Smart Agricultural Technology 4 (2023) 100152

Contents lists available at ScienceDirect

Smart Agricultural Technology

journal homepage: www.journals.elsevier.com/smart-agricultural-technology

Methodology for multi-temporal prediction of crop rotations using recurrent neural networks

Ambre Dupuis [*,a,b,c], Camélia Dadouchi [a,b,c], Bruno Agard [a,b,c]

[a] Laboratoire en Intelligence des Données (LID), Canada
[b] Centre Interuniversitaire de Recherche sur les Réseaux dEntreprise, la Logistique et le Transport (CIRRELT), Canada
[c] Département de mathématiques et génie industriel Polytechnique Montréal, CP 6079, succ. Centre-Ville, Montréal, Québec, Canada

ARTICLE INFO

Keywords:
Agriculture 4.0
Crop rotation
Deep learning
Seq2Seq
LSTM

ABSTRACT

In a context of growing demand for food and the scarcity of natural resources, the development of more sustainable agriculture is imperative. This means it is necessary to limit the environmental impact of agricultural activities on soil and water and to be mindful of the carbon footprint, while maintaining crop yields and economic benefits for producers. Crop rotation is a valuable tool in sustainable agriculture, but this technique has to be appropriately coupled with sustainable fertilization plans to optimize crops. The proposed methodology uses recurrent neural networks (RNN); more precisely, LSTMs, in a Seq2Seq architecture, to predict the most probable scenarios of crop rotations to be exploited in a field in subsequent growing seasons, according to cropping habits. The output can be used in crop models to build a decision support system for greater sustainability in agricultural production by allowing producers to choose the strategy that offers the best compromise between profitability and environmental impact.

1. Introduction

An increase in global food requirements [1,2] coupled with a decrease in natural resources [3,4] and environmental awareness promote a rethinking of agricultural practices towards sustainable agriculture [5]. In this context, crop rotations are increasingly relevant. It is an ancestral practice [6,7] that favors the regeneration and fertility of the soil, limits soil erosion as well as the proliferation of pests, weeds and diseases [8]. These qualities make crop rotation a pillar of sustainable agriculture [9].

Several studies have focused on the prediction of crop rotation based on agronomic knowledge [10–13] and with a data-driven vision [14–22]. However, to the best of our knowledge, none of these studies have addressed forecasting for different years for which crops could be grown at the field level.

This paper aims to fill this gap by proposing an RNN model to predict the crops that are most likely to be grown in periods n+1 to n+x, in a specific field, considering the past sequences of crops grown in a specific field.

Knowing the most likely sequences of crop rotations considered by a producer could help develop sustainable fertilization plans [23]. Since each crop has specific needs [24], knowledge of the future sequence of crops grown in a field will help determine the current and future fertilizer needs of the field. The Government of Quebec in Canada, wants to rely on agronomic knowledge to meet the challenges of the agricultural sector. It supports producers in the implementation of sustainable practices by accompanying them with certified agronomists [25]. The use of a producer's cropping patterns allows for greater acceptability and support for a relevant discussion by producers and agronomists. This knowledge will reduce the amount of fertilizer and herbicides used. Hence, it will allow producers to choose the strategy that offers the best compromise between mid-term profitability and environmental impact. Those results could then be used in crop models to build a decision support system for greater sustainability in agricultural production.

The article is constructed so that that Section 2 presents the tools used in the methodology. First, preliminary concepts related to recurrent neural networks (RNNs) (2.1) are introduced as well as LSTMs (2.2), the Seq2Seq architecture (2.3) and the Teacher Forcing method (2.4) used to train RNN models. Then, Section 3 will present the proposed methodology developed to predict the most probable scenarios of crop rotations to be exploited in a field in the subsequent growing seasons. Finally, Section 4 will discuss the limitations and future research axes related to the proposed methodology.

* Corresponding author.
E-mail address: ambre-manon.dupuis@polymtl.ca (A. Dupuis).

https://doi.org/10.1016/j.atech.2022.100152
Received 8 December 2022; Accepted 9 December 2022
Available online 12 December 2022
2772-3755/© 2022 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

2. Background

Sequence-based forecasting is a specific supervised learning problem. Sequence-based forecasting places particular importance on the order of observed events. Indeed, sequence processing requires an explicit representation of the order of occurrence of the observations in the period under study. Crop rotation prediction can be considered a Seq2Seq problem, since the objective is to predict the order of crops grown within a field, i.e., the order of elements within a sequence. Sequence prediction is a recurrent problem, and recurrent neural networks (RNN) are a type of neural network specially designed for sequence processing [26] and one of the most widespread solutions for modeling sequential data [27].

