder-decoder architectures with low state usage.

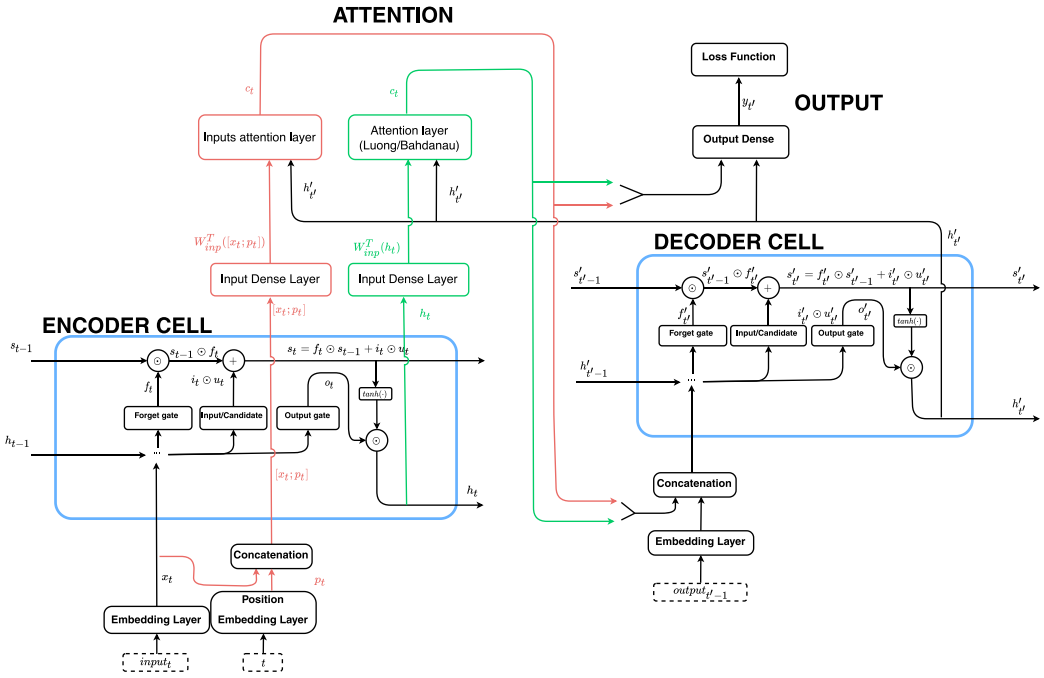

Fig. 2. Representation of the architectures used in the experiments. The attention connection differs from the standard and input attention setups, red for input attentions and green for standard attentions. One time step from the encoder and one time step from the decoder is represented. For the naive architectures no attention module is used.

We also explored the gate activation distributions in the test data. Our interest was to find out if the activation values, specially for the forget gate, were able to explain the differences in the state vector, as a high forget gate activation is a sufficient condition to have different states between consecutive time steps. Specifically, we made use of a box-plot to compare the activation distribution graphically for different time steps on several sample sequences.

On top of the distributions, we were also interested in measuring the amount of information kept by each of the architectures from the first state to the end of the source sequence processing. Expanding Eq. (10), and considering the final factor of Eq. (12), we can find that the product of the forget gate vectors is the one controlling the information flow from the first encoder state to the first decoder state in the forward pass, and vice versa in the backward pass. Therefore, we studied this factor averaged on a trained architecture among the test samples:

$$\gamma_t = \frac{1}{d}\sum_{i=0}^{d}\prod_{t=1}^{T} f_t^i \tag{21}$$

where d is the dimensionality of the state vector and T is the sequence length. The idea behind this measure is to analyse if the attention network is more prone to forget past information than the other architectures. As the activation value information is highly dependant on the size of the state vector, the experiments to obtain γ_t were repeated for different state sizes. We expect that, as the state size becomes bigger, the distributions will be more spread between $[0, 1]$.

3.2. Datasets and model parameters

In order to test the robustness of our proposals, we tested the models in different environments, namely, where the sequence symbols are independent, where some sequences symbols are irrelevant for the output and finally a real-world problem which mixed the previous environments. Eight different problems, where the alignment vector distribution has low entropy (close to sparse), were considered:

1. Sequence-reverse copy problem: the goal is to copy a randomly drawn sequence of symbols from a vocabulary into reverse order. This problem can be considered as easy for the LSTM architecture as it can reach a 100% accuracy even without attention.
2. Sequence-reverse copy problem with repetitions: the same problem but adding repetitions to the sequence of symbols.
3. Filter-sequence problem: similarly to the sequence-reverse copy problem the goal is to copy a sequence but in this case the copy must not include some of the symbols that are predefined in a tabu vocabulary. This problem adds the factor that now some of the inputs are completely redundant creating a limited version of the previous problem.
4. Sequence-reverse copy problem with bigrams: this problem is similar to the sequence-reverse copy problem but some of the symbols are predefined highly-frequent bigrams, and hence, some pairs of symbols appear together with high frequency. This problem is evaluated separately as it is a particular case of the translation example exposed in Section 2.2. This is the case where the sequence can be efficiently learnt without state change but we expect that the attention will try to align the common bigrams even at the cost of changing the state and not exploiting the recurrent connection memory.
5. Four translation problems: we selected an English-to-Spanish, English-to-French, English-to-German and English-to-Italian translation datasets in order to perform the experiment in a real-world problem. The datasets were obtained from [2]. We used a small dataset like this one mainly because of the training time, as we were interested in carrying out many experiments. As these datasets are comparable to any large-scale translation dataset except from the sequence lengths and sample sizes, we expect the achieved results to be generalizable other sequence-to-sequence tasks as long as the attention distribution premise follows. Not only does the translation problem offer all the features seen in the previous problems but it is also a mapping task between source and target language where the alignment is not obvious.

Table 1
Examples of the problems used for testing. Generic symbols s_i and $s\prime_i$ are used for representing the problem tokens. The Sequence-reverse copy problem with bigrams is the same as Sequence-reverse copy problem but with some bigrams of symbols set to appear together frequently.

Problem name	Sources sentence	Target sentence
Sequence-reverse copy problem	$s_1\, s_2\, s_3\, s_4\, s_5\, s_6\, s_7$	$s_7\, s_6\, s_5\, s_4\, s_3\, s_2\, s_1$
Sequence-reverse copy problem with repetitions	$s_1\, s_1\, s_1\, s_2\, s_2\, s_3\, s_3$	$s_3\, s_3\, s_2, s_2\, s_1\, s_1\, s_1$
Filter-sequence problem	$s_1\, s_2\, s_3\, s_4\, s_5\, s_6\, s_7$	$s_2\, s_4\, s_5\, s_6$
Translation ENG-SPA/FRA/GER/ITA problem	$s_1\, s_2\, s_5\, s_1\, s_3$	$s\prime_3\, s\prime_7\, s\prime_2\, s\prime_1$,

We generated 30,000 different sequences of length between 5 to 18 for the first four problems and extracted the first 30,000 sentences in the translation problem datasets. A random split of 75%/25% was used for constructing training/validation datasets for all eight problems. An example of input and target sequences for each sequence task is shown in Table 1.

All the implementations were carried out using Tensorflow python module [1]. The input sequence was padded and transformed from a one-hot symbol representation in $\mathbb{R}^{|V|}$, where V is the encoder input vocabulary into a $\mathbb{R}^d$ embedding space, being d the embedding size:

$$x_t^{emb} = \mathbf{W}_{emb}^T \mathbb{1}_{[x_t=\mathbb{i}]} \tag{22}$$

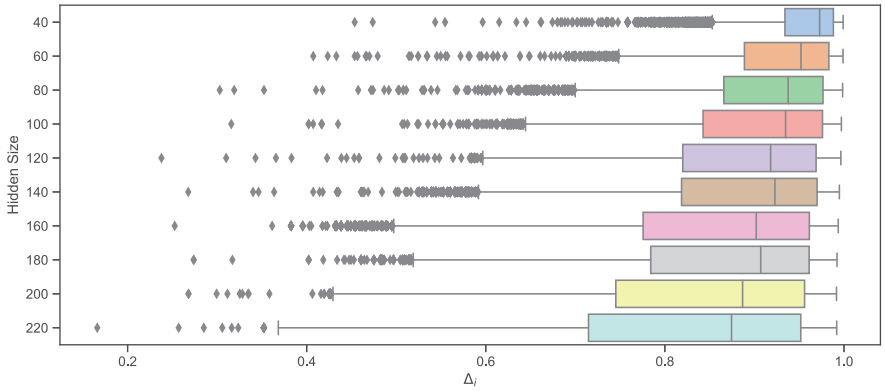

Fig. 3. Δ values for different state sizes on the sequence-reverse copy problem with the naive encoder-decoder architecture. We can observe that as the state size grows, the distribution of the state differences become more spread suggesting that not all the state is used in building the output.

