Seq2Biseq : Bidirectional Output-wise Recurrent Neural Networks for Sequence Modelling

Marco Dinarelli(1), Loïc Grobol(2,3)

(1) LIG, Bâtiment IMAG - 700 avenue Centrale - Domaine Universitaire de Saint-Martin-d'Hères

(2) Lattice CNRS, 1 rue Maurice Arnoux, 92120 Montrouge, France

(3) ALMAnaCH Inria, 2 rue Simone Iff, 75589 Paris, France

17 avril 2019

Résumé

During the last couple of years, Recurrent Neural Networks (RNN) have reached state-of-the-art performances on most of the sequence modelling problems. In particular, the *sequence to sequence* model and the neural CRF have proved to be very effective in this domain. In this article, we propose a new RNN architecture for sequence labelling, leveraging gated recurrent layers to take arbitrarily long contexts into account, and using two decoders operating forward and backward. We compare several variants of the proposed solution and their performances to the state-of-the-art. Most of our results are better than the state-of-the-art or very close to it and thanks to the use of recent technologies, our architecture can scale on corpora larger than those used in this work.

1 Introduction

Sequence modelling is an important problem in NLP, as many NLP tasks can be modelled as sequence-to-sequence decoding. Among them are POS tagging, chunking, named entity recognition [1], Spoken Language Understanding (SLU) for human-computer interactions [2], and also machine translation [3, 4].

In other cases, NLP tasks can be decomposed, at least in principle, in several subtasks, the first of which is a sequence modelling problem. For instance, syntactic parsing can be performed by applying syntactic analysis to POS-tagged sentences [5] ; coreference chain detection [6, 7, 8] can be decomposed into mention detection and coreferent mention linking ; and structured named entity detection [8, 9, 10], can be done by first detecting simple entity components then combining them to construct complex tree-shaped entities.

Most of these tasks can also be performed by a single model : either as a joint architecture like the joint model for POS tagging and syntactic analysis from [11] or with a fully end-to-end model like the one developed by [12] for coreference detection. In any case, these models still include at some point a sequence modelling module that could be improved by studying successful models for the related sequence labelling tasks.

This is even more true for neural models, since designing a single complex neural architecture for a complex problem may indeed lead to sub-optimal learning. For this reason, it may be more desirable to train a sequence labelling model alone at first and to learn to perform the other steps using the pre-trained parameters of the first step's model, as is done for instance when using pre-trained lexical embeddings in a downstream model [13, 14]. In that case, care must be taken to avoid too unrelated downstream tasks that could lead to *Catastrophic forgetting* [15], though some hierarchical multi-task architectures have proven successful [16].

Finally, [17] has shown that it is possible to model syntactic analysis as a sequence labelling problem by adapting a *Seq2seq* model. As a consequence, we could actually design a unified multi-task learning neural architecture for a large class of NLP problems, by recasting them as sequence decoding tasks.

Recurrent Neural Networks (RNNs) hold state-of-the-art results in many NLP tasks, and in particular in sequence modelling problems [13, 14, 18, 12]. Gated RNNs such as GRU and LSTM are particularly effective for sequence labelling thanks to an architecture that allows them to use long-range information in their internal representations [19, 20, 21].

In this paper we focus our work to searching for more effective neural models for sequence labelling tasks such as POS tagging or Spoken Language Understanding (SLU). Several very effective solutions already exist for these problems, in particular the sequence-to-sequence model [3] (*Seq2seq* henceforth), the *Transformer* model [22], and the whole family of models using a neural CRF layer on top of one or several LSTM or GRU layers [20, 21, 13, 14, 23, 24, 25].

We propose an alternative neural architecture to those mentioned above. This architecture uses GRU recurrent layers as internal memory capable of taking into account arbitrarily long contexts of both input (words and characters), and output (labels). Our architecture is a variant of the *Seq2seq* model where two different decoders are used instead of only one of the original architecture. The first decoder goes backward through the sequence, outputting label predictions, using the hidden states of the encoder and its own previous hidden states and label predictions as input. The second decoder is a more standard forward decoder that uses the hidden states of the encoder, the hidden states and *future* predictions generated by the backward decoder and its own previous hidden states and predictions to output labels. We name this architecture *Seq2biseq*, as it generates output sequences from output-wise bidirectional, global decisions.

Our work is inspired by previous work published in [18, 26, 27, 28], where bidirectional output-wise decisions were taken using a simple recurrent network. A similar idea, called *deliberation network*, has been proposed in [29], where however two forward decoders were used. In this respect we believe that using a backward decoder for the first pass may encode more different, expressive information for the second, forward pass. Our architecture takes global decisions like a *LSTM+CRF* model [13] thanks to the use of the two decoders. These take global context into account on both sides of a given position of the input sequence.

We compare our solution with state-of-the-art models for SLU and POS-tagging in particular the models described in [18, 26] and in [13]. In order to make a direct comparison, we evaluate our models on the same tasks : a French SLU task provided with the MEDIA corpus [30], and the well-known task of POS-tagging of the Wall Street Journal portion of the Penn Treebank [31].

Our results are all reasonably close to the state of the art, and most of them are actually better.

The paper is organized as follows : in the next section we describe the state-of-the-art of neural models for sequence labelling. In the section 3 we describe the neural model we propose in this paper, while in the section 4 we describe the experiments we performed to evaluate our models. We draw our conclusions in the section 5

2 State of the Art

The two main neural architectures used for sequence modelling are the *Seq2seq* model [3] and a group of models where a neural CRF output layer is stacked on top of one or several LSTM or GRU layers [20, 21, 13, 14, 23, 24, 25].

The *Seq2seq* model, also known as *encoder-decoder*, uses a first module to encode the input sequence as a single vector c. In the version of this model proposed in [3] c is the hidden state of the encoder after seeing the whole input sequence. A second module decodes the output sequence using its previous predictions and c as input.

The subsequent work of [4] extends this model with an attention mechanism. This mechanism provides the decoder with a dynamic representation of the input that depends on the decoding step, which proved to be more efficient for translating long sentences.

This mechanism has also been turned out to be effective for other NLP tasks [12, 32, 33].

Concerning models using a neural CRF output layer [14, 13], a first version was already described in [1]. These solutions use one or more recurrent hidden layers to encode input items (words) in context. Earlier simple recurrent layers like *Elman* and *Jordan* [34, 35], which showed limitations for learning long-range dependencies [36], have been replaced by more sophisticated layers like LSTM and GRU [20, 21], which reduced such limitations by using gates.

In this type of neural models, a first representation of the prediction is computed with a local output layer. In order to compute global predictions with a CRF neural layer, the *Viterbi* algorithm is applied over the sequence of local predictions [1, 37].

