





Introducing the Hidden Neural Markov Chain Framework

Elie Azeraf^{1,2}^a, Emmanuel Monfrini¹^b, Emmanuel Vignon²^c and Wojciech Pieczynski²^d

¹Watson Department, IBM GBS, avenue de l'Europe, Bois-Colombes, France

²SAMOVAR, CNRS, Telecom SudParis, Institut Polytechnique de Paris, Evry, France

Keywords: Hidden Markov Model, Entropic Forward-Backward, Recurrent Neural Network, Sequence Labeling, Hidden Neural Markov Chain.

Abstract: Nowadays, neural network models achieve state-of-the-art results in many areas as computer vision or speech processing. For sequential data, especially for Natural Language Processing (NLP) tasks, Recurrent Neural Networks (RNNs) and their extensions, the Long Short Term Memory (LSTM) network and the Gated Recurrent Unit (GRU), are among the most used models, having a “term-to-term” sequence processing. However, if many works create extensions and improvements of the RNN, few have focused on developing other ways for sequential data processing with neural networks in a “term-to-term” way. This paper proposes the original Hidden Neural Markov Chain (HNMC) framework, a new family of sequential neural models. They are not based on the RNN but on the Hidden Markov Model (HMM), a probabilistic graphical model. This neural extension is possible thanks to the recent Entropic Forward-Backward algorithm for HMM restoration. We propose three different models: the classic HNMC, the HNMC2, and the HNMC-CN. After describing our models’ whole construction, we compare them with classic RNN and Bidirectional RNN (BiRNN) models for some sequence labeling tasks: Chunking, Part-Of-Speech Tagging, and Named Entity Recognition. For every experiment, whatever the architecture or the embedding method used, one of our proposed models has the best results. It shows this new neural sequential framework’s potential, which can open the way to new models, and might eventually compete with the prevalent BiLSTM and BiGRU.


1 INTRODUCTION


During the last years, neural networks models (Goodfellow et al., 2016; LeCun et al., 2015) show impressive performances in many areas, as computer vision or speech processing. Among them, Natural Language Processing (NLP) has one of the most significant expansions. The Recurrent Neural Network (Rumelhart et al., 1985; Jordan, 1990; Jozefowicz et al., 2015) (RNN) based models, treating text as sequential data, are among the most often used models for NLP tasks, especially the Long Short Term Memory network (LSTM) (Hochreiter and Schmidhuber, 1997) and the Gated Recurrent Unit (GRU) (Chung et al., 2014). They can cover all textual applications as word embedding (Akbik et al., 2018) or text translation (Sutskever et al., 2014). They are the most prevalent sequential models with neural networks, having a


term-to-term data processing.


However, if many works have been done to create extensions of the RNN, very few of them focused on a different way to use neural networks to treat sequential data with term-to-term processing. There are Transformer (Vaswani et al., 2017) based models, as BERT (Devlin et al., 2018) or XLNet (Yang et al., 2019), but they have a different structure as they catch all the observations of the sequence in one time (under padding limitations) and require many more parameters and training power. In this paper, we only focus on neural models with term-to-term processing.

Among the sequential models, one of the most popular is the Hidden Markov Model (HMM) (Stratonovich, 1965; Baum and Petrie, 1966; Rabiner and Juang, 1986), also called Hidden Markov Chain, which is a probabilistic graphical model (Koller and Friedman, 2009). In this paper, we propose a new framework of sequential neural models based on HMM, named Hidden Neural Markov Chains (HNMCs), composed of the classic HNMC, the HNMC of order 2 (HNMC2), and the HNMC with complexified noise (HNMC-CN). As RNN, they are neural

^a <https://orcid.org/0000-0003-3595-0826>

^b <https://orcid.org/0000-0002-7648-2515>

^c <https://orcid.org/0000-0002-9560-9630>

^d <https://orcid.org/0000-0002-1371-2627>

term-to-term models for sequential data processing. Their interest is due to a new way of HMM's posterior marginal distribution computation based on the Entropic Forward-Backward (EFB) algorithm, which allows considering arbitrary features (Azeraf et al., 2020) with HMM. We adapt EFB to HMM of order 2 (HMM2) and HMM with complexified noise (HMM-CN), presented in the next section. Therefore, we present HNMC as the HMM neural extension, HNMC2 as the HMM2 one, and HNMC-CN as the HMM-CN one.