2.1. Recurrent neural networks

Sequential data assumes a dependency between the elements of the sequence. Hence, a specific neural network is needed to overcome the independence assumption involved in traditional neural networks [28]. A RNN can be seen as a loop within a "typical" neural architecture (ANN) [26]. This loop (Fig. 1) supports the temporal aspect of the data in a dynamic way, since the value of the previous hidden state is used as input for the current state [29]. It is supported with memory, allowing the neural network to learn from the ordered nature of sequential data.

In contrast to the ANNs, RNNs take into account the temporality of the data and, therefore, the order in which the data was observed. Moreover, RNNs are able to model sequences of inputs and outputs of variable sizes [26]. RNNs are therefore used to address various problems and applications, for example, classification problems in medicine [30, 31], textual analysis [32,33], detection of fraudulent transactions and abnormal behaviors [34,35], as well as for music generation or text generation. Finally, they are used in multi-temporal prediction problems [36,37], translation problems [38,39], or document summarization problems [40].

Despite their great potential, RNNs are not widely used on practical cases in industry, as they are considered difficult to train [41]. The *Vanishing* and *exploding gradient* are two problems frequently encountered when using this type of model [42]. The model is trained by *Backpropagation* as a function of time. The longer the sequence is, the more the gradient will tend to decrease (*Vanishing gradient*) or to increase disproportionately (*Exploding gradient*), making it impossible to update the model weight. To overcome this problem, Hochreiter and Schmidhuber [43] propose a new RNN cell, the *Long Short-Term Memory* (LSTM) whose internal mechanism allows for a better management of the *Backpropagation* during training.

2.2. The LSTM cells

An LSTM neural network is a neural network made of n identical LSTM cells distributed over m layers. An LSTM cell (Fig. 2) consists of a memory (C_t) and three gates: a forget gate (f_t), an input gate (i_t) and an output gate (o_t). This internal architecture allow information to be transmitted from period to period.

The equations governing the operation of an LSTM cell are described below [44].

$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + bf\right) \tag{1}$$

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + bi) \tag{2}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + bo) \tag{3}$$

$$\widehat{c}_t = \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{4}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \widehat{c}_t \tag{5}$$

$$h_t = o_t \odot \tanh(c_t) \tag{6}$$

With:

$$\sigma(x) = \left(\frac{1}{1+\exp(-x)}\right), \textit{called sigmoid function}$$

$$x \odot y = \textit{element} - \textit{wise product}$$

$$x_t \textit{ and } h_t = \textit{input and output vector at time } t$$

$$W_{xi}, W_{hi}, W_{xf}, W_{hf}, W_{xo}, W_{ho}, W_{xc}, W_{hc} = \textit{weight matrix for linear transformations}$$

$$b_i, b_f, b_o, b_c = \textit{bias vector}$$

$$i_t, f_t, o_t = \textit{gate vectors}$$

$$c_t = \textit{memory cell state vector}$$

$$h_{t-1} = \textit{output vector of the previous time step}$$

Eqs. (1) and (2) define which parts of the past state vector (h_{t-1}) and the input vector (x_t) will be used to update the memory cell state (c_t) (Eq. (5)). The output gate (Eq. (3)) is used to determine the part of the updated cell state used to define the output state vector of the LSTM cell (h_t) (Eq. (6)).

This structure within the LSTM cells limits the loss of information over time when training the model by *Backpropagation*. Therefore, this type of RNN will be used in the following methodology (Section 3).

2.3. The Seq2Seq architecture

A sequence-based prediction of a sequence, also called *Seq2Seq Prediction*, is a generalization of the sequence-based prediction principle. Initially proposed to answer the translation problem [45], the RNN autoencoder is an architecture used to answer problems of *Seq2Seq prediction* [44].

As presented in Fig. 3, the auto-encoder is composed of two main parts, namely, the encoder and the decoder.