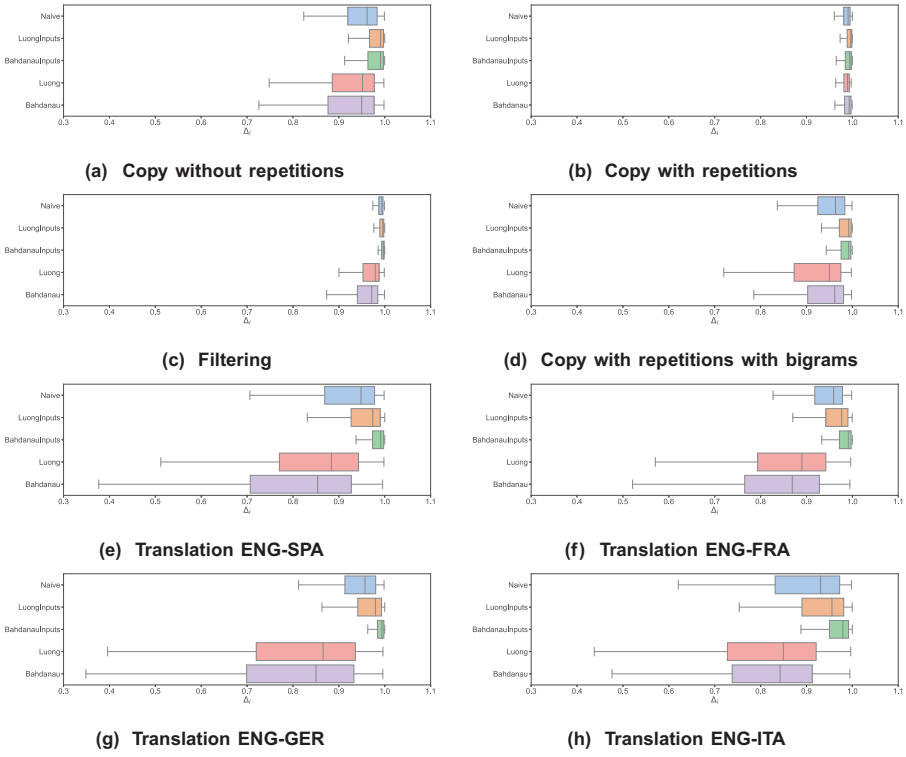

Fig. 4. Box-plots representing the distributions of Δvalues (cosine similarities) between contiguous states, for all eight problems and the five studied models. Outliers, which represented around 5% of the samples of each problem, were not included for a better readability.

where $\mathbf{W}_{emb}$ is a $|V| \times d$ embedding matrix. Here j is a specific index in the vocabulary corresponding to the sequence symbol x_t. The size of the embedding vector was selected to be the same as the size of the hidden dimension in the LSTM cells, which was selected to be 50 for the first four problems and 100 for translation problems. In the case of the input attention model we appended a position vector when computing the attention scores for a proper identification of the different input positions even if the symbols were equal; we

Table 2
Mean values and standard deviations between experiments of Θ for the experiments carried out on the different problems.

	Luong Attention			Bahdanau Attention		
Problem name	Naive	Luong Inputs	Bahdanau Inputs	Naive	Luong Inputs	Bahdanau Inputs
Sequence reverse copy problem	0.77 ± 0.05	0.99 ± 0.01	0.97 ± 0.03	0.72 ± 0.11	0.99 ± 0.01	0.97 ± 0.02
Sequence reverse copy problem with repetitions	0.55 ± 0.09	0.89 ± 0.08	0.83 ± 0.04	0.32 ± 0.09	0.76 ± 0.16	0.61 ± 0.05
Filter-sequence problem	0.97 ± 0.02	0.92 ± 0.02	0.96 ± 0.02	0.98 ± 0.01	0.94 ± 0.03	0.97 ± 0.02
Sequence-reverse copy problem with bigrams	0.82 ± 0.08	1.00 ± 0.00	0.99 ± 0.01	0.59 ± 0.16	0.97 ± 0.03	0.97 ± 0.05
Translation problem ENG-SPA	0.87 ± 0.10	0.96 ± 0.04	1.00 ± 0.00	0.95 ± 0.04	0.98 ± 0.02	1.00 ± 0.00
Translation problem ENG-FRA	0.95 ± 0.04	0.99 ± 0.02	1.00 ± 0.00	0.97 ± 0.02	0.99 ± 0.02	1.00 ± 0.00
Translation problem ENG-GER	0.98 ± 0.02	1.00 ± 0.01	1.00 ± 0.00	0.99 ± 0.01	0.99 ± 0.02	1.00 ± 0.00
Translation problem ENG-ITA	0.90 ± 0.03	0.92 ± 0.04	1.00 ± 0.01	0.89 ± 0.04	0.90 ± 0.03	0.99 ± 0.02