The paper is organized as follows. The next section presents the HMM model, its EFB algorithm, the HMM2, the HMM-CN, and their EFB algorithms. Then we introduce the HNMC, the HNMC2, and the HNMC-CN models. We specify the computational graph and related training process of the HNMC. We also describe the differences between our proposed models and some previous ones combining HMM and neural networks. The fourth part is devoted to experiments. We compare our models with RNN and Bidirectional RNN (BiRNN) (Schuster and Paliwal, 1997) for different sequence labeling tasks: Part-Of-Speech (POS) tagging, Chunking, and Named-Entity-Recognition (NER). We implement many architectures with various embedding methods to reach a convincing empirical comparison. We only compare with RNN and BiRNN, as the latter's extensions to catch longer memory information, leading to LSTM and GRU, is discussed as the perspectives for HNMC based models in the last section.

2 HIDDEN MARKOV MODEL

2.1 Description of the HMM

The Hidden Markov Model is a sequential model created sixty years ago and used in numerous applications (Rabiner, 1989; Li et al., 2000; Brants, 2000). It allows the restoration of a hidden sequence from an observed one.

Let $x_{1:T} = (x_1, \dots, x_T)$ be a hidden realization of a stochastic process, taking its values in $\Lambda_X = \{\lambda_1, \dots, \lambda_N\}$, and let $y_{1:T} = (y_1, \dots, y_T)$ be an observed realization of a stochastic one, taking its values in $\Omega_Y = \{\omega_1, \dots, \omega_M\}$. The couple $(x_{1:T}, y_{1:T})$ is a HMM if its probabilistic law can be written:

$$p(x_{1:T}, y_{1:T}) = p(x_1)p(y_1|x_1)p(x_2|x_1)p(y_2|x_2)\dots p(x_T|x_{T-1})p(y_T|x_T)$$

The probabilistic oriented graph of the HMM is given in figure 1.

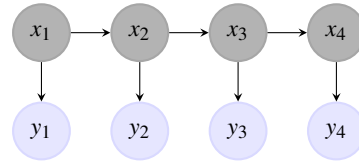


Figure 1: Probabilistic oriented graph of the HMM.

2.2 The Entropic Forward-Backward Algorithm for HMM

There are different ways to restore a hidden chain from an observed one using the HMM. With the Maximum A Posteriori criterion (MAP), one can use the classic Viterbi (Viterbi, 1967) algorithm. About the Maximum Posterior Mode (MPM), one can use the classic Forward-Backward (Rabiner, 1989) (FB) one. However, both Viterbi and FB algorithms use probabilities $p(y_t|x_t)$, making them impossible to consider arbitrary features of the observations (Jurafsky, 2000; Sutton and McCallum, 2006), especially the output of a neural network function. To correct this default, the Entropic Forward Backward (EFB) algorithm specified below computes the MPM using $p(x_t|y_t)$ and can take into account any features (Azeraf et al., 2020). This makes possible the neural extension of the HMM we are going to present.

For stationary HMM we consider in the whole paper, the EFB deals with the following parameters:

- $\pi(i) = p(x_t = \lambda_i)$;
- $a_i(j) = p(x_{t+1} = \lambda_j | x_t = \lambda_i)$;
- $L_y(i) = p(x_t = \lambda_i | y_t = y)$;

The MPM restoration method we consider consists of maximization of the probabilities $p(x_t = \lambda_i | y_{1:T})$. They are given from entropic forward α and entropic backward β functions with:

$$p(x_t = \lambda_i | y_{1:T}) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^N \alpha_t(j)\beta_t(j)} \quad (1)$$

Entropic forward functions are computed recursively as follows:

- For $t = 1$:

$$\alpha_1(i) = L_{y_1}(i)$$

- For $1 \leq t < T$:

$$\alpha_{t+1}(i) = \frac{L_{y_{t+1}}(i)}{\pi(i)} \sum_{j=1}^N \alpha_t(j)a_j(i) \quad (2)$$

And the entropic backward ones:

- For $t = T$:

$$\beta_T(i) = 1$$

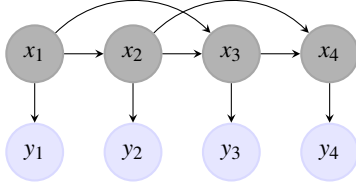


Figure 2: Probabilistic oriented graph of the HMM of order 2.