The encoder consists of an LSTM layer containing $n_{neurons}$ of LSTM type. LSTM cells will be used a memories holding a summary representation of past events in a sequence. This representation, called the

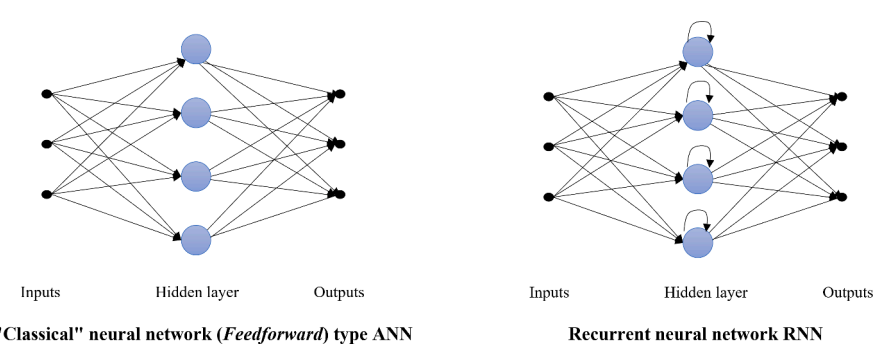

Fig. 1. Schematic comparison of ANN (*Feedforward*) and RNN architectures.

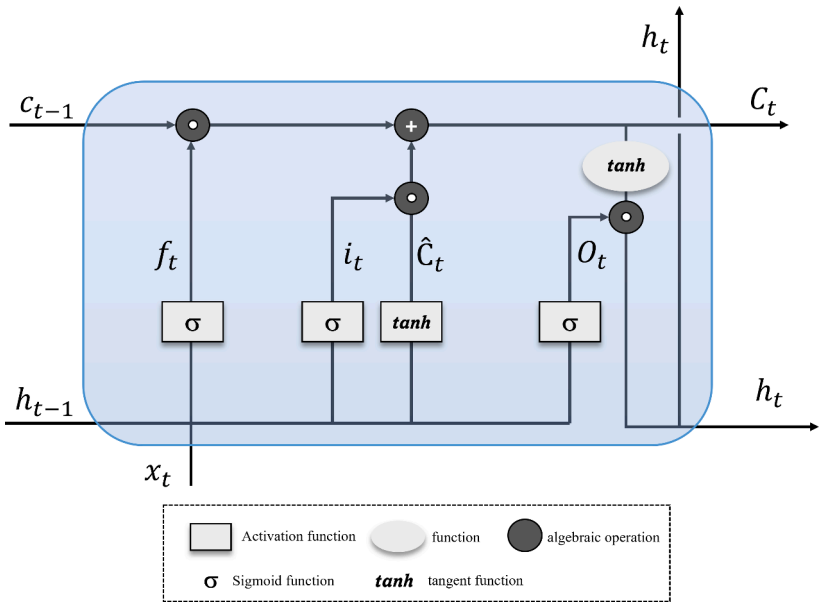

Fig. 2. Diagram of an LSTM cell adapted from Wang et al. [44].

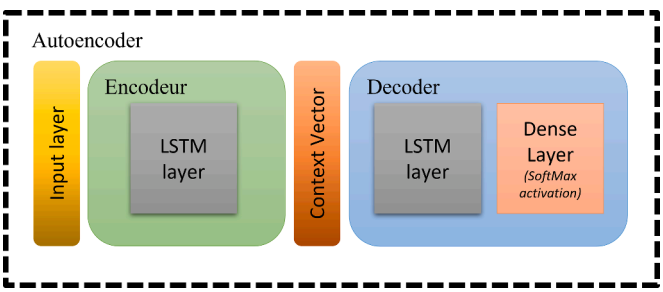

Fig. 3. Autoencoder in a Seq2Seq architecture context.

memory state (C_t) and the output vector state (h_t), evolves with the sequence. Each element that constitutes the sequence (X_t) is associated with the representation (C_{t-1}, h_{t-1}) of the previous element in order to update them (Eqs. (1) to (6)). When all the elements of the sequence have been evaluated, the C_t and h_t states are used as a simplified representation of the complete input sequence. This representation is often referred to as the context vector. This context vector is used as the initial state of the decoder cells.

The decoder uses the same structure as the encoder except the input elements from the sequence (X_t) are replaced by the vector provided for each element of the output sequence (s_t). A dense layer is added after the LSTM layer. This last layer, equipped with an activation function *SoftMax*, allows us to obtain a probability of occurrence matrix (Y_{pred}) of each of the activities for the n_{out} years to be predicted.

These two elements, encoder and decoder, permit sequential data to be processed that have different lengths of input and output [46].