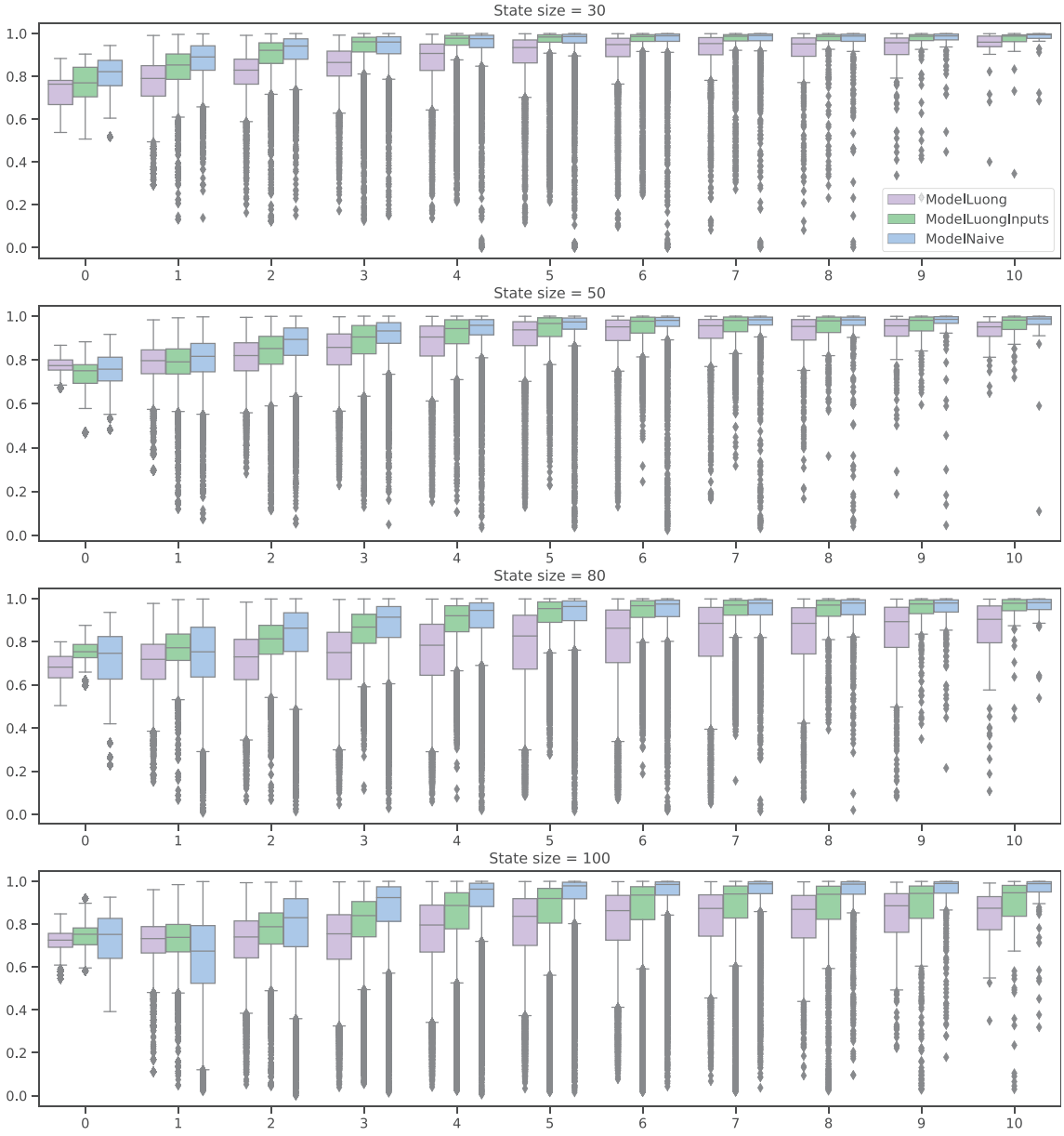

Fig. 5. Translation English-to-Spanish problem: box-plots representing the distribution of the forget gate activations for three models (naive, Luong attention and Luong input attention) at each time step of the sequence.

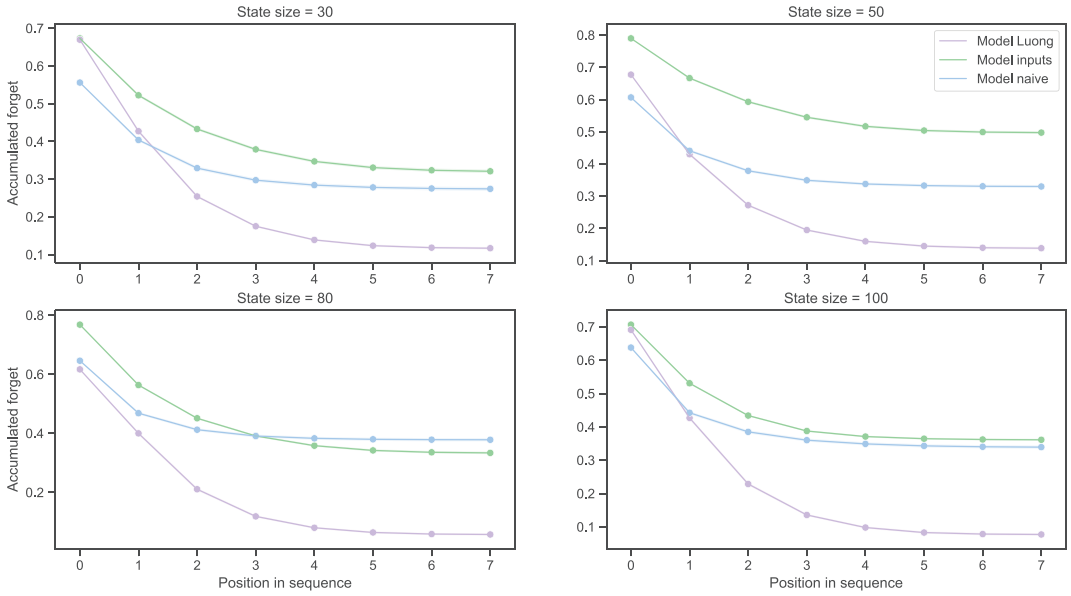

Fig. 6. English-to-Spanish translation problem: γ_t values for three models (naive, Luong attention and Luong input attention) at each time step in the sequence.

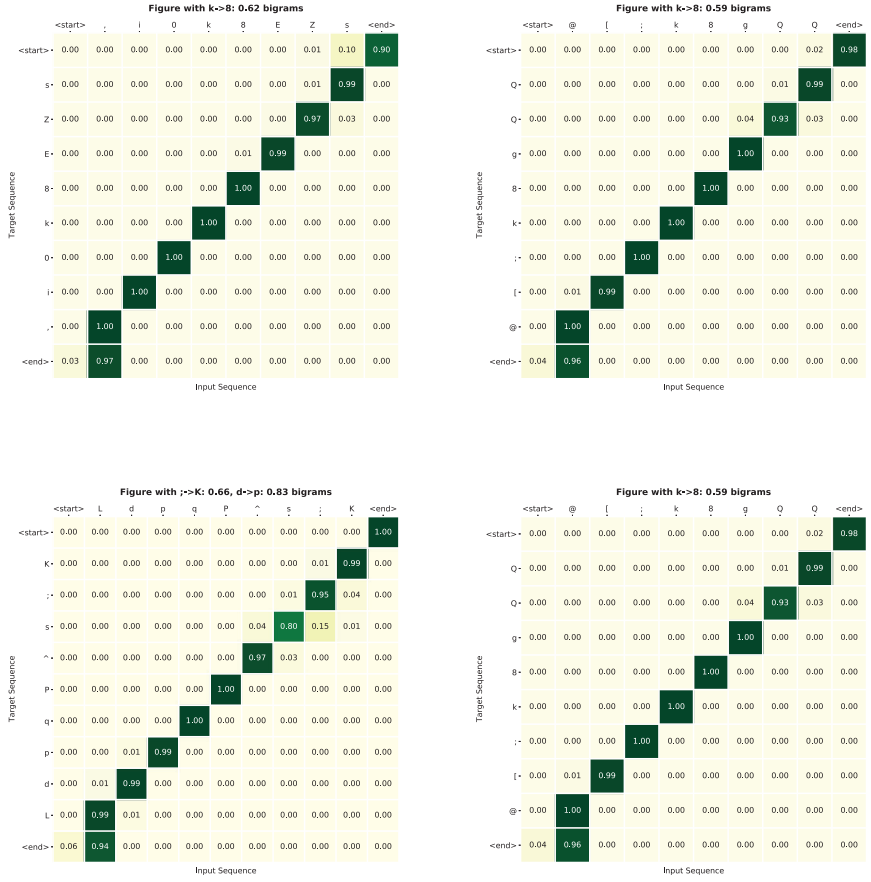

Fig. 7. Heatmaps of attention distributions for the sequence-reverse copy problem with bigrams using Luong attention. The specific symbols which are set a priori in the dataset as bigrams are shown in the title of the plot together with the cosine similarity of their LSTM states.

used a size of 10 for the first four problems and 20 for the translation problems. These appended vector was not used in the LSTM input, only for the computation of the attention scores, as shown in Fig. 2. A linear layer was used for mapping the augmented vector to a d dimensional space for the dot products of the attention layer. This layer was also used in the attentional setups. The size of d was equal to the hidden size of the LSTM network. Then, the final score function for the standard and input attention was:

$$z_t = \mathbf{S}\mathbf{W}_{inp}^T h_t \tag{23}$$

where $\mathbf{W}_{inp}$ is the linear layer transformation matrix with the size of $d \times d$ or $d \times d + m$ depending on using the Luong attention or input attention, respectively. Here m is the size of the position vector. Also $\mathbf{S}$ is the memory matrix which is $\mathbf{H}$ (a matrix of encoder outputs) for the Luong approach and $[\mathbf{X}; \mathbf{P}]$ (a matrix of concatenation of input embeddings and position embeddings) for the input attention approach. Although we used the parameter values exposed earlier, our results were stable for large choices of parameters as long as the attention distribution hypothesis followed. The output in the decoder was obtained by concatenating the context vector c_t and the decoder output h_t and using a linear layer to the target vocabulary space with a softmax activation.