A more recent neural architecture for sequence modelling is the *Transformer* model [22]. This model use an innovative deep non-recurrent neural architecture, relying heavily on attention mechanisms [4] and skip connections [38] to

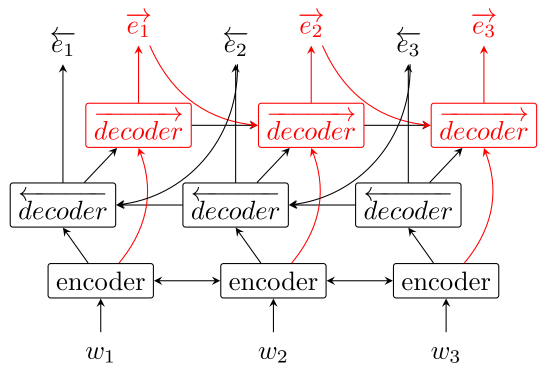

FIGURE 1 – Overall network structure

overcome limitations of recurrent networks in propagating the learning signal over long paths. The Transformer model has been designed for computational efficiency reasons, but it captures long-range contexts with multiple attention mechanisms (multi-head attention) applied to the whole input sequence. Skip-connections guarantee that the learning signal is back-propagated effectively to all the network layers.

Concerning previous works on the same tasks used in this work, namely MEDIA [30] and the Penn Treebank (WSJ) [31], several publications have been produced starting from 2007 (MEDIA) and 2002 (WSJ) [39, 40, 41, 42, 43, 44], applying several different models like *SVM* and *CRF* [45, 46]. Starting from 2013 several works also focused on neural models. At first simple recurrent networks have been used [47, 48, 49]. In the last few years also more sophisticated models have been studied [50, 23, 18].

3 The Seq2biseq Neural Architecture

As an alternative to the *Seq2seq* and *LSTM+CRF* neural models for sequence labelling, we propose in this paper a new neural architecture inspired from the original *Seq2seq* model and from models described in [18, 26]. Figure 1 shows the overall architecture.

Our architecture is similar to the *Seq2seq* model in that we use modules to encode a long-range context on the output side similar to the decoder of the *Seq2seq* architecture. The similarity with respect to models described in [18, 26] is the use of a bidirectional context on the output side in order to take into account previous, but also future predictions for the current model decision. Future predictions are computed by an independent decoder which processes the input sequence backward.

Our architecture extends the *Seq2seq* original model through the use of an additional backward decoder that allows taking into account both past and future information at decoding time. Our architecture also improves the models described in [18, 26] since it uses more sophisticated layers to model long-range contexts on the output side, while previous models used fixed-size windows and simple linear hidden layers. Thanks to these modifications our model makes predictions informed by a global distributional context, which approximates a global decision function. We also improve the character-level word representations by using a similar solution to the one proposed in [14].

Our neural architecture is based on the use of GRU recurrent layers at word, character and label levels. GRU is an evolution of the LSTM recurrent layer which has often shown better capacities to model contextual information [21, 23].

In order to make notation clear, in the following sections, bidirectional GRU hidden layers are noted GRU, while we use $\overrightarrow{GRU}$ and $\overleftarrow{GRU}$ for a forward and backward hidden layer respectively. For the output of these layers we use respectively h_{w_i}, $\overrightarrow{h_{e_i}}$ and $\overleftarrow{h_{e_i}}$, with a letter as index to specialize the GRU layer for a specific input (e.g. w for the GRU layer used for words, e for labels, or entities, and so on), and an index i to indicate the index position in the current sequence. For example $\overleftarrow{h_{e_i}}$ is the backward hidden state, at current position (i), of the GRU layer for labels. The models described in this work always use as input words, characters and labels. Their respective embedding matrices are all noted E_x, with x denoting the different input unit types (e.g. E_w is the embedding matrix for words), and their dimensions D_x.

3.1 Character-level Representations

The character-level representation of words was computed at first as in [14], substituting a GRU to the LSTM layer : the characters $c_{w,1}, \ldots, c_{w,n}$ of a word w are first represented as a sequence $S_c(w)$ of n D_c-dimensional embeddings. These are fed to the GRU_c layer. The final state $h_c(w)$ is kept as the character level representation of w.

We improved this module so that it generates a character-level representation using all the hidden states generated by GRU_c :

$$\begin{aligned} S_c(w) &= (E_c(c_{w,1}), \ldots, E_c(c_{w,n})) \\ (h_c(c_{w,1}), \ldots, h_c(c_{w,n})) &= \text{GRU}_c(S_c(w), h_0^c) \\ h_c(w) &= \text{FFNN}(Sum(h_c(c_{w,1}), \ldots, h_c(c_{w,n}))) \end{aligned} \tag{1}$$

FFNN is again a general, possibly multi-layer Feed-Forward Neural Network with non-linear activation functions. This new architecture was inspired by [22], where FFNNs were used to extract deeper features at each layer.

Preliminary experiments have shown that this character-level representation is more effective than the one inspired by the work of [14].

3.2 Word-level Representations

Words are first mapped into embeddings, then the embedding sequence is processed by a GRU_w bidirectional layer. Using the same notation as for characters, a sequence of words $S = w_1, \ldots, w_N$ is converted into embeddings $E_w(w_i)$ with $1 \leq i \leq N$. We denote $S_i = w_1, \ldots, w_i$ the sub-sequence of S up to the words w_i. In order to augment the word representations with their character-level representations, and to use a single distributed representation, we concatenate the character-level representations $h_c(w_i)$ (eq. 1) to the word embeddings before feeding the GRU_w layer with the whole sequence. Formally :

$$\begin{aligned} S_w &= (E_w(w_1), \ldots, E_w(w_N)) \\ S^{lex} &= ([E_w(w_1), h_c(w_1)], \ldots, [E_w(w_N), h_c(w_N)]) \\ h_{w_i} &= \text{GRU}_w(S_i^{lex}, h_{w_{i-1}}) \end{aligned} \tag{2}$$

Where we used S_w for the whole sequence of word embeddings generated from the word sequence S.

In the same way, S^{lex} is the sequence obtained concatenating word embeddings and character-level representations, which constitute the lexical-level information given as input to the model. [ ] is the matrix (or vector) concatenation, and we also used the notation S_i^{lex} for the sub-sequence of S^{lex} up to position i.

3.3 Label-level Representations

In order to obtain label representations encoding long-range contexts, we use a GRU hidden layer also on label embeddings. We apply first a backward step on label embeddings in order to compute representations that will be used as future label predictions, or right context, in the following forward step. Using the same notation as used previously, we have :

$$\overleftarrow{h_{e_i}} = \overleftarrow{GRU}_e(E_l(e_{i+1}), \overleftarrow{h_{e_{i+1}}}) \tag{3}$$

for $i = N \ldots 1$. We note that here we use the label on the right of the current position, e_{i+1}, e_i is not known at time step i.

The hidden state $\overleftarrow{h_{e_{i+1}}}$ is the hidden state computed at previous position in the backward step, thus associated to the label on the right of the current label to be predicted. In other words we interpret $\overleftarrow{h_{e_i}}$ as the right context of the (unknown) label e_i, instead of as the in-context representation of e_i itself, and similarly for $\overleftarrow{h_{e_{i+1}}}$. The right context of e_i, $\overleftarrow{h_{e_i}}$, is used to predict e_i at time step i.

In the same way, we compute the representation of the left context of the label e_i by scanning the input sequence forward, which gives :

$$\overrightarrow{h_{e_i}} = \overrightarrow{GRU}_e(E_l(e_{i-1}), \overrightarrow{h_{e_{i-1}}}) \tag{4}$$

for $i = 1 \ldots N$. The neural components described so far are already sufficient to build rich architectures. However, we believe that the information from the lexical context is useful not only to disambiguate the current word in-context, but also to disambiguate the contextual representations used for label prediction.