- For $1 \leq t < T$:

$$\beta_t(i) = \sum_{j=1}^N \frac{L_{y_{t+1}}(j)}{\pi(j)} \beta_{t+1}(j) a_i(j) \quad (3)$$

One can normalize values at each time in (2) and (3) to avoid underflow problems without modifying the probabilities' computation.

2.3 EFB Algorithm for HMM of Order 2

In this paragraph, we describe an extension of EFB above to HMM2, which allows to catch longer memory information than the HMM. The probabilistic law of $(x_{1:T}, y_{1:T})$ for the HMM2 is:

$$p(x_{1:T}, y_{1:T}) = p(x_1) p(x_2|x_1) p(x_3|x_1, x_2) \dots p(x_T|x_{T-2}, x_{T-1}) p(y_1|x_1) p(y_2|x_2) \dots p(y_T|x_T)$$

Its probabilistic graph is given in figure 2.

We introduce the following notation to present the EFB algorithm for HMM2:

$$a_{i,j}^2(k) = p(x_{t+2} = \lambda_k | x_t = \lambda_i, x_{t+1} = \lambda_j)$$

The EFB algorithm for HMM2 is the following:

- For $t = 1$:

$$p(x_1 = \lambda_i | y_{1:T}) = \frac{\sum_j \alpha_2^2(i, j) \beta_2^2(i, j)}{\sum_k \sum_j \alpha_2^2(k, j) \beta_2^2(k, j)}$$

- For $2 \leq t \leq T$:

$$p(x_t = \lambda_i | y_{1:T}) = \frac{\sum_j \alpha_t^2(j, i) \beta_t^2(j, i)}{\sum_k \sum_j \alpha_t^2(j, k) \beta_t^2(j, k)}$$

The entropic forward-2 functions α^2 are computed with the following recursion:

- For $t = 2$:

$$\alpha_2^2(j, i) = L_{y_1}(j) a_j(i) \frac{L_{y_2}(i)}{\pi(i)}$$

- And for $2 \leq t < T$:

$$\alpha_{t+1}^2(j, i) = \sum_k \alpha_t^2(k, j) a_{k,j}^2(i) \frac{L_{y_{t+1}}(i)}{\pi(i)}$$

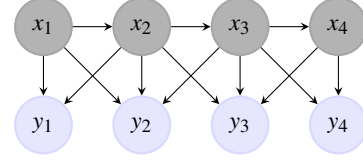


Figure 3: Probabilistic oriented graph of the HMM-CN.

And the backward-2 functions β^2 with the following one:

- For $t = T$:

$$\beta_T^2(j, i) = 1$$

- And for $2 \leq t < T$:

$$\beta_t^2(j, i) = \sum_k \beta_{t+1}^2(i, k) a_{j,i}^2(k) \frac{L_{y_{t+1}}(k)}{\pi(k)}$$

2.4 HMM-CN and Related EFB

This paragraph describes the new HMM-CN model with related new EFB. It is another extension of HMM aiming to improve its results. Its probabilistic oriented graph is presented in figure 3.

In this case, the hidden sequence is still a Markov chain, and the conditional law of the observation y_t given $x_{1:T}$ depends on x_{t-1}, x_t , and x_{t+1} , implying stronger dependency with the hidden chain. The HMM-CN has the probabilistic law:

$$p(x_{1:T}, y_{1:T}) = p(x_1) p(x_2|x_1) p(x_3|x_2) \dots p(x_T|x_{T-1}) p(y_1|x_1, x_2) p(y_2|x_1, x_2, x_3) \dots p(y_T|y_{T-1}, y_T)$$

To present the EFB algorithm for HMM-CN, we set:

- $I_{j,y}(i) = p(x_{t+1} = \lambda_i | x_t = \lambda_j, y_t = y)$
- $J_{j,y}(i) = p(x_t = \lambda_i | x_{t+1} = \lambda_j, y_{t+1} = y)$