A teacher-forcing procedure was used for training, based on feeding the correct past output symbol to the next decoding time step as input. We also used this procedure in the decoding. Adam optimization algorithm [9] was used for training the stochastic gradient descent with a batch size of 32 samples and a learning rate of 0.001. We used an early stopping criterion finishing the training if the train loss had not changed more that 0.01 for more than 300 time steps or the training steps reached $3,500$ for attentional architectures and $10,000$ for the naive encoder-decoder architectures. We saw that our results were stable for many choices of stopping criteria once the alignment criterion was

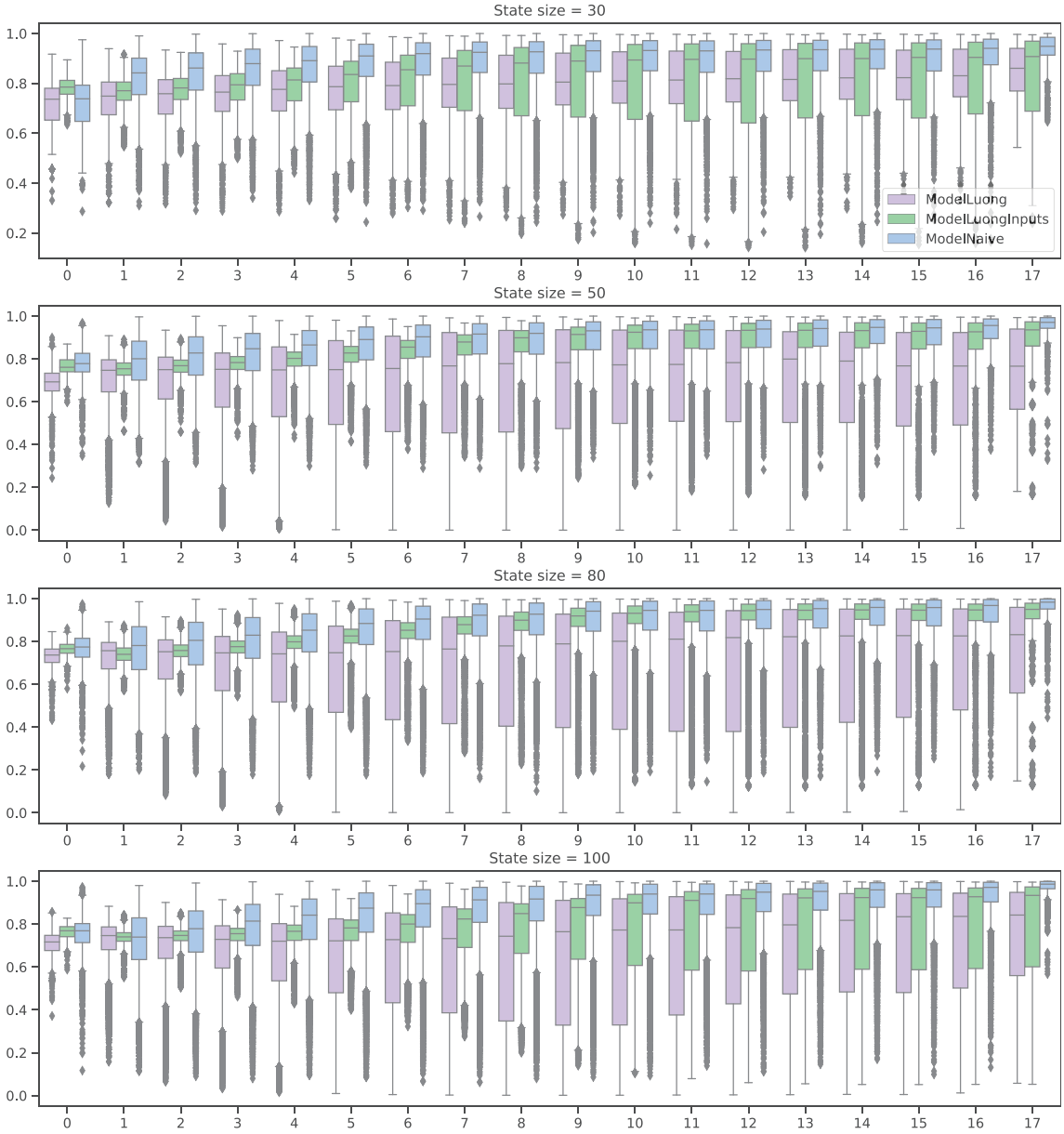

Fig. 8. Sequence-reverse copy problem: box-plots representing the distribution of the forget gate activations for three models (naive, Luong attention and Luong input attention) at each time step of the sequence.

met, which usually occurred around the first 500 time steps. This fact was mainly due to the fact that our experiments depend on the alignments and not on the final performance scores.

For the forget gate distribution experiments we expanded the study on hidden sizes and repeated the same procedure for the following hidden sizes: 30, 50, 80 and 100. For representation purposes we only report these distributions for naive encoder-decoder, Loung standard attention and Luong input attention architectures, but similar results were achieved for the Bahdanau attention score function.

4. Results

Two main results are shown in this section. Firstly, an assessment of the state change in the five analysed models, namely, a standard attention model, our proposal based on an input attention, both with Loung and Bahdanau attentions, and the naive encoder-decoder model. Secondly, an analysis of how these changes are produced, basically analysing the activation of the forget gate and the cumulative product of forget gate activations.

4.1. State difference evaluation

Differences in the state are based on the factor Δ, described in Eq. (19). Δ values were tested on 10,000 samples for the copy reverse problem on the naive encoder-decoder architecture. The results can be found in Fig. 3, that shows that the bigger the state size, the more spread the distribution of Δ as the model reaches high-accuracy perfomances, situation in which we can assume most of the state is not used in the output computing.

Fig. 4 shows Δ values for all models in all eight problems. The representation was computed from 10,000 different test sequences from each problem and from 10 independent model trainings for each problem, i.e., 100,000 samples for each model and problem. High values of this metric mean that the contiguous states are similar and, therefore, the change between the two steps is low. The state vector was obtained for non-padding samples, as in the padding samples the processing is masked and the state is directly copied to the next time step. As Fig. 4 shows, the input attention and the naive model keeps the states more similar than the standard attention model. This supports our theoretical