Indeed, in sequence labelling labels only provide abstract lexical or semantic information. It thus seems reasonable to think that they are not sufficient to effectively encode features in the label context representations $\overleftarrow{h_{e_i}}$ and $\overrightarrow{h_{e_i}}$.

For this reason, we add to the input of the layers $\overleftarrow{GRU}_e$ and $\overrightarrow{GRU}_e$ the lexical hidden representation h_{w_i} computed by the GRU_w layer. Taking this into account, the computation of the right context for the current label prediction becomes :

$$\overleftarrow{h_{e_i}} = \overleftarrow{GRU}_e([h_{w_i}, E_l(e_{i+1})], \overleftarrow{h_{e_{i+1}}}) \tag{5}$$

The computation of the left context is done in a similar way.

This modification makes the layers $\overleftarrow{GRU}_e$ and $\overrightarrow{GRU}_e$ in our architecture similar to the decoder of a *Seq2seq* architecture [3]. The modules $\overleftarrow{GRU}_e$ and $\overrightarrow{GRU}_e$ are indeed like two decoders from an architectural point of view, but also they encode the contextual information in the same way using gated recurrent layers.

However, the full architecture differs from a traditional *Seq2seq* model by the use of an additional decoder, capable of modelling the right label context, while the original model used a single decoder, modelling only the left context. The idea of using two decoders is inspired mainly by the evidence that both left and right output-side contexts are equally informative for the current prediction.

Another difference with respect to the *Seq2seq* model is that the $\overleftarrow{GRU}_e$ and $\overrightarrow{GRU}_e$ layers have access to the lexical-level hidden states h_{w_i}. This allows these layers to take the current lexical context into account and is thus more adapted to sequence labelling than using the same representation of the input sentence for all the positions, which is the solution of the original *Seq2seq* model.

As we mentioned above, the *Seq2seq* model has been improved with an attention mechanism [4], which is another way to provide the model with a lexical representation focusing dynamically on different parts of the input sequence depending on the position i. This attention mechanism has also proved to be efficient for sequence labelling, and it might be that our architecture could benefit from it too, but this is out of our scope for this article and we leave it for future work. [1]

We can motivate the use of the lexical information h_{w_i} in the decoders $\overleftarrow{GRU}_e$ and $\overrightarrow{GRU}_e$ with complex systems theory considerations, as suggested in [51]. [52] state that a complex system, either biological or artificial, is not equal to the sum of its components. More precisely, the behaviour of a complex system evolves during its existence and shows the emergence of new functionalities, which can not be explained by simply considering the system's components individually. [53] qualitatively characterizes the evolution of a complex system's behaviour with three different types of adaptation, two of which are particularly interesting in the context of this work and can be concisely named *aggregation* and *specialization*.

In the first, several components of the system adapt in order to become a single *aggregated* component from a functioning point of view. In *specialization*,

1. This is currently in progress

several initially identical components of the system adapt to perform different functionalities. These adaptations may take place at different unit levels, a neuron, a simple layer, or a whole module.

The most evident cases of *specialization* are the gates of the LSTM or GRU layers [21], as well as the attention mechanism [4]. Indeed, the **z** and **r** gates of a GRU recurrent layer are defined in the exact same way, with the same number of parameters, and they use exactly the same input information.

However, during the evolution of the system — that is, during the learning phase — the **r** gate adapts (specialises) to become the reset gate, which allows the network to forget the past information, when it is not relevant for the current prediction step. In the same way, the **z** gate becomes the equivalent of the input gate of a LSTM, which controls the amount of input information that will affect the current prediction.

In our neural architecture we can observe *aggregation* : the layers $\overleftarrow{GRU}_e$ and $\overrightarrow{GRU}_e$ adapt at the whole layer level, they become like gates which filter label-level information that is not useful for the current prediction. In the same way as the input to gates of GRU or LSTM is made of current input and previous hidden state, the input to the $\overleftarrow{GRU}_e$ and $\overrightarrow{GRU}_e$ layers is made of lexical level and previous label level information, both needed to discriminate the abstract semantic information provided by the labels alone. We will show in the evaluation section the effectiveness provided by this choice.

While both of the two decoders used in our models are equivalent to the decoder of the original *Seq2seq* architecture, we believe it is interesting to analyse the contribution of each piece of information given as input to this component, which we will show in the evaluation section.

3.4 Output Layer

Once all pieces of information needed to predict the current label are computed, the output of the backward step is computed as follows :

$$\begin{aligned} o_{bw} &= W_{bw}[h_{w_i}, \overleftarrow{h_{e_i}}] + b_{bw} \\ e_i &= \text{argmax}(\text{log-softmax}(o_{bw})) \end{aligned} \tag{6}$$

We start the backward step using a conventional symbol (`<EOS>`) as end-of-sentence marker. We repeat the backward step prediction for the whole input sequence. The process is shown in figure 2.

This allows to have all the pieces of information needed to predict the current label in the forward step, at character and word level, but also at right and left label context level, with respect to the current position to be labeled :

$$\begin{aligned} o_i &= W_o[\overrightarrow{h_{e_i}}, h_{w_i}, \overleftarrow{h_{e_i}}] + b_o \\ e_i &= \text{argmax}(\text{log-softmax}(o_i)) \end{aligned} \tag{7}$$

A high-level schema of the forward pass is shown in figure 3.

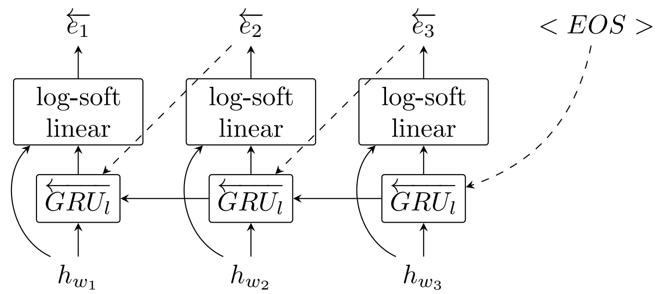

FIGURE 2 – Structure of the backward decoder

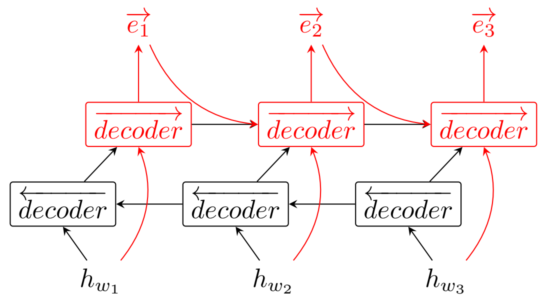

FIGURE 3 – High-level schema of the forward pass

The log-softmax function computes log-probabilities and it is thus suited for the loss-function used to learn the model described in the next section.

We note that the forward decoder is in fact a bidirectional decoder, as it uses both backward and forward hidden states $\overrightarrow{h_{e_i}}$ and $\overleftarrow{h_{e_i}}$ for the current prediction.

The hypothesis motivating the architecture of our neural models is the following : gated hidden layers such as LSTM and GRU can keep relatively long contexts in memory and to extract from them the information that is relevant to the current model prediction. This is supported by the findings in recent works, such as [54], which shows that most of the modelling power of gated RNN comes from their ability to compute at each step a context-dependent weighted sum on their inputs, in a way that is akin to dynamical attention mechanism. As an immediate consequence, we think that using such hidden layers is an effective way to keep in memory a relatively long context on the output item level, that is labels, as well as on the input item level, that is words, characters and possibly other information.