The goal of the EFB algorithm is to compute $p(x_t = \lambda_i | y_{1:T})$, using $I_{j,y}(i)$ and $J_{j,y}(i)$ above, we show¹:

$$p(x_t = \lambda_i | y_{1:T}) = \frac{\alpha_t^{CN}(i) \beta_t^{CN}(i)}{\sum_j \alpha_t^{CN}(j) \beta_t^{CN}(j)},$$

with the entropic forward-cn functions α^{CN} computed with the following recursion:

- For $t = 1$:

$$\alpha_1^{CN}(i) = L_{y_1}(i)$$

- And for $1 \leq t < T$:

$$\alpha_{t+1}^{CN}(i) = \sum_j \alpha_t^{CN}(j) I_{j,y_t}(i) \frac{L_{y_{t+1}}(i) J_{i,y_{t+1}}(j)}{\pi(j) a_j(i)} \quad (4)$$

¹Proofs of the EFB algorithms for HMM2 and HMM-CN are available here.

And the entropic backward-cn functions β^{CN} computed with the following one:

- For $t = T$:

$$\beta_T^{CN}(i) = 1$$

- And for $1 \leq t < T$:

$$\beta_t^{CN}(i) = \sum_j \beta_{t+1}^{CN}(j) I_{i,y_t}(j) \frac{L_{y_{t+1}}(j) J_{j,y_{t+1}}(i)}{\pi(i) a_i(j)} \quad (5)$$

3 HIDDEN NEURAL MARKOV CHAIN FRAMEWORK

3.1 Construction of the HNMC

To extend the HMM considered above to the HNMC, we have to model the three functions, π , a , and L , with a feedforward neural network function modeling $\frac{L_{y_{t+1}}(i)}{\pi(i)} a_j(i)$. This neural network function has y_{t+1} concatenated with the one-hot encoding of j as input, and outputs a positive vector of size N . To do that, we use a last positive activation function as the exponential, the sigmoid, or a modified Exponential Linear Unit (mELU):

$$f(x) = \begin{cases} 1+x & \text{if } x > 0 \\ e^x & \text{otherwise.} \end{cases}$$

Then, we apply the EFB algorithm for sequence restoration. The first step of the algorithm is performed thanks to the introduction of an initial state, which can be drawn randomly or equals to a constant different from 0. Therefore, we have constructed the HNMC, a new model able to process sequential data in a ‘‘term-to-term’’ way with neural network functions.

We can stack HNMCs to add hidden layers, similarly to the stacked RNN practice, to achieve greater model complexity. The output of a first HNMC based EFB restoration layer becoming the input of the next one, and so on, applying the EFB layer after layer. For example, a computational graph of a HNMC composed of four layers is specified in figure 4. In the general case, we have $K + 2$ layers:

- An input layer y ;
- K hidden layers $h^{(1)}, h^{(2)}, \dots, h^{(K)}$;
- An output layer x .

We consider that $(H^{(1)}, Y)$, $(H^{(2)}, H^{(1)})$, ..., $(H^{(K)}, H^{(K-1)})$, are HMMs, and the last layer $H^{(K)}$ is connected with the output layer x thanks to a feedforward neural network function denoted f . Finally, we compute for each $t \in \{1, \dots, T\}$, x_t from $y_{1:T}$ as follows:

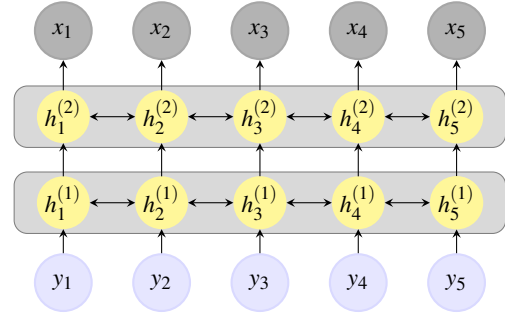


Figure 4: Computational graph of the HNMC with two hidden layers.

1. Computing $h^{(1)}$ from $y_{1:T}$ using EFB;
2. Computing $h^{(2)}$ from $h^{(1)}$ using EFB, considering $h^{(1)}$ as the observations; then compute $h^{(3)}$ from $h^{(2)}$ using EFB, ...
3. Computing $h^{(K)}$ from $h^{(K-1)}$ using EFB, considering $h^{(K-1)}$ as the observations;
4. Computing $x_t = f(h_t^{(K)})$, x_t is the output vector of probabilities of the different states at time t .