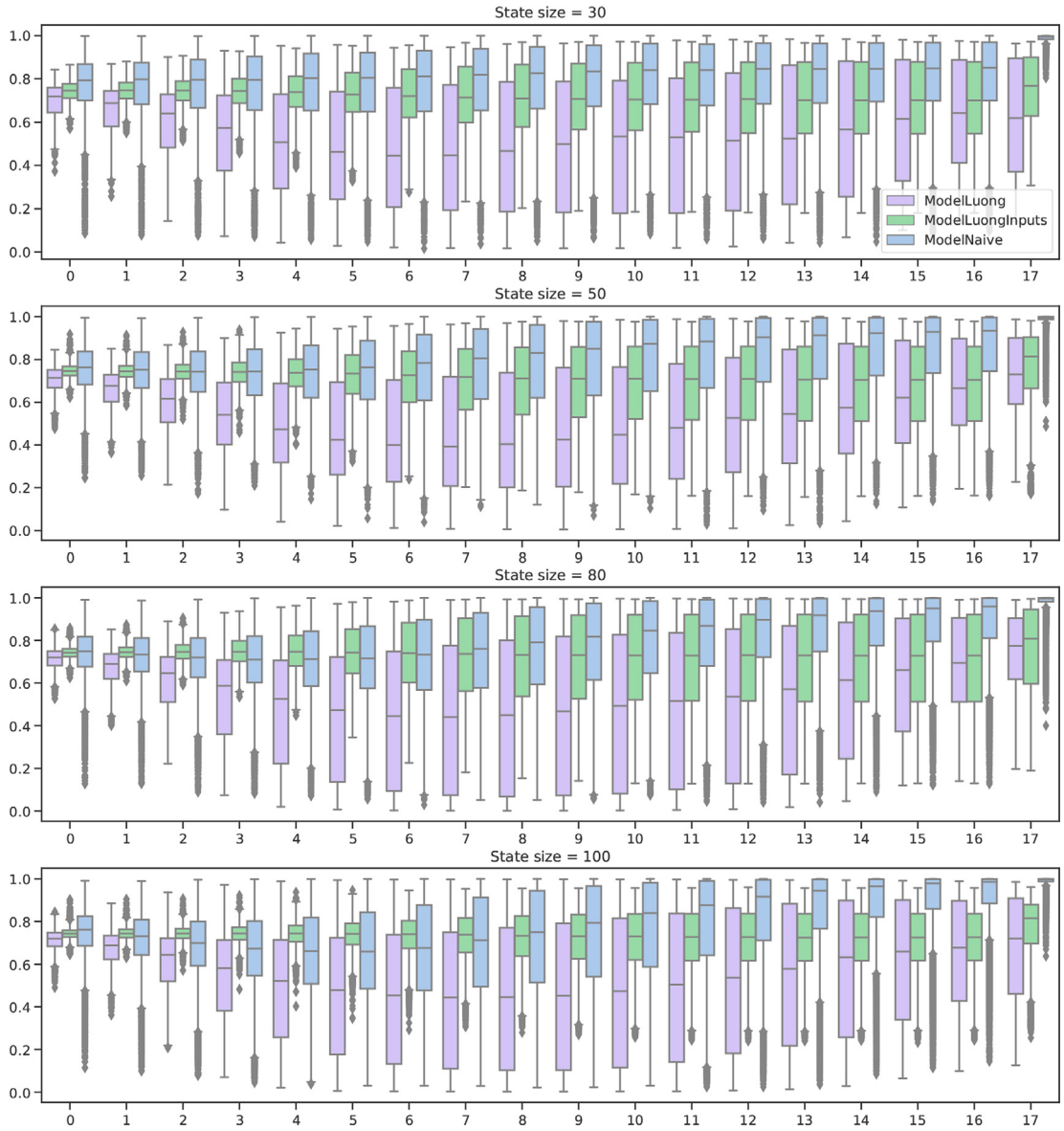

Fig. 9. Sequence-reverse copy problem with repetitions: box-plots representing the distribution of the forget gate activations for three models (naive, Luong attention and Luong input attention) at each time step of the sequence.

proposal described in Section 2.2. Note that Luong and Bahdanau standard models have the distribution of the state changes similar to the naive encoder-decoder architecture with a high number of state size. We can deduce from this observation that most of the state is not used for the final output decoding. It is also remarkable the need of the alignment criteria for this analysis, that is, the alignments must have low entropy for this behaviour to happen, as shown in Fig. 4b. In this problem, repetitions of the same symbol are present, and hence, the alignments do not have low entropy for the repeated symbols (the model only aligned properly the first of the symbols), transferring the functionality of the architecture to the memory instead of to the attention.

As the graphical evaluation contains some overlapping between the box whiskers, the metric Θ proposed in Eq. (20) was also considered; the mean values of this metric and the corresponding standard deviations are shown in Table 2. For the first problem, we have that in a one-to-one fashion, 99.0% of Δ_i values were lower for the Luong input attention model than for the Luong attention model. In general, for most cases of the eight problems, the difference between two contiguous state values was higher for the standard attention model than naive or input attention models. This suggests that the attention connection to the output of the encoder, with independence of the problem, forces the state to change for fulfilling the alignment requirements, and this involves, in turn, that the information inside the state degrades faster due to the new information update in the standard attention models. Furthermore, the standard deviations indicate that these results are stable among different trainings. These experiments were also conducted with different hidden sizes obtaining similar values which are not reported here.

4.2. Gate activation evaluation

The state change can be obtained by two main ways: either by an update vector $i_t \odot c_t$, or by the forget gate activation f_t. Although we did not find relevant differences in the values of $i_t \odot c_t$ between the models, interesting disagreements did appear in f_t vectors. By way of example, we focus on the translation problem; the equivalent results for the other seven problems, as shown in appendix B. Fig. 5 shows the activation distribution of the forget gate values for four different state sizes and the naive, Luong attention and Luong input attention models. The forget values of the naive and the

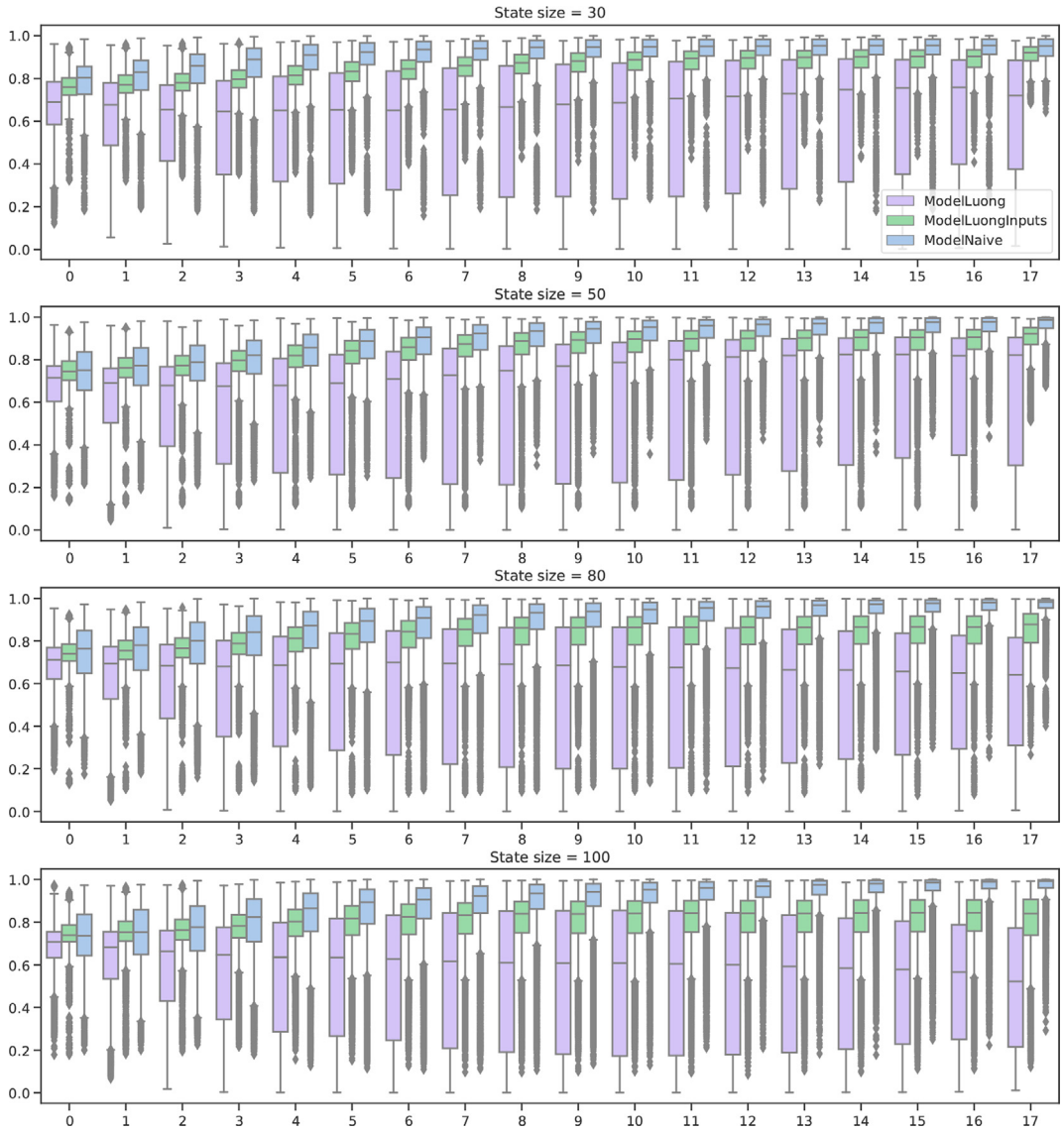

Fig. 10. Filter-sequence problem: box-plots representing the distribution of the forget gate activations for three models (naive, Luong attention and Luong input attention) at each time step of the sequence.

input attention models have a similar convergence to 1 as the sequence position grows for various hidden-size models. This behaviour is due to the fact that the state receives all the forget activation products from the first time steps, and, the encoder, in order to keep this information for decoding, must ensure that it is not completely erased. In contrast, the Luong attention yields a much wider range of forget values, supporting the idea of a more frequent state change present in those models as most of the responsability of building the output falls on the attention connection and not on the recurrent one.