An alternative, non-recurrent architecture, the Transformer model [22] has been proposed with the goal of using attention mechanisms to overcome the learning issues of RNN in contexts where the learning signal has to back-propagate through very long paths. However, the recent work of [55] shows that integrating a concept of recurrence in Transformers can improve their performances in some contexts. This leads us to believe that recurrence is a fundamental feature for neural architectures for NLP and all of the domains where data are sequential by nature.

As a side note, the main features of the Transformer model - the multi-head attention mechanism and the skip connections [22] - could in principle be integrated into our architecture. Investigations of the costs and benefits of such additions is left for future work.

Finally, while the decision function of our model remains local, its decisions are informed by global information at the word, character and label level thanks to the use of long-range contexts encoded by the GRU layers. In that sense, it can be interpreted as an approximation of a global decision function and provides a viable alternative to the use of a CRF output layer [13, 14].

3.5 Learning

Our models are learned by minimizing the negative log-likelihood $\mathcal{LL}$ with respect to the data. Formally :

$$- \mathcal{LL}(\Theta|D) = - \sum_{d=1}^{|D|} \sum_{i=1}^{N_d} \frac{1}{2}(\text{log-p}(\overrightarrow{e_i}) + \text{log-p}(\overleftarrow{e_i})) + \frac{\lambda}{2} |\Theta|^2) \qquad (8)$$

log-p($\overrightarrow{e_i}$) and log-p($\overleftarrow{e_i}$) are the log-probabilities over predictions of the forward and backward decoders, respectively, we thus strengthen the global character of our model's predictions. The first sum scans the learning data D of size $|D|$, while the second sum scans each learning sequence S_d, of size N_d.

Given the relatively small size of the data we use for the evaluation, and the relatively high complexity of the models proposed in this paper, we add a L_2 regularization term to the cost function with a λ coefficient. The cost-function is minimized with the *Back-propagation Through Time* algorithm (BPTT) [19], provided natively by the *Pytorch* library (see section 4.2).

4 Evaluation

4.1 Data

We evaluate our models on two tasks, one of Spoken Language Understanding (SLU), and one of POS tagging, namely *MEDIA* and *WSJ* respectively. These tasks have been widely used in the literature [49, 23, 18, 14, 56] and allow thus for a direct comparison of results.

The French MEDIA corpus [30] was created for the evaluation of spoken dialog systems in the domain of hotel information and reservation in France. It is made of 1 250 human-machine dialogs acquired with a *Wizard-of-OZ* approach, where 250 users followed 5 different reservation scenarios.

Data have been manually transcribed and annotated with domain concepts, following a rich ontology. Semantic components can be combined to build rela-

MEDIA corpus example		
Words	**Classes**	**Labels**
Oui	-	Answer-B
l'	-	BDObject-B
hotel	-	BDObject-I
le	-	Object-B
prix	-	Object-I
à	-	Comp.-payment-B
moins	relative	Comp.-payment-I
cinquante	tens	Paym.-amount-B
cinq	units	Paym.-amount-I
euros	currency	Paym.-currency-B

Table 1 – An example of sentence with its semantic annotation and word classes, taken from the French corpus MEDIA. The translation of the sentence in English is "Yes, the hotel with a price less than fifty euros per night"

	Training		Validation		Test	
# sentences	12 908		1 259		3 005	
	Words	Concepts	Words	Concepts	Words	Concepts
# words	94 466	43 078	10 849	4 705	25 606	11 383
# dict.	2 210	99	838	66	1 276	78
# OOV%	–	–	1,33	0,02	1,39	0,04

Table 2 – Statistics on the French MEDIA corpus

tively complex semantic classes. [2]

Statistics on the training, development and test data of the MEDIA corpus are shown in table 2. The MEDIA task can be modelled as a sequence labelling task by segmenting concepts over words with the BIO formalism [57]. An example of sentence with its semantic annotation is shown in table 1. For exhaustive, we also show some word-classes available for this task, allowing models for a better generalization. However, our model does not use these classes, as explained in section 4.2.

The English corpus Penn Treebank [31], and in particular the section of the corpus corresponding to the articles of Wall Street Journal (WSJ), is one of the most known and used corpus for the evaluation of models for sequence labelling.

The task consists of annotating each word with its Part-of-Speech (POS) tag. We use the most common split of this corpus, where sections from 0 to 18 are used for training (38 219 sentences, 912 344 tokens), sections from 19 to 21 are used for validation (5 527 sentences, 131 768 tokens), and sections from 22 to 24 are used for testing (5 462 sentences, 129 654 tokens).

4.2 Experimental settings

In order to keep our architecture as general as possible, we limit our model inputs to the strict word (and character) information available in the raw text

2. For example, the label `localisation` can be combined with the components `ville` (city), `distance-relative` (relative-distance), `localisation-relative-générale` (general-relative-localisation), `rue` (street), etc.

data and ignore the additional features available in the MEDIA dataset.

For convenience, the hyperparameters of our system have been tuned by simple independent linear searches on the validation data — rather than a grid search on the full hyperparameters space.

All of the parameters of neural layers are initialised with the Pytorch 0.4.1 default initializers [3] and trained by SGB with a 0.9 momentum for 40 epochs on MEDIA, and ADAM optimizer for 52 epochs on WSJ, keeping the model that gave the best accuracy on the development data set.

For training, we start with a learning rate of 0.125 that we decay linearly after each epoch to end up at 0 at the end of the chosen number of training epochs. Following [59], we also apply a random dropout to the embeddings and the output of the hidden layers that we optimized to a rate of 0.5, and L_2 regularization to all the parameters with an optimal coefficient of 10^{-4}.

Finally, we have conducted experiments to find the optimal layer sizes, which gave us 200, 150 and 30 for word, labels and character embeddings respectively, 100 for the GRU_c layer and 300 for all the other GRU layers. Those values are for the MEDIA task ; for WSJ only the word embeddings and hidden layer sizes (respectively 300 and 150) are different.

In order to reduce the training time, we use mini-batches of size [4] 100. In the current neural network frameworks, all the sequences in a mini-batch must have the same length, which we enforced at first by padding all of the sentences with the conventional symbol `<s>` to the length of the longest one. However this caused two problems : first, there are a few unusually long sentences in the datasets we used, for instance, there is a single sentence of 198 words in MEDIA. Secondly, in order to compute automatically the gradients of the parameters, Pytorch keeps in memory the whole graph of operations performed on the input of the model [60], which was far too large for the hardware we used, since for our model, we have to keep track of all the operations at all of the timesteps.

We found two solutions to these problems. The first was to train on fixed-length, overlapping sub-sequences, or segments [5], truncated from the whole sentences, which did not appear to impair the performances significantly and allowed us to avoid more involved solutions such as back-propagation through time with memorization [61]. The second was to cluster sentences by their length. This makes small clusters for unusually long sentences, which fit thus in memory, and big clusters of average-length sentences, which are further split into sub-clusters to have an optimal balance between the learning signals of different clusters, and alleviate us to find adaptive learning rates for different clusters.