Let us notice that, from a probabilistic point of view, this stacked HNMC can be seen as a particular Triplet Markov Chain (Pieczynski et al., 2003) having $K + 2$ layers, and our restoration method would be an approximation of this model.

Thus, the HNMC can be used as a sequential neural model with term-to-term processing, like the RNN. However, unlike the latter, the HNMC uses all the observation $y_{1:T}$ to restore x_t , whereas the RNN uses only $y_{1:t}$. One can use the BiRNN to correct this default, consisting of applying a RNN from right to left, another one from left to right, then concatenate the outputs.

3.2 Neural Extension of HMM2 and HMM-CN

Neural extensions of HMM2 and HMM-CN follow the same principles as for HMM. For the HMM2, we model $a_{k,j}^2(i) \frac{L_{y_{t+1}}(i)}{\pi(i)}$ with a feedforward neural functions with a positive last activation function, taking as input y_{t+1} and the one-hot encoding of (k, j) . This model is denoted HMNC2.

Concerning the HMM-CN, we use two different neural functions: one to model $\frac{J_{i,y_{t+1}}(j)}{\pi(j)}$, and the other one to model $I_{j,y_t}(i) \frac{L_{y_{t+1}}(i)}{a_j(i)}$, with the relevant inputs, and positive outputs. This model is denoted HMNC-CN.

3.3 Learning HNMC based Models' Parameters

To learn the different parameters of each of our new models, we consider the backpropagation algorithm (LeCun et al., 1988; LeCun et al., 1989) frequently used for neural network learning. Given a loss, for example the cross-entropy L_{CE} , θ a parameter of one of the model's functions, and a sequence $y_{1:T}$, we compute $\frac{\partial L_{CE}}{\partial \theta}$ with gradient backpropagation over all the intermediary variables. Then, we apply the gradient descent (Ruder, 2016) algorithm:

$$\theta^{(new)} = \theta - \kappa \frac{\partial L_{CE}}{\partial \theta}$$

with κ the learning rate.

As for any neural network architectures, we can apply the gradient descent algorithm for HNMC based models. Therefore, we can create different architectures and combine them with other neural network models as Convolutional Neural Networks (LeCun et al., 1999) or feedforward ones.

3.4 Related Works

The combination of HMM with neural networks starts in the 1990s (Bengio et al., 1994), focusing on the concatenation of the two models. Nowadays, few papers deal with the subject. The closest model to HNMC is the neural HMM proposed in (Tran et al., 2016). However, the proposed method is not EFB based, and neural networks model different parameters from those considered in this paper. Indeed, they model $p(y_i|x_i = \lambda_i)$. This implies a sum over all the possible observations to be computed, which considerably increases the number of parameters for NLP applications, where observations are words. It also avoids the combination with embedding methods, aiming to convert a word into a continuous vector. Moreover, the proposed training method is based on the Baum-Welch algorithm with Expectation-Maximization (Welch, 2003), or Direct Marginal Likelihood (Salakhutdinov et al., 2003), so the ability to create various architectures as it is done with RNN is not trivial. It focuses on unsupervised tasks, which is not the case for HNMC. Comparable works can be found in (Wang et al., 2017; Wang et al., 2018). Therefore, the proposed HNMC, based on different neuralized parameters with gradient descent training and aiming a different objective, is an original way to combine HMM with neural networks.

4 EXPERIMENTS

This section presents some experimental results comparing the RNN, the BiRNN, the HNMC, the HNMC2, and the HNMC-CN. After some preliminary presentations of the different tasks and the word embedding process, we create different architectures for all the models and test them for sequence labeling applications. Motivations to the choice of comparing our models with RNN and BiRNN are discussed in perspectives.

4.1 Sequence Labeling Tasks

We select sequence labeling applications as they are the most intuitive tasks to apply a sequential model in the NLP framework. It consists of labeling every word in a sentence with a specific tag. We apply the different models to POS Tagging, Chunking, and NER, which are among the most popular sequence labeling applications.