Fig. 6 analyses γ_t values, introduced in Eq. (21), for the English-to-Spanish translation problem. As expected, the memory decay of the Luong model is faster than the input attention and naive models. The figures for the rest of the problems can be found in appendix B. This result is congruent with our first experiment using the naive architecture. As the state size grows, the forget activation values become more spread and the similarity between contiguous states is lower because the model is able to completely learn the problem without using its full capacity.

In summary, it can be concluded that the Luong attention architecture tends to change the state by resetting it faster and more frequently than the other two architectures, thus vanishing the state vector from the first time steps.

4.3. Sequence-reverse copy problem with bigrams

The fact that standard attention models are biased towards alignment is explored in the sequence-reverse copy problem with bigrams. We trained a Luong attention model and explored the attention distribution for different input–output pairs. We also computed the state similarity for the state between each two bigram sequence symbols. We set the bigrams to have a random prior probability within the range $[0.8, 0.9]$. Although this is much higher than in real language-based problems, it also stresses the

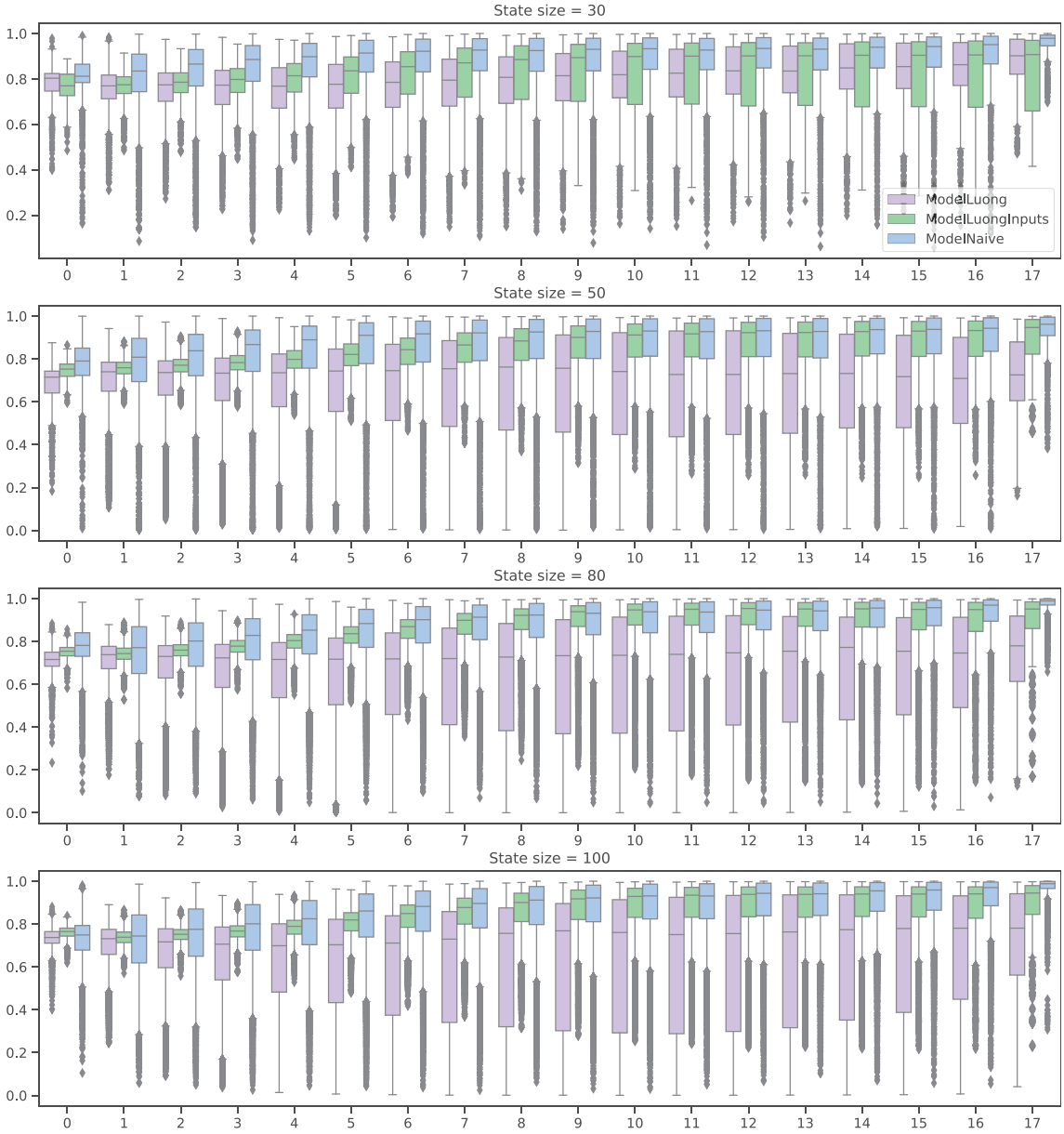

Fig. 11. Sequence-reverse copy problem with bigrams: box-plots representing the distribution of the forget gate activations for three models (naive, Luong attention and Luong input attention) at each time step of the sequence.

assumption exposed in Section 2.2 as it makes it easier to learn these bigrams for the encoder. Fig. 7 shows that for some examples of input–output sequences, despite the fact that the bigrams are highly frequent, the Luong attention model tends to build specific alignments for each of the bigram symbols, thus keeping the state similarity low for these time steps. This observation further supports our results from the last sections.

5. Discussion and conclusions

This paper has presented an empirical analysis about memory degradation in neural recurrent architectures. Our conjecture, supported by the results of our experiments, is that attention may transform RNN architectures from their normal function as memory accumulators by forcing big changes in their memory due to the additional output connection. This analysis reinforces the idea of superiority of pure non-recurrent architectures like Transformers since if the recurrent connection of the RNN networks does not add any advantage, then it becomes redundant thus making attention be the only relevant mechanism. It is also important to note that this conclusion about the redundancy of the recurrent connection may not be applicable if the remaining information in the state is, for example, useful for alignment decoding; therefore, the claims made in the paper refer to a limit situation rather than a common one.

A remarkable finding that indirectly validates our results is the use of the attention alignments as an interpretation of which input is observed in order to obtain the output. This interpretation is commonly used in almost all current literature. It is relevant to note that even if this interpretation usually works and the distribution of the alignments coincides with the expected one (like in translation or sentiment analysis problems), these alignments are obtained from the output of the encoder network and not from the input. This fact indicates that the output and the input of the encoder are highly related to each other, which intuitively should not happen. The output of the encoder must be more related to the kept information than to the newly introduced information, especially in the last time steps where the memory carries almost