In the optimization phase, we found out that the first solution works far better for the MEDIA task. We believe that this is due to the noisy nature of the corpus (speech transcription), and to its relatively small size Using fixed-length segments reduces the amount of noise the network must filter, while the fact that segments shift and overlap makes the network more robust, as it can

3. Uniform random initialization for the GRU layers and [58] initialization for the linear layers.
4. Using larger batches is faster but degrades the overall accuracy.
5. Shifting each segment one token ahead with respect to the previous

see any token as the beginning of a segment, which in turns helps overcoming scarcity of the dataset. This robustness is not needed when using bigger amount of grammatically well-formed textual data, like the WSJ corpus. Indeed the two solutions gave similar results on this corpus, we thus preferred sentence clusters which is a more intuitive solution and may better fit bigger data sets.

After performing these optimization on the development set for each task, we kept the best models and evaluated them on the corresponding test sets, which we report and discuss in the next section.

All of our development and experiments were done on *2,1 GHz Intel Xeon E5-2620* CPUs and *GeForce GTX 1080* GPUs.[6].

4.3 Results

Results presented in this section on the MEDIA corpus are means over 10 runs, while results on the WSJ corpus are obtained in a single run, as it seems the most common practice.[7]

Concerning the MEDIA task, since the model selection during the training phase is done based on the accuracy on the development data, we show accuracy in addition to F1 measure and Concept Error Rate (CER) as it is common practice in the literature on this task. F1 measure is computed with the script made available to the community for the *CoNLL* evaluation campaign.[8]. CER is computed by Levenshtein alignment between reference annotation and model hypothesis, with an algorithm much similar to the one implemented in the *sclite* toolkit.[9]

Since our model is similar to *Seq2seq* model, but it uses two decoders, in the remainder of this paper our model will be named *Seq2Biseq*. The model training is performed using gold labels in the training data, while in test phase the model uses predicted labels to build left and right label-level contexts. This corresponds to the best strategy, according to [47].

We compare our results to those obtained by running the software developed for [18][10] and tuning its hyperparameters[11].

Concerning our hypothesis about the capability of our models to encode a long-range context, and to filter out useless information with respect to the current labelling decision, we show results of two (sets of) experiments to validate such hypothesis.

In the first one, we compare the results obtained by models with and without the use of the lexical information as input to the decoders $\overleftarrow{GRU}_e$ and $\overleftarrow{GRU}_e$ (section 3.3). These results are shown in the first two lines of the table 3. The model using the lexical information is indicated with Seq2Biseq$_{le}$ in the table (for

6. 1600 MHz, 2560 cores
7. We can note that results over different runs on the WSJ have a very small variation, less or equal to 0.01 accuracy points
8. `https://github.com/robertostling/efselab/blob/master/3rdparty/conlleval.perl`
9. `http://www1.icsi.berkeley.edu/Speech/docs/sctk-1.2/sclite.htm`
10. Available upon request at `http://www.marcodinarelli.it/software.php`
11. The optimal settings being more or less those provided in the original article

Model	**Accuracy**	**F1 measure**	**CER**
MEDIA DEV			
Seq2Biseq	89.11	85.59	11.46
Seq2Biseq$_{le}$	89.42	86.09	10.58
Seq2Biseq$_{le}$ seg-len 15	**89.97**	**86.57**	**10.42**
fw-Seq2Biseq$_{le}$ seg-len 15	89.51	85.94	11.40

TABLE 3 – Comparison of results on the development data of the MEDIA corpus, with and without the lexical information (Seq2Biseq$_{le}$) as input to the modules $\overleftarrow{GRU}_e$ and $\overleftarrow{GRU}_e$

labels and l**e**xical information). As we can see in the table, this model obtains much better results than the one not using the lexical information as input to the label decoders. This confirms that this information helps discriminating the semantic information provided by labels at a given processing step of the input sequence.

In the second experiment, we test the capability of our models to filter out useless semantic information, that is on the label side, for the current labelling decision. In order to do this, we increase the size of the segments in the learning phase : 15 instead of 10 by default. It is important to note that in the context of a SLU task, where input sequences are transcriptions of human speech, using longer segments is possibly risky, since a longer context may be much more noisy even if it is slightly more informative.

Moreover, the models in the literature applied to the MEDIA task and using a fixed-size window to capture contextual information, never use a window wider than 3 tokens around the current token to be labelled. This confirms the difficulty to extract useful information from a longer context. Results of this experiment are shown in the third line of table 3. Our hypothesis seems to be also valid in this case, as models using segments of length 15 obtain better results than those using the default size of 10 and this with respect to all the evaluation metrics.

We note that, while the effectiveness of the decoder's architecture of the *Seq2seq* model does not need any more to be proved, these results still provide possibly interesting analyses in the particular context of sequence labelling.[12]

In order to show the advantage provided by the use of two decoders instead of only one like in the original *Seq2seq* model, we show results obtained using only one decoder for the left label-side context in table 3 These results are indicated in the table with *fw-Seq2Biseq*$_{le}$ *seg-len 15* (this model corresponds basically to the original *Seq2seq*). This model is exactly equivalent to our best model *Seq2Biseq*$_{le}$ *seg-len 15*, the only difference is that it uses only the left label context. As we can see, this model is much less effective than the version using two decoders, which also confirms that the right context on the output side (labels) is very informative.

12. The *Seq2seq* model has been designed and mainly used for machine translation

Model	Accuracy	F1 measure	CER	p-value
MEDIA DEV				
LD-RNN$_{\text{deep}}$	89.26 (0.16)	85.79 (0.24)	10.72 (0.14)	–
Seq2Biseq$_{le}$ seg-len 15	89.97 (0.20)	86.57 (0.22)	10.42 (0.26)	0.043
Seq2Biseq$_{\text{2-opt}}$	**90.22** (0.14)	**86.88** (0.16)	**9.97** (0.24)	–
MEDIA TEST				
LD-RNN$_{\text{deep}}$	89.51 (0.21)	87.31 (0.19)	10.02 (0.17)	–
Seq2Biseq$_{le}$ seg-len 15	89.57 (0.12)	87.50 (0.17)	10.26 (0.19)	0.047
Seq2Biseq$_{\text{2-opt}}$	**89.79** (0.22)	**87.69** (0.20)	**9.93** (0.28)	–

Table 4 – Comparison of results obtained on the MEDIA corpus by the system LD-RNN$_{\text{deep}}$, ran by ourselves for this work, and our model Seq2Biseq$_{le}$, using segments of size 15 (see section 4.2).

Our hypothesis concerning the *aggregation* specialization of our model during the learning phase seems also confirmed (section 3.3). The fact that the Seq2Biseq$_{le}$ model obtains better results than the simpler model Seq2Biseq tends to confirm the hypothesis.

Indeed, if the model Seq2Biseq$_{le}$ gave more importance to the lexical information than the semantic information given by labels at the input of the decoders $\overleftarrow{GRU}_e$ and $\overrightarrow{GRU}_e$, its better results would not have a clear explanation, as both Seq2Biseq$_{le}$ and Seq2Biseq models (table 3) use the lexical information separately (indicated with h_{w_i} in the equation 2).

Since the information provided by labels alone is already taken into account by the model Seq2Biseq, we can deduct that the Seq2Biseq$_{le}$ model can extract more effective semantic representations, and this even when we provide it with longer contexts (with segments of size 15).