The POS tagging consists of labeling every word with its grammatical function as noun (*NOUN*), verb (*VERB*), determinant (*DET*), etc. For example, the sentence (*Batman, is, the, vigilante, of, Gotham, .*) has the labels (*NOUN, VERB, DET, NOUN, PREP, NOUN, PUNCT*). The accuracy score is used to evaluate this task.

Chunking consists of segmenting a sentence with a more global point of view than the POS tagging. It decomposes the sentence by groups of words linked by a syntactic function, as a noun phrase (*NP*), a verb phrase (*VP*), an adjective phrase (*ADJP*), among others. For example, the sentence (*The, worst, enemy, of, Batman, is, the, Joker, .*) has the following chunk tags (*NP, NP, NP, PP, NP, VP, NP, NP, O*). *O* denotes a word having no chunk tag. The F_1 score is used to measure the performance of this task.

The objective of the NER is to find the different entities in a sentence. Entities can be the name of a person (*PER*), of a city (*LOC*), or of a company (*ORG*). For example, the sentence (*Bruce, Wayne, ,, a, citizen, of, Gotham, ,, is, the, secret, identity, of, Batman, .*) can have the entities (*PER, PER, O, O, O, O, LOC, O, O, O, O, O, PER, O*). The entity set depends on the use-case, and one can it change according to the objective. As for Chunking, the F_1 score is used to evaluate the performances of a model.

For our experiments, we use three reference datasets: Universal Dependencies English (UD En) (Nivre et al., 2016) for POS Tagging, CoNLL 2000 (Tjong Kim Sang and Buchholz, 2000) for Chunking, and we use general entities with the CoNLL 2003 (Tjong Kim Sang and De Meulder, 2003) dataset for

Table 1: Results of the different models for POS Tagging, Chunking, and NER, for the Architecture 1 - the model only.

Architecture 1					
	RNN	BiRNN	HNMC	HNMC2	HNMC-CN
POS Ext UD	88.40% ± 0.02	91.38% ± 0.04	90.98% ± 0.03	91.33% ± 0.04	92.62% ± 0.04
Ch GloVe 00	86.68 ± 0.08	90.76 ± 0.55	87.77 ± 0.13	88.18 ± 0.04	92.02 ± 0.03
NER FT 03	81.91 ± 0.14	82.62 ± 0.56	83.41 ± 0.10	83.49 ± 0.06	87.49 ± 0.19

Table 2: Results of the different models for POS Tagging, Chunking, and NER, for the Architecture 2 - the model followed by a feedforward neural function, the hidden size is denoted HS.

Architecture 2						
	RNN	BiRNN	HNMC	HNMC2	HNMC-CN	HS
POS Ext UD	89.84% ± 0.04	93.07% ± 0.05	92.77% ± 0.06	93.01% ± 0.04	93.29% ± 0.05	50
Ch GloVe 00	93.85 ± 0.06	95.02 ± 0.11	95.43 ± 0.09	95.59 ± 0.13	95.36 ± 0.07	32
NER FT 03	84.53 ± 0.21	87.52 ± 0.13	88.22 ± 0.13	88.47 ± 0.05	89.40 ± 0.03	20

Table 3: Results of the different models for POS Tagging, Chunking, and NER, for the Architecture 3 - two models stacked, the hidden size is denoted HS.

Architecture 3						
	RNN	BiRNN	HNMC	HNMC2	HNMC-CN	HS
POS Ext UD	89.20% ± 0.09	92.80% ± 0.21	92.73% ± 0.12	92.97% ± 0.08	93.36% ± 0.03	50
Ch GloVe 00	93.13 ± 0.14	94.91 ± 0.09	95.53 ± 0.13	95.59 ± 0.06	95.40 ± 0.14	32
NER FT 03	85.10 ± 0.12	88.68 ± 0.31	88.02 ± 0.19	88.66 ± 0.33	89.37 ± 0.12	20

NER².

4.2 Word Embedding Methods

A sentence is composed of textual data; this type of data cannot be the input of feedforward neural network functions. Indeed, these functions have as input a numerical vector or scalar. Our experiments' first step consists of a pre-processing task to convert a word into a numerical vector, called word embedding, or word encoding. In order to make our conclusions independent from embedding, we use three different embedding methods: GloVe (Pennington et al., 2014), FastText (Bojanowski et al., 2017), and EXT encoding (Komninos and Manandhar, 2016).