2.4. Teacher forcing training

There are several methods for training Seq2Seq templates. One of the most frequently used is the *Teacher Forcing* [47]. The principle of Teacher Forcing is to correct the model at each step of the prediction in order to limit the propagation of errors in the predicted sequence.

In the context of *Teacher Forcing*, a set *DTI* is generated in order to compare the predictions of the model with the real observed values. The set *DTI* corresponds to the set Y_{train} translated by one increment such that $DTI_t = Y_{train_{t+1}}$. To do so, a code signifying the beginning of the sequence (< *BOS* >=*Begining Of Sentence*) is added to the beginning of the set Y_{train} and the last element of the same set is deleted.

DTI=[<BOS>, Y_{train}, ..., $Y_{train_{t+|n-out|-1}}$]

The correction is made by comparing the predicted value at time t (s_t) with the value actually observed at time t (DTI_{t+1}). This method limits the accumulation of errors in the prediction, thus facilitating the convergence of the model and limiting the training time of the model [48].

3. Methods

The prediction of crops grown at the field level in year n+1 [16,17] as well as the regional prediction of crop trends in year n+x [15] are discussed in the literature. Even though agricultural strategies are elaborated upon in a mid-term horizon, to the best of our knowledge, no methodology has been developed to propose a crop rotation model that is able to predict producer's intention for periods n+1 to n+x at field level. This article intends to fill this gap by proposing a methodology to predict the most probable sequence of crops grown in a field in year n+1 to n+x (Fig. 4). Each step of the methodology is detailed in the following subsections.

In order to present the methodology, a didactical example is proposed. Table 1 represents events data in which a case is a unique realisation of a process by the succession of activities in different time stamps. 6 activities (A, B, C, D, E, F) and two cases (1, 2) are represented. The time stamp indicates the time during which the activity was carried out in the given case. Note that the unit of the time marker (year, quarters, months, etc.) must be consistent with the context of production. In the context of predicting crop rotations, activities (A, B,..., F) can be thought of as crop types while cases can be thought of as the fields in which the crops are grown. For example, a record with case 1 and activity A could represent the corn crop in field 1. Then from Table 1, we may understand that in field 1, activity A occurred during the time periode t1, activity D occurred on during the time periode t2, and so on.

3.1. Step 1: Data preparation

The data preparation in Fig. 4 (step 1) mainly deals with record tracking, missing and duplicate data management, data encoding and formatting.

First, the data has to be transformed in order to obtain the sequences of activities performed within a process over time. The traceability of

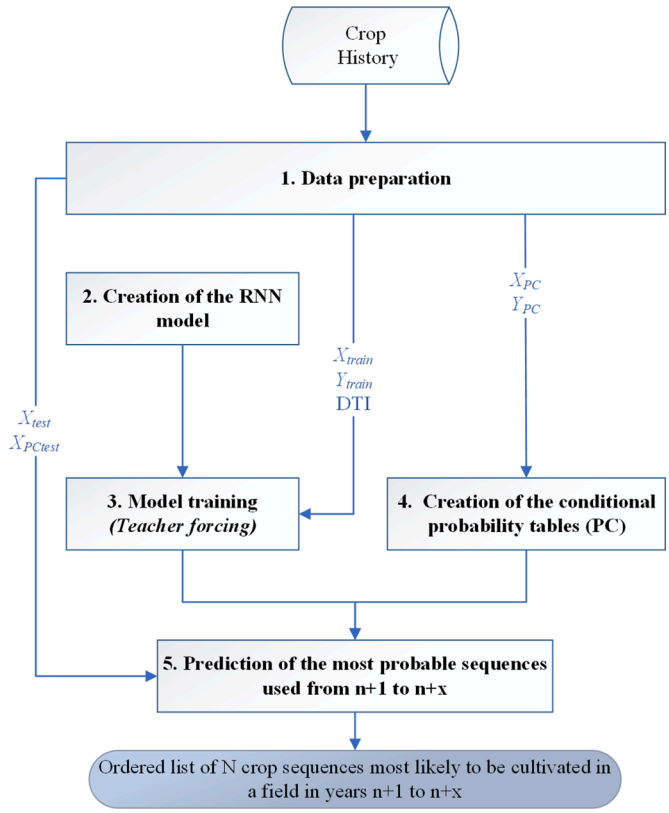

Fig. 4. Proposed methodology.

Table 1
Event data of the didactical example.