In another set of experiments, we compared our model with the one proposed in [18], from which we inspired our neural architecture. We downloaded the software associated to the paper [13], and we ran experiments on the MEDIA corpus in the same conditions as our experiments. We used the deep variant of the model described in [18], LD-RNN$_{\text{deep}}$, which gives the best results on MEDIA. The results of these experiments are shown in the table 4. As we can see in the table, on the development data of the MEDIA task (MEDIA DEV), our model is more effective than the LD-RNN$_{\text{deep}}$ of [18], which holds the state-of-the-art on this task. These gains are also present for the test data (MEDIA TEST), even if they are smaller, and the LD-RNN$_{\text{deep}}$ model is still the more effective in terms of Concept Error Rate (CER).

We would like to underline that we did not perform an exhaustive optimization of all the hyper-parameters. [14] As we can see in table 4, results obtained

13. Described at `http://www.marcodinarelli.it/software.php` and available upon request
14. This because it takes a lot of time, but more importantly because we believe a good model should give good results without too much effort, otherwise a previous model which already proved comparably effective should be preferred

Model	Accuracy	F1 measure	CER
MEDIA TEST			
BiGRU+CRF [18]	–	86.69	10.13
LD-RNN$_{\text{deep}}$ [18]	–	87.36	**9.8**
LD-RNN$_{\text{deep}}$	89.51	87.31	10.02
Seq2Biseq$_{le}$ seg-len 15	89.57	87.50	10.26
Seq2Biseq$_{\text{2-opt}}$	**89.79**	**87.69**	9.93

Table 5 – Comparison of results on MEDIA with our best models and the best models in the literature

with the model LD-RNN$_{\text{deep}}$ on the test data are always better than those obtained on the development data. In contrast, our model obtains a worse accuracy, which leads the model selection in the training phase, on the test data. This lack of generalization may indicate a sub-optimal parameter choice or an over-training problem.

In the table 4 we also report standard deviations on the 10 experiments (between parentheses), and the results of the significance tests performed on the output of our model and of the model LD-RNN$_{\text{deep}}$. We used the significance test described in [62], which applies on the output of the two compared systems, and it is suited for the evaluation metrics used most often in NLP. [15] We re-implemented the significance test script based on the one described in [63]. [16] Our model is compared to the LD-RNN$_{\text{deep}}$ model in terms of F1 measure, which is more constraining than the accuracy and as constraining as the CER. The result of the significance test is given in the column *p-value* of the table, and it represents the probability that the gain is not significant. Most often the gains are considered significant with a p-value equal or smaller than 0.05.

We ran another set of experiments on the MEDIA task with our best model in order to compare to the best models in the literature on this task, which are those described in [18]. In particular we compared our results to the models using a neural CRF output layer for modelling label sequences and take global decisions.

The results of these experiments are shown in the table 5. In this table we indicate simply with LD-RNN$_{\text{deep}}$ the results obtained in our experiments using the software *LD-RNN*[17], while we add the reference [18] after LD-RNN$_{\text{deep}}$ to indicate that results have been taken directly from the reference. As we can see, the only new outcome in this table with respect to those already shown in previous tables, is the best CER of 9.8 obtained by the model LD-RNN$_{\text{deep}}$ published in [18]. These results are obtained however using also the word-classes available with the MEDIA corpus. Our model is still more effective than the others in terms of accuracy and F1 measure, providing thus the new state-of-

15. In contrast to several other significance tests, this test doesn't make any assumption on the classes independence, nor on the representative coverage of the sample
16. `https://nlpado.de/~sebastian/software/sigf.shtml`
17. `http://www.marcodinarelli.it/software.php`

Model	Accuracy	
	WSJ DEV	WSJ TEST
LD-RNN$_{\text{deep}}$	96.90	96.91
LSTM+CRF [14]	–	97.13
Seq2Biseq	97.13	97.20
Seq2Biseq$_{\text{2-opt}}$	**97.33**	**97.35**
LSTM+CRF + Glove [14]	97.46	97.55
LSTM+LD-RNN + Glove [56]	–	97.59

TABLE 6 – Comparison of our model with the model *LD-RNN*$_{\text{deep}}$, and the best models of the literature, on the POS tagging task of the WSJ corpus

the-art results on this task.

The experiments performed on the MEDIA task with different variants of our model allowed us to find the best neural architecture for sequence modelling. In order to have a more general view on the effectiveness of our model on the problem of sequence labelling, we performed some experiments of POS tagging on the WSJ corpus, which is a well-known benchmark for sequence labelling, used since more than 15 years. In order to show the effectiveness of the model alone, without the impact of any external resources, we performed experiments without using pre-trained embeddings. This is however a quite common practice and can lead to remarkable improvements [14].

On this task we compare to the model *LD-RNN*$_{\text{deep}}$ of [18], and to the model *LSTM-CRF* of [14]. To the best of our knowledge the latter is one of the rare work on neural models where results are given also without pre-trained embeddings, allowing a direct comparison. The *LSTM-CRF* model is moreover one of the best models on the WSJ corpus when using embeddings pre-trained with GloVe [64].

The results of the POS tagging task on the WSJ corpus are shown in the table 6. As we can see our model obtains the best results among those not using any pre-trained embeddings. Our results are however worse than those obtained with pre-trained embeddings, which constitute the state-of-the-art on this task. In this respect, we would like to underline that the overall best results are obtained with a neural model described in [56]. This model is only slightly better than the *LSTM-CRF* model, which we outperform when not using pre-trained embeddings. Moreover the model proposed in [56] (*LSTM+LD-RNN* in the table) is very similar to our model.

In order to compare our model to the model LD-RNN$_{\text{deep}}$ also in terms of complexity and computation efficiency, we show in the table 7 the number of parameters as well as the training time on the MEDIA and WSJ corpora. For the sake of completeness, we also report the number of parameters of the other models mentioned in this paper. Except for the model *GRU+CRF* for which we took the number of parameters from the reference [18] (hidden layers of size 200), all the other numbers are computed based on the same layer sizes.

Model	**# of parameters**	**Training time**	
	MEDIA	MEDIA	WSJ
Seq2Biseq$_{le}$	2,139,950	3h30'	16h-17h
LD-RNN$_{\text{deep}}$	2,551,700	1h30'	> 6 days
GRU+CRF [18]	2,328,360	–	–
Seq2seq	1,703,450	–	–
Seq2seq+Att.	2,244,050	–	–

Table 7 – Comparison of the neural models proposed or mentioned in this paper, in terms of number of parameters, and of training time for our model and the the model LD-RNN$_{\text{deep}}$

We can see in the table 7 that the training time for our model is longer than for the model LD-RNN$_{\text{deep}}$ on the MEDIA task. This is because our neural architecture is quite more complex, and since the corpus is relatively small, we can not fuly take advantage of GPU parallelism.

This is confirmed on the WSJ corpus, where the training time of our model is much smaller than the time needed by the LD-RNN$_{\text{deep}}$ model, despite this corpus is quite bigger than MEDIA. [18] The time needed for testing are not reported in the table, we can note that they are negligible for both models, as it never exceeded a few minutes

While the results described in this paper can be considered satisfactory, considering the complexity of our neural network with respect to the LD-RNN$_{\text{deep}}$ model, we were surprised to find out that the gains were not larger on the MEDIA task. At first we thought that our network suffered from overfitting on such a small task, and given the complexity of our network, nothing could be done to solve this problem beyond reducing the total number of parameters. However, after a quick analysis of the output of our model on the MEDIA development data, we found clear signs revealing that our model was actually ignoring the learning signal coming from the backward decoder (eq. 6).