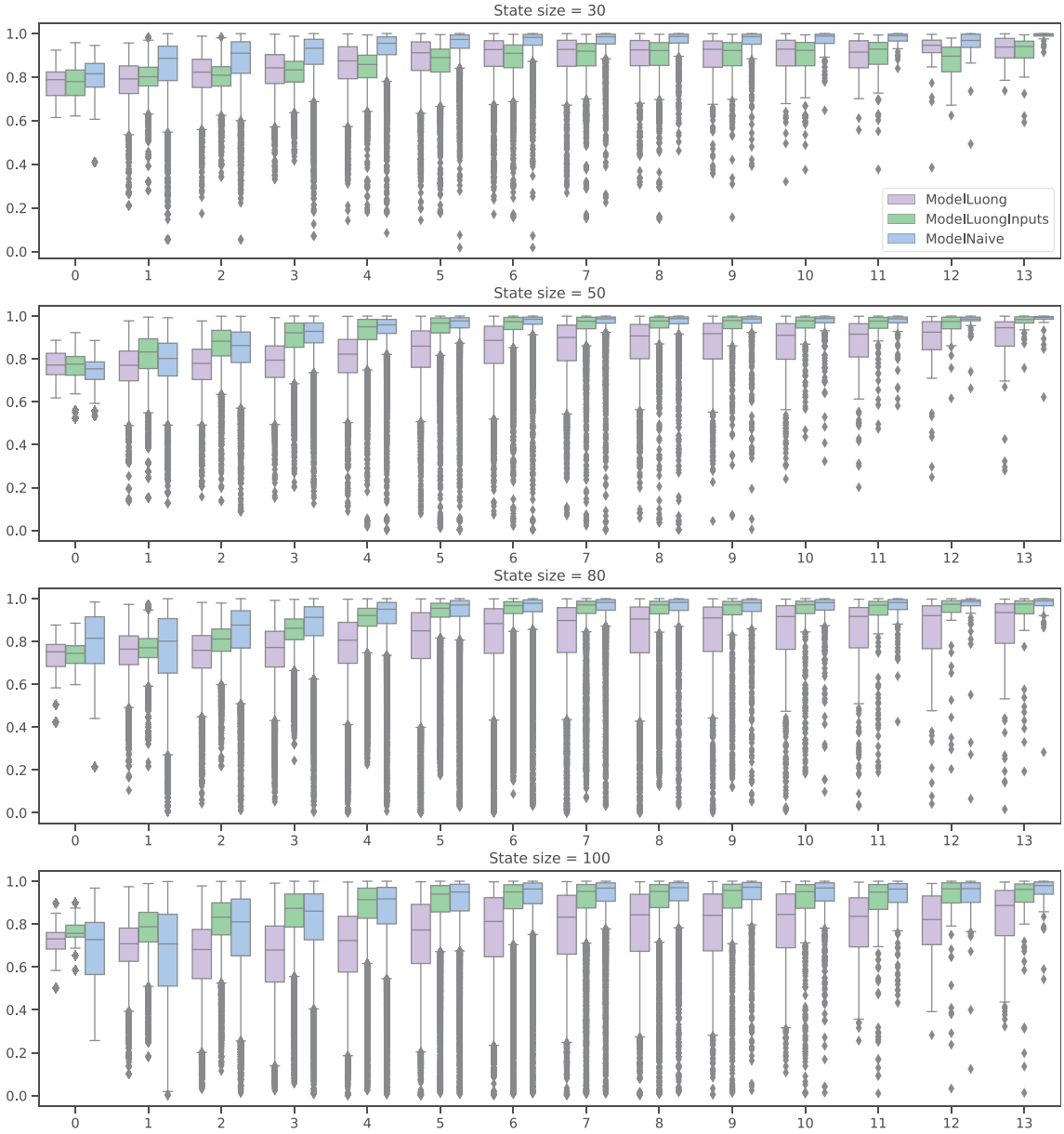

Fig. 12. Translation English-to-French: box-plots representing the distribution of the forget gate activations for three models (naive, Luong attention and Luong input attention) at each time step of the sequence.

all the information of the sequence. The success of this attention interpretation is just another way to suspect that LSTM is not working as it is supposed to do in classical attention architectures, basically becoming a feedforward network with additional operations.

A promising future work can be the study of the real influence of both output branches on the final output of the encoder. There is not a direct way to measure this due to their hard interconnection in this type of architectures. However, studying the norm of the gradients among these branches can be a research experiment for validating the results presented in this paper. We also hypothesise that if we can balance the amount of usage of the recurrent part of the RNN with the direct output part we will be paving the way to build hybrid recurrent-parallel architectures where the recurrent part (memory) will take care of the alignment learning and the parallel part (attention) will be in charge of the transformation learning without influencing on each other.

CRediT authorship contribution statement

Mykola Harvat: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing - original draft, Visualization, Data curation. **José D. Martín-Guerrero:** Conceptualization, Resources, Writing - review & editing, Supervision, Funding acquisition, Project administration.

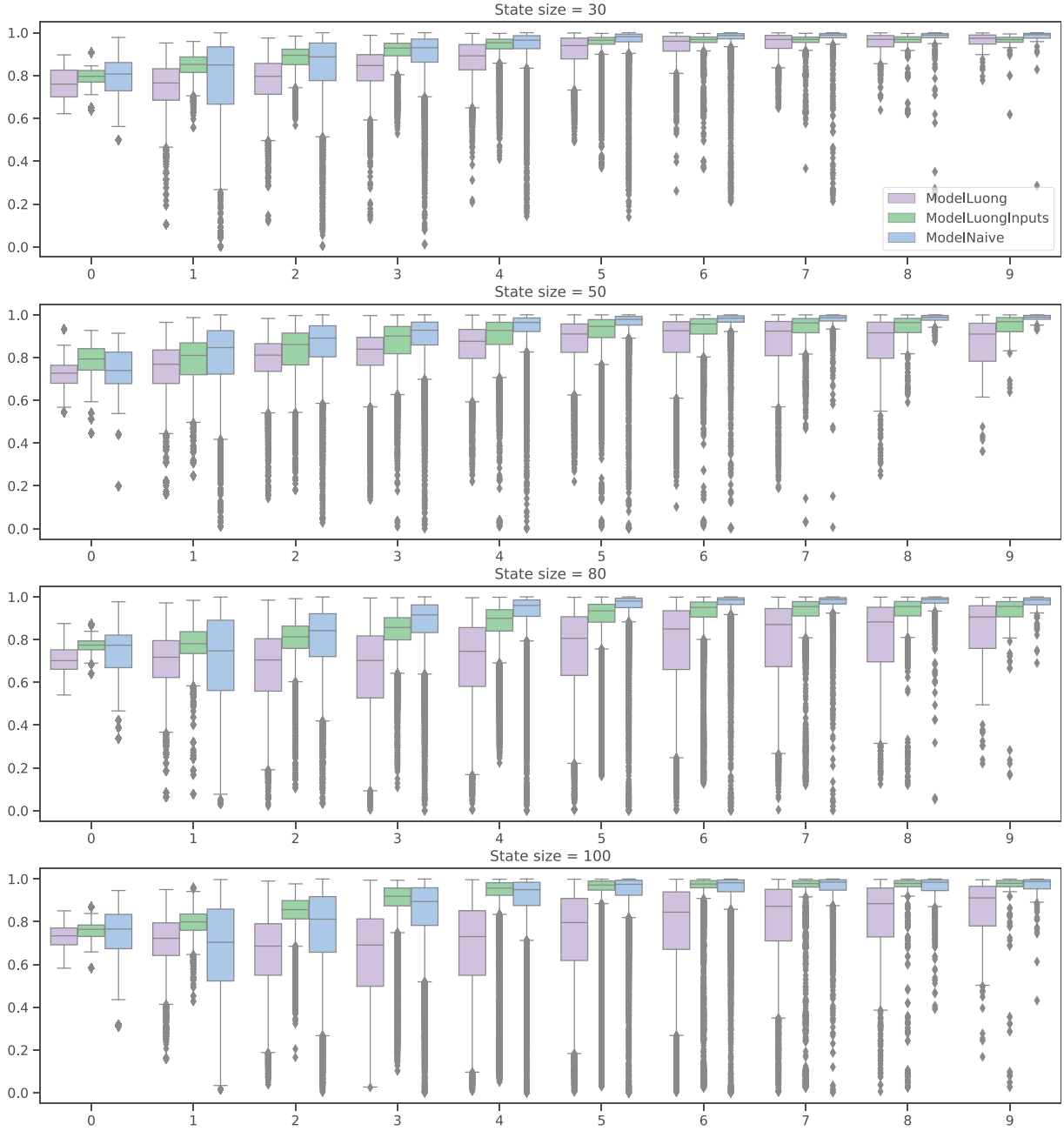

Fig. 13. Translation English-to-German box-plots representing the distribution of the forget gate activations for three models (naive, Luong attention and Luong input attention) at each time step of the sequence.