Activity	Case	Temporal stamp
A	1	t1
B	2	t1
C	2	t3
D	1	t2
E	1	t2
F	1	t3
F	1	t3

records has to be ensured by defining an attribute that acts as an identification key. An identification key has to, by definition, be unique to the object it identifies. In Table 1, the attribute 'Case' can act as an identification key for each field. Ensuring the traceability of records allows duplicate records within a process to be identified and treated.

As shown in Fig. 5, there are two types of duplicates that have to be treated accordingly. Records in which all information is repeated more than once are called complete duplicates (see red records in Fig. 5). On the contrary, partial duplicates are records that have an identical identification key and time marker but have activities that vary (see yellow records in Fig. 5).

In the first case, the excess records are removed. In the second case, the excess records are transformed into a code that represent the notion of partial duplication. The mention "<DUP>" is used to indicate the presence of partial duplicates. Fig. 5 shows how duplicate records are handled.

This transformation limits data noise and the number of deleted records.

Once traceability is guaranteed and duplicates have been processed, the activity sequences are highlighted using a pivot table. The activities are represented according to the process identification key and the attribute used as a time marker. This operation makes it possible to highlight the missing data within the processes (Fig. 6).

Once the sequences are highlighted, the use of a sliding window, as proposed by Dupuis et al. [17], Zhang et al. [18], allows datasets to be defined, while guaranteeing the preservation of the order within the sequences. The sliding window is defined by 2 dimensions, noted as L and W. $W = |n_{in}| + |n_{out}|$, with $|n_{in}|$ = size of the sequence considered in input, $|n_{out}|$ = size of the sequence to predict, and $L \geq W$

In Fig. 7, the dimension L (in green) corresponds to the history on which a sliding window of size W (in yellow) is used. The dimensions L and W are two hyperparameters defined according to the context.

The value of L is chosen according to the context of the study. In Fig. 7, the value L=7 allows to consider only the events after May 2021. The adjustment of the value of L allows more flexibility to the model. The use of sliding windows permits sequences of size W to be generated. However, these sequences often contain missing data that has to be processed. As previously defined, W can be decomposed into two parts, namely n_{in} (the input sequence used for the prediction) and n_{out} (the sequence to be predicted). This distinction implies two different treatments of missing data.

1. Any record containing at least one missing piece of data in the n_{out} set is removed from the data set. This decision is explained by the use of the data set. The n_{out} data is the data we are trying to predict. However, it seems undesirable to predict missing data.
2. Any record that exceeds the filtering threshold (SF) of missing data in the n_{in} set is removed from the data set. For example, if a sequence length $|n_{in}|$=10 is considered as input and if we set SF=75%, then any sequence containing more than 2 missing data will be removed. This choice is explained by the need for training and robustness of the model. The model has to be relatively robust and accept, to a certain extent, missing data. However, in order to be trained, the model has to be based on information. An empty sequence will only confuse the model.

Afterward, the remaining missing data is encoded with "<PAD>".

Fig. 8 shows the missing data handling process for sequences with SF=50%, $|n_{in}|$=3 and $|n_{out}|$=2. The records colored in red meet the first

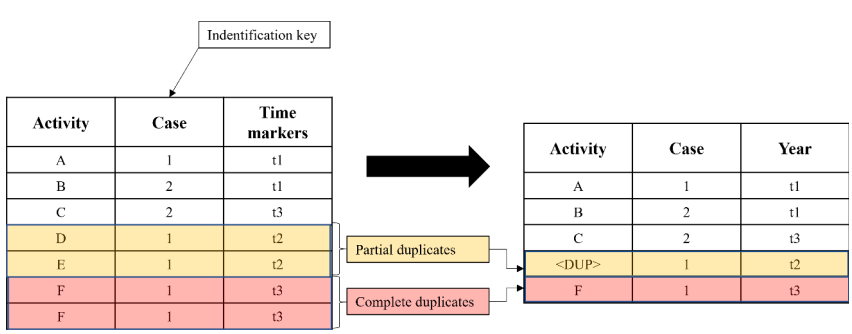

Fig. 5. Treatment of duplicate records according to two predefined rules.

Activity	Case	Time markers
A	1	t1
B	2	t1
C	2	t3
<DUP>	1	t2
F	1	t3

	t1	t2	t3
1	A	<DUP>	F
2	B	NaN	C

Fig. 6. Sequencing of activities for each case according to time markers.