Since our neural network was explicitly designed to take both left and right label-side contexts into account, we thought that the problem was coming from the learning phase. In particular we thought that our model was underfitting due to the problem of *very-long back-propagation paths* described in [22], and which motivated the design of the Transformer model, without recurrent layers and with skip connections to enforce the back-propagation of the learning signal. We adopted a different approach : we applied two different optimizers to the two decoders, one for a negative log-likelihood computed with the output of the backward decoder (only log-p($\overleftarrow{e_i}$), see eq. 6), and another one for the global negative log-likelihood computed from the output of both forward and backward decoders (see equation 8). We note that the forward decoder also uses predictions and hidden states of the backward decoder, the second optimizer thus also refines

18. The model LD-RNN$_{\text{deep}}$ is coded in Octave, and while it can run on GPUs, this framework is not fully optimized to scale on GPUs

the parameters of the backward decoder with left, forward information.

We ran new experiments in exactly the same conditions as described before, the only difference being that we used these two optimizers. The final results are reported in table 4 for MEDIA and in the table 6 for the WSJ, where the model learned using two optimizers is indicated with $\text{Seq2Biseq}_{\text{2-opt}}$.

As we can see in the tables, the results improved on both tasks, on both development and test data, and in terms of all the evaluation metrics. To the best of our knowledge, the results obtained on MEDIA are the best on this task, except for the CER where the model $\text{LD-RNN}_{\text{deep}}$ using class features is still the best (9.8 vs. our 9.93 on the test set). Also, the results obtained on the WSJ corpus are the best obtained without any external resource and without pre-trained embeddings. We leave the integration of pre-trained embeddings as future work.

5 Conclusions

In this article, we propose a new neural architecture for sequence modelling heavily based on GRU recurrent hidden layers. We use these layers to encode long-range contextual information at several levels : words, characters and labels.

Our main contributions are the use of two different decoders for label prediction, one modelling a backward (future, or right) label context, and one for a forward label context. The combination of the two contexts allow our model to take labelling decisions informed by a global context, approximating a global decision function. Another contribution is the use of two different optimizers to optimize separately the two decoders. This improves even further the results obtained on the two evaluation tasks studied in this work.

The results obtained are state-of-the-art on the MEDIA task. On the POS tagging task of the WSJ corpus, our results are state-of-the-art if we do not consider the models that use pre-trained word embeddings, and still close to the state-of-the-art if we do so.

6 Acknowledgements

This work is part of the “Investissements d’Avenir” overseen by the French National Research Agency ANR-10-LABX-0083 (Labex EFL), and is also supported by the ANR DEMOCRAT (Describing and Modelling Reference Chains : Tools for Corpus Annotation and Automatic Processing) project ANR-15-CE38-0008.

Références

[1] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P. : Natural language processing (almost) from scratch. J. Mach. Learn. Res. **12** (2011)

[2] De Mori, R., Bechet, F., Hakkani-Tur, D., McTear, M., Riccardi, G., Tur, G. : Spoken language understanding : A survey. IEEE Signal Processing Magazine **25** (2008) 50–58

[3] Sutskever, I., Vinyals, O., Le, Q.V. : Sequence to sequence learning with neural networks. In : Proceedings of NIPS, Cambridge, MA, USA, MIT Press (2014)

[4] Bahdanau, D., Cho, K., Bengio, Y. : Neural machine translation by jointly learning to align and translate. CoRR **abs/1409.0473** (2014)

[5] Collins, M. : Three generative, lexicalised models for statistical parsing. In : Proceedings of ACL, Stroudsburg, PA, USA, Association for Computational Linguistics (1997) 16–23

[6] Soon, W.M., Ng, H.T., Lim, D.C.Y. : A Machine Learning Approach to Coreference Resolution of Noun Phrases. Computational Linguistics **27** (2001) 521–544

[7] Ng, V., Cardie, C. : Improving Machine Learning Approcahes to Corefrence Resolution. In : Proceedings of ACL'02. (2002) 104–111

[8] Grouin, C., Dinarelli, M., Rosset, S., Wisniewski, G., Zweigenbaum, P. : Coreference resolution in clinical reports. the limsi participation in the i2b2/va 2011 challenge. In : In Proceedings of i2b2/VA 2011 Coreference Resolution Workshop. (2011)

[9] Dinarelli, M., Rosset, S. : Tree representations in probabilistic models for extended named entity detection. In : European Chapter of the Association for Computational Linguistics (EACL), Avignon, France (2012) 174–184

[10] Dinarelli, M., Rosset, S. : Tree-structured named entity recognition on ocr data : Analysis, processing and results. In Chair), N.C.C., Choukri, K., Declerck, T., Dogan, M.U., Maegaard, B., Mariani, J., Odijk, J., Piperidis, S., eds. : Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12), Istanbul, Turkey, European Language Resources Association (ELRA) (2012)

[11] Rush, A.M., Reichart, R., Collins, M., Globerson, A. : Improved parsing and pos tagging using inter-sentence consistency constraints. In : Proceedings of EMNLP-CoNLL, Stroudsburg, PA, USA (2012)

[12] Lee, K., He, L., Lewis, M., Zettlemoyer, L. : End-to-end neural coreference resolution. In : Proceedings of EMNLP, Association for Computational Linguistics (2017)

[13] Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C. : Neural architectures for named entity recognition. arXiv preprint arXiv :1603.01360 (2016)

[14] Ma, X., Hovy, E. : End-to-end sequence labeling via bi-directional lstm-cnns-crf. In : Proceedings of ACL. (2016)

[15] Kemker, R., McClure, M., Abitino, A., Hayes, T.L., Kanan, C. : Measuring catastrophic forgetting in neural networks. In : Thirty-Second AAAI Conference on Artificial Intelligence. (2018)

[16] Augenstein, I., Ruder, S., Søgaard, A. : Multi-task learning of pairwise sequence classification tasks over disparate label spaces. In : Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long Papers), Association for Computational Linguistics (2018) 1896–1906

[17] Vinyals, O., Kaiser, L., Koo, T., Petrov, S., Sutskever, I., Hinton, G.E. : Grammar as a foreign language. CoRR **abs/1412.7449** (2014)

[18] Dinarelli, M., Vukotic, V., Raymond, C. : Label-dependency coding in Simple Recurrent Networks for Spoken Language Understanding. In : Interspeech, Stockholm, Sweden (2017)

[19] Werbos, P. : Backpropagation through time : what does it do and how to do it. In : Proceedings of IEEE. Volume 78. (1990) 1550–1560

[20] Hochreiter, S., Schmidhuber, J. : Long short-term memory. Neural Comput. **9** (1997) 1735–1780

[21] Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., Bengio, Y. : Learning phrase representations using RNN encoder-decoder for statistical machine translation. CoRR **abs/1406.1078** (2014)

[22] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I. : Attention is all you need. (2017)

[23] Vukotic, V., Raymond, C., Gravier, G. : A step beyond local observations with a dialog aware bidirectional GRU network for Spoken Language Understanding. In : Interspeech, San Francisco (2016)