4.3 The Different Architectures

To compare the different models for the different sequence labeling tasks, we implement three architectures for each model:

- Architecture 1: only the model;

²All these datasets are freely available: UD En on the website <https://universaldependencies.org/#language->, CoNLL 2000, for example, with NLTK (Loper and Bird, 2002) library, and CoNLL 2003 after a demand on <https://www.clips.uantwerpen.be/conll2003/ner/>

- Architecture 2: the model followed with a feed-forward neural network function, equivalent of the figure 4 with the layers $(y, h^{(1)}, x)$ for the HNMC;
- Architecture 3: two models stacked, equivalent of the figure 4 with the layers $(y, h^{(1)}, h^{(2)})$ for the HNMC.

4.4 Experimental Details

Every model is programmed in python using PyTorch (Paszke et al., 2019) library for automatic differentiation, and Flair library (Akbik et al., 2019) for word encoding. The loss function is the cross-entropy. All the different parameters are modeled with feedforward neural networks without hidden layers, equivalent to the logistic regression. About the activation functions, the HNMC based models always use mELU. For the RNN and BiRNN, we use them as usual, with hyperbolic tangent functions. Every model uses the softmax function at the end of the architecture to output probabilities. We use Adam optimizer (Kingma and Ba, 2014) for all experiments, with a mini-batch size of 32. For architecture 1, the learning rate equals 0.005. We use different learning rates for the different layers for the other architectures: 0.05, then 0.005. This configuration gives the best experimental results for every model.

4.5 Results

For each architecture, we realize three experiments: POS Tagging with UD En using EXT (POS Ext UD), Chunking with CoNLL 2000 using GloVe (Ch GloVe 00), and NER with CoNLL 2003 using Fast-Text (NER FT 03). Each experiment is done five times; we report the mean and the 95%-confidence interval in Table 1, Table 2, and Table 3, with the different sizes of hidden layers, denoted HS.

First of all, we can notice that HNMC is always better than RNN. It is certainly because HNMC uses all the observations to restore any hidden variable, making it a bidirectional alternative to the RNN without increasing the number of parameters, which are slightly equivalent. As expected, the HNMC2 achieves better results than the HNMC, and therefore the RNN. However, HNMC2 does not reach BiRNN scores, except in some cases, especially for Chunking.

Another interesting comparison concerns HNMC-CN and BiRNN. Indeed, the HNMC-CN achieves better results than the BiRNN for every experiment. It is a promising result, as prevalent models as BiLSTM and BiGRU are based on the BiRNN. Therefore, the HNMC-CN can be an alternative to the BiRNN for sequence labeling applications. These different results, comparing HNMC based models with RNN and BiRNN, show the proposed sequential neural framework's potential.

5 CONCLUSION AND PERSPECTIVES

We have presented the HNMC framework, a new family of sequential neural models, introducing the classic HNMC, the HNMC2, and the HNMC-CN. We have compared these three models with the RNN and the BiRNN ones. On the one hand, the HNMC achieves better results than the RNN with an equivalent number of parameters. On the other hand, the HNMC-CN has achieved better results than BiRNN for the different sequence labeling tasks.

As a promising perspective, we can extend the HNMC-CN with long-memory methods, as BiRNN is extended to BiLSTM and BiGRU. Therefore, these extensions of HNMC-CN are expected to compete with BiLSTM and BiGRU. It is a challenging perspective, as these models are the most prevalent ones for sequential data processing.