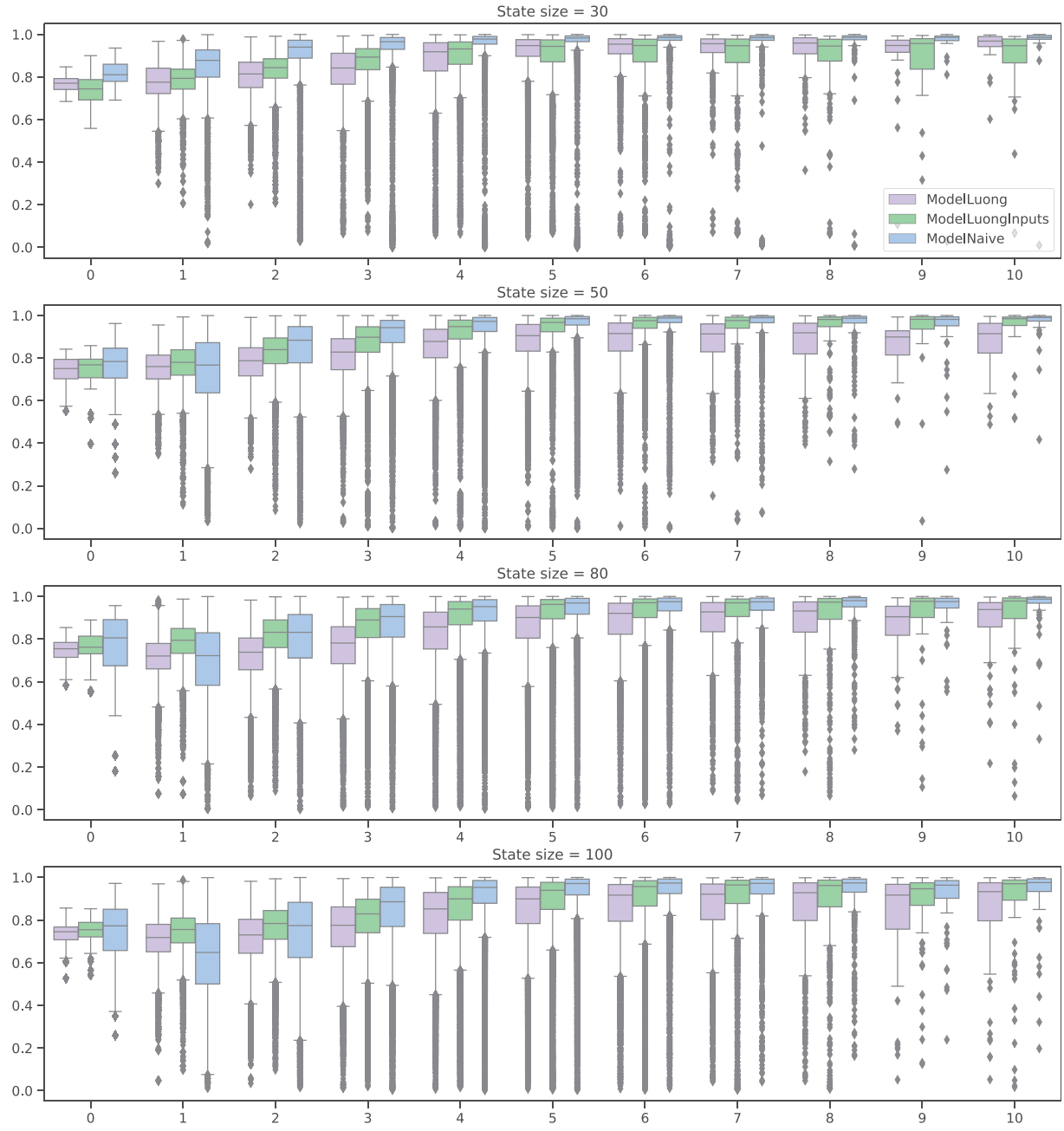

Fig. 14. Translation English-to-Italian: box-plots representing the distribution of the forget gate activations for three models (naive, Luong attention and Luong input attention) at each time step of the sequence.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This research was partially funded by the Valencian Government under its program for doctoral grants (reference number ACIF/2020/043).

Appendix A

In this appendix we will detail how the first four toy datasets in our work were generated:

A.1. Sequence-reverse copy problem

For this problem we generated a vocabulary of 128 random symbols and we built input sequences of random length between 3 and 18. The goal of this task was to transform the input sequence into an output sequence which is the same but with the symbols in a reverse order. The first and the last symbol were a generic

"< *start* >" and "< *end* >" symbol, respectively, which was not reversed. We padded the sequences to be all of the same length.

A.2. Sequence-reverse copy problem with repetitions

This problem is similar to the last one, as we also used a vocabulary of 128 random symbols and generated sequences of 18 symbols which were to be reversed by the model. The difference with the previous problem is that we first produced a sequence of 7 symbols, and then repeated each symbol of that sequence a random number between 1 and 4 times. We ensured that the final length was 18 by either dropping symbols or repeating the last symbol the number of times needed to reach a length of 18.

A.3. Filter-sequence problem

In this problem we generated an additional tabu vocabulary T of 20 symbols, picking random symbols from a vocabulary V of size 128. We generated sequences of length 18 from V ensuring that a random number between 1 and 7 symbols belong to T. The target sequences were defined as the input sequences, but taking into account that any symbol from T was filtered. The sequences were padded to have a final length of 18.

A.4. Sequence-reverse copy problem with bigrams

This problem is similar to the sequence-reverse copy problem; we also worked with random sequences of up to 18 symbols from a vocabulary of 128. The goal was to copy the sequence in the reverse order. The difference is that during the sequence generation we forced 15 random bigrams to have a random frequency between 0.5 and 0.9 while keeping the rest of the bigrams unaltered.

Appendix B

Figs. 8–14 show the activation exploration for the other seven problems, namely, sequence-reverse copy problem, sequence-reverse copy problem with repetitions, filter-sequence problem, sequence-reverse copy problem with bigrams and English to French, English to German and English to Italian translation problems.

References

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., & Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. https://www.tensorflow.org/ software available from tensorflow.org.

[2] Anki (Accessed 2020). Tab-delimited bilingual sentence pairs. http://www.manythings.org/anki/. Accessed: 2020-01-15.

[3] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

[4] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

[5] Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q.V., & Salakhutdinov, R. (2019). Transformer-xl: Attentive language models beyond a fixed-length context. arXiv preprint arXiv:1901.02860.

[6] F.A. Gers, J.A. Schmidhuber, F.A. Cummins, Learning to forget: Continual prediction with lstm, Neural Comput. 12 (2000) 2451–2471, https://doi.org/10.1162/089976600300015015.

[7] A. Graves, J. Schmidhuber, Framewise phoneme classification with bidirectional lstm and other neural network architectures, Neural Networks 18 (2005) 602–610.

[8] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Computation 9 (1997) 1735–1780, https://doi.org/10.1162/neco.1997.9.8.1735, DOI: 10.1162/neco.1997.9.8.1735.

[9] Kingma, D.P., & Ba, J. (2015). Adam: A method for stochastic optimization. In Proceedings of the International Conference on Learning Representations (ICLR 2015).

[10] Luong, T., Pham, H., & Manning, C.D. (2015). Effective approaches to attention-based neural machine translation. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (pp. 1412–1421). Lisbon, Portugal: Association for Computational Linguistics. https://www.aclweb.org/anthology/D15-1166. 10.18653/v1/D15-1166.

[11] A. Martins, R. Astudillo, From softmax to sparsemax: A sparse model of attention and multi-label classification, in: International Conference on Machine Learning, 2016, pp. 1614–1623.

[12] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language models are unsupervised multitask learners, OpenAI blog 1 (2019) 9.

[13] Sutskever, I., Vinyals, O., & Le, Q.V. (2014). Sequence to sequence learning with neural networks. In Advances in neural information processing systems (pp. 3104–3112).

[14] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998–6008).

Prof. **José D.** Martín-Guerrero holds a B. Sc degree in Theoretical Physics (1997), a M. Eng. degree in Electronics (1999) and a Ph.D. degree in Machine Learning (2004). He is currently a Full Professor at the Department of Electronic Engineering, Universitat de Valéncia. His research interests include Machine Learning and Computational Intelligence, with special emphasis in Quantum Machine Learning. His research has been reflected in more than 60 papers published in relevant journals and in the organization of more than 30 special sessions and workshops in relevant conferences of his field of research.

Dr. **Mykola Harvat** holds a B. Sc. degree in Telecommunications Electronic Engineering (2017), a B. Sc. In Telematics Engineering (2017), a M. Sc. In Data Science (2018) and a B. Sc degree in Medicine (2019), all from University of Valencia. At present, he is developing his PhD in Artificial Intelligence at University of Valencia focusing his research on Natural Language Processing, Deep Learning and new Machine Learning paradigms.