[24] Chiu, J.P.C., Nichols, E. : Named entity recognition with bidirectional lstm-cnns. CoRR **abs/1511.08308** (2015)

[25] Huang, Z., Xu, W., Yu, K. : Bidirectional lstm-crf models for sequence tagging. arXiv preprint arXiv :1508.01991 (2015)

[26] Dupont, Y., Dinarelli, M., Tellier, I. : Label-dependencies aware recurrent neural networks. In : Proceedings of CICling, Budapest, Hungary, LNCS, Springer (2017)

[27] Dinarelli, M., Tellier, I. : Improving recurrent neural networks for sequence labelling. CoRR **abs/1606.02555** (2016)

[28] Dinarelli, M., Tellier, I. : New recurrent neural network variants for sequence labeling. In : Proceedings of the 17th International Conference on Intelligent Text Processing and Computational Linguistics, Konya, Turkey, Lecture Notes in Computer Science (Springer) (2016)

[29] Xia, Y., Tian, F., Wu, L., Lin, J., Qin, T., Yu, N., Liu, T.Y. : Deliberation networks : Sequence generation beyond one-pass decoding. In Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., eds. : Advances in Neural Information Processing Systems 30. Curran Associates, Inc. (2017) 1784–1794

[30] Bonneau-Maynard, H., Ayache, C., Bechet, F., Denis, A., Kuhn, A., Lefèvre, F., Mostefa, D., Qugnard, M., Rosset, S., Servan, S. Vilaneau, J. : Results of the french evalda-media evaluation campaign for literal understanding. In : LREC, Genoa, Italy (2006) 2054–2059

[31] Marcus, M.P., Santorini, B., Marcinkiewicz, M.A. : Building a large annotated corpus of english : The penn treebank. COMPUTATIONAL LINGUISTICS **19** (1993)

[32] Kim, Y., Denton, C., Hoang, L., Rush, A.M. : Structured attention networks. CoRR **abs/1702.00887** (2017)

[33] Simonnet, E., Camelin, N., Deléglise, P., Estève, Y. : Exploring the use of Attention-Based Recurrent Neural Networks For Spoken Language Understanding. In : SLUNIPS, Montreal, Canada (2015)

[34] Elman, J.L. : Finding structure in time. COGNITIVE SCIENCE **14** (1990)

[35] Jordan, M.I. : Serial order : A parallel, distributed processing approach. (1989)

[36] Bengio, Y., Simard, P., Frasconi, P. : Learning long-term dependencies with gradient descent is difficult. Trans. Neur. Netw. **5** (1994) 157–166

[37] Mesnil, G., Dauphin, Y., Yao, K., Bengio, Y., Deng, L., Hakkani-Tur, D., He, X., Heck, L., Tur, G., Yu, D., Zweig, G. : Using recurrent neural networks for slot filling in spoken language understanding. IEEE/ACM TASLP (2015)

[38] Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C. : A neural probabilistic language model. JOURNAL OF MACHINE LEARNING RESEARCH **3** (2003) 1137–1155

[39] Raymond, C., Riccardi, G. : Generative and discriminative algorithms for spoken language understanding. In : Interspeech, Antwerp, Belgium (2007)

[40] Dinarelli, M., Moschitti, A., Riccardi, G. : Concept segmentation and labeling for conversational speech. In : Proceedings of the International Conference of the Speech Communication Assosiation (Interspeech), Brighton, U.K. (2009)

[41] Hahn, S., Dinarelli, M., Raymond, C., Lefèvre, F., Lehen, P., De Mori, R., Moschitti, A., Ney, H., Riccardi, G. : Comparing stochastic approaches to spoken language understanding in multiple languages. IEEE TASLP **99** (2010)

[42] Dinarelli, M. : Spoken Language Understanding : from Spoken Utterances to Semantic Structures. PhD thesis, International Doctoral School in Information and Communication Technology, Dipartimento di Ingegneria e Scienza dell' Informazione, via Sommarive 14, 38100 Povo di Trento (TN), Italy (2010)

[43] Dinarelli, M., Rosset, S. : Hypotheses selection criteria in a reranking framework for spoken language understanding. In : Conference of Empirical Methods for Natural Language Processing, Edinburgh, U.K. (2011) 1104–1115

[44] Dinarelli, M., Moschitti, A., Riccardi, G. : Discriminative reranking for spoken language understanding. IEEE TASLP **20** (2011) 526–539

[45] Vapnik, V.N. : Statistical Learning Theory. John Wiley and Sons (1998)

[46] Lafferty, J., McCallum, A., Pereira, F. : Conditional random fields : Probabilistic models for segmenting and labeling sequence data. In : Proceedings of ICML, Williamstown, USA (2001)

[47] Mesnil, G., He, X., Deng, L., Bengio, Y. : Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In : Interspeech. (2013)

[48] Yao, K., Zweig, G., Hwang, M.Y., Shi, Y., Yu, D. : Recurrent neural networks for language understanding, Interspeech (2013)

[49] Vukotic, V., Raymond, C., Gravier, G. : Is it time to switch to word embedding and recurrent neural networks for spoken language understanding ? In : InterSpeech. (2015)

[50] Yao, K., Peng, B., Zhang, Y., Yu, D., Zweig, G., Shi, Y. : Spoken language understanding using long short-term memory neural networks, IEEE (2014)

[51] Wang, C. : Network of recurrent neural networks. CoRR **abs/1710.03414** (2017)

[52] Holland, J.H. : Emergence : From Chaos to Order. Perseus Publishing (1999)

[53] Arthur, W.B. : On the evolution of complexity. Working papers, Santa Fe Institute (1993)

[54] Levy, O., Lee, K., FitzGerald, N., Zettlemoyer, L. : Long short-term memory as a dynamically computed element-wise weighted sum. In : Proceedings of ACL. (2018)

[55] Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., Kaiser, L. : Universal transformers. CoRR **abs/1807.03819** (2018)

[56] Zhang, Y., Chen, H., Zhao, Y., Liu, Q., Yin, D. : Learning tag dependencies for sequence tagging. In : International Joint Conference on Artificial Intelligence (IJCAI). (2018)

[57] Ramshaw, L., Marcus, M. : Text chunking using transformation-based learning. In : Proceedings of the 3rd Workshop on Very Large Corpora, Cambridge, MA, USA (1995) 84–94

[58] He, K., Zhang, X., Ren, S., Sun, J. : Delving deep into rectifiers : Surpassing human-level performance on imagenet classification. In : 2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015. (2015) 1026–1034

[59] Bengio, Y. : Practical recommendations for gradient-based training of deep architectures. CoRR **abs/1206.5533** (2012)

[60] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A. : Automatic differentiation in pytorch. In : NIPS-W. (2017)

[61] Gruslys, A., Munos, R., Danihelka, I., Lanctot, M., Graves, A. : Memory-efficient backpropagation through time. (2016) 4125–4133

[62] Yeh, A. : More accurate tests for the statistical significance of result differences. (In : Proceedings of Coling)

[63] Padó, S. : User's guide to `sigf` : Significance testing by approximate randomisation. (2006)

[64] Pennington, J., Socher, R., Manning, C.D. : Glove : Global vectors for word representation. In : Empirical Methods in Natural Language Processing (EMNLP). (2014) 1532–1543