REFERENCES

- Akbik, A., Bergmann, T., Blythe, D., Rasul, K., Schweter, S., and Vollgraf, R. (2019). Flair: An easy-to-use framework for state-of-the-art nlp. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59.
- Akbik, A., Blythe, D., and Vollgraf, R. (2018). Contextual String Embeddings for Sequence Labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649.
- Azeraf, E., Monfrini, E., Vignon, E., and Pieczynski, W. (2020). Hidden Markov Chains, Entropic Forward-Backward, and Part-Of-Speech Tagging. *arXiv preprint arXiv:2005.10629*.
- Baum, L. E. and Petrie, T. (1966). Statistical inference for probabilistic functions of finite state Markov chains. *The annals of mathematical statistics*, 37(6):1554–1563.
- Bengio, Y., LeCun, Y., and Henderson, D. (1994). Globally trained handwritten word recognizer using spatial representation, convolutional neural networks, and hidden Markov models. In *Advances in neural information processing systems*, pages 937–944.
- Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.
- Brants, T. (2000). TnT – a statistical part-of-speech tagger. In *Sixth Applied Natural Language Processing Conference*, pages 224–231, Seattle, Washington, USA. Association for Computational Linguistics.
- Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. <http://www.deeplearningbook.org>.
- Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Jordan, M. I. (1990). Attractor dynamics and parallelism in a connectionist sequential machine. In *Artificial neural networks: concept learning*, pages 112–127.
- Jozefowicz, R., Zaremba, W., and Sutskever, I. (2015). An empirical exploration of recurrent network architectures. In *International conference on machine learning*, pages 2342–2350.
- Jurafsky, D. (2000). *Speech & language processing*. Pearson Education India.
- Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*.

- Komninos, A. and Manandhar, S. (2016). Dependency based embeddings for sentence classification tasks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1490–1500, San Diego, California. Association for Computational Linguistics.
- LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.
- LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Back-propagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.
- LeCun, Y., Haffner, P., Bottou, L., and Bengio, Y. (1999). Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pages 319–345. Springer.
- LeCun, Y., Touresky, D., Hinton, G., and Sejnowski, T. (1988). A theoretical framework for back-propagation. In *Proceedings of the 1988 connectionist models summer school*, volume 1, pages 21–28. CMU, Pittsburgh, Pa: Morgan Kaufmann.
- Li, J., Najmi, A., and Gray, R. M. (2000). Image classification by a two-dimensional hidden Markov model. *IEEE transactions on signal processing*, 48(2):517–533.
- Loper, E. and Bird, S. (2002). NLTK: the natural language toolkit. *arXiv preprint cs/0205028*.
- Nivre, J., De Marneffe, M.-C., Ginter, F., Goldberg, Y., Hajic, J., Manning, C. D., McDonald, R., Petrov, S., Pyysalo, S., Silveira, N., et al. (2016). Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666.
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037.
- Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.
- Pieczynski, W., Hulard, C., and Veit, T. (2003). Triplet Markov chains in hidden signal restoration. In *Image and Signal Processing for Remote Sensing VIII*, volume 4885, pages 58–68. International Society for Optics and Photonics.
- Rabiner, L. and Juang, B. (1986). An introduction to hidden Markov models. *IEEE ASSP Magazine*, 3(1):4–16.
- Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.
- Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.
- Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.
- Salakhutdinov, R., Roweis, S. T., and Ghahramani, Z. (2003). Optimization with EM and expectation-conjugate-gradient. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 672–679.
- Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.
- Stratonovich, R. L. (1965). Conditional Markov processes. In *Non-linear transformations of stochastic processes*, pages 427–453. Elsevier.
- Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.
- Sutton, C. and McCallum, A. (2006). An introduction to conditional random fields for relational learning. *Introduction to statistical relational learning*, 2:93–128.
- Tjong Kim Sang, E. F. and Buchholz, S. (2000). Introduction to the CoNLL-2000 shared task chunking. In *Fourth Conference on Computational Natural Language Learning and the Second Learning Language in Logic Workshop*.
- Tjong Kim Sang, E. F. and De Meulder, F. (2003). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.
- Tran, K., Bisk, Y., Vaswani, A., Marcu, D., and Knight, K. (2016). Unsupervised neural hidden markov models. *arXiv preprint arXiv:1609.09007*.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.
- Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269.
- Wang, W., Alkhouli, T., Zhu, D., and Ney, H. (2017). Hybrid neural network alignment and lexicon model in direct HMM for statistical machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 125–131.
- Wang, W., Zhu, D., Alkhouli, T., Gan, Z., and Ney, H. (2018). Neural hidden Markov model for machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 377–382.
- Welch, L. R. (2003). Hidden Markov models and the Baum-Welch algorithm. *IEEE Information Theory Society Newsletter*, 53(4):10–13.
- Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763.