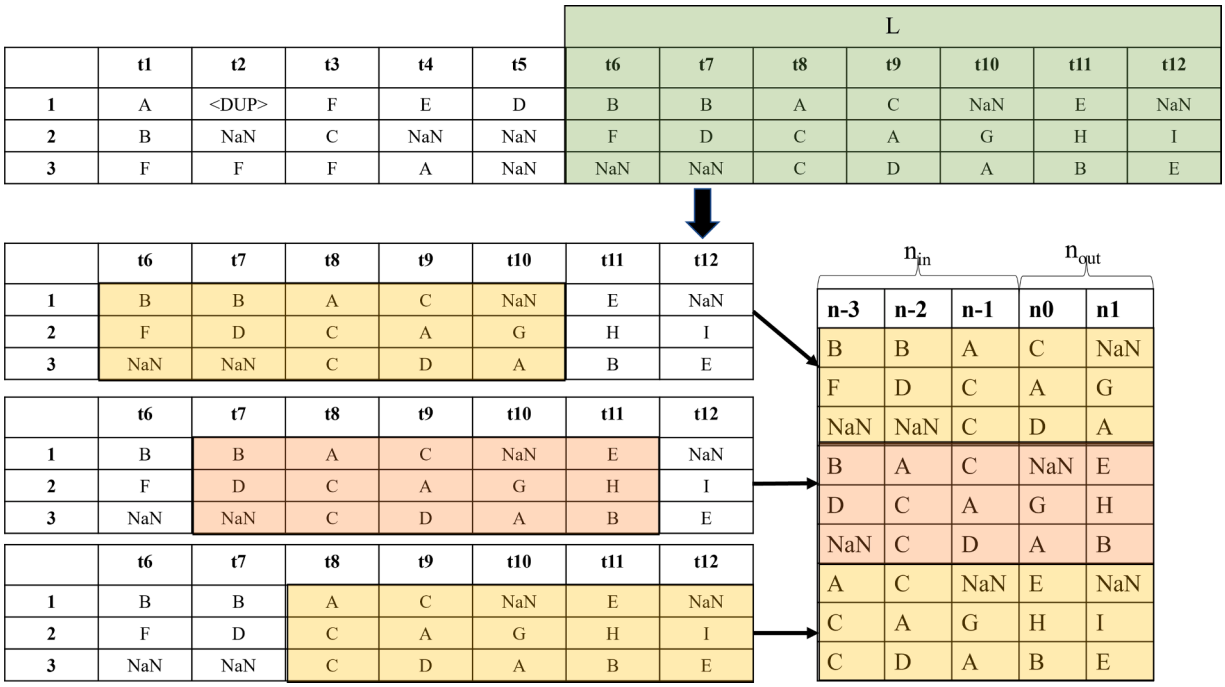

Fig. 7. Sequence generation using sliding window with L=7 and W=5.

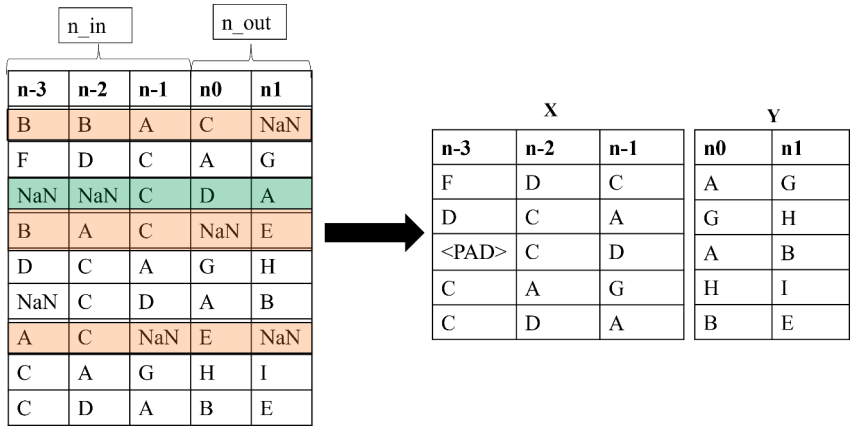

Fig. 8. Process of handling missing data with SF=50% and generating X and y examples.

missing data decision rule, while the record colored in green meets the second missing data decision rule. These records are therefore removed from the dataset. The remaining missing data is encoded with the value "<PAD>".

Thus, the remaining dataset has a fixed size and does not contain any missing data. It can therefore be split into two sets X and Y, as shown in Fig. 8, corresponding to the input sequences (size $nb_{sequences} \times |n_{in}|$) and to the sequences to be forecast (size $nb_{sequences} \times |n_{out}|$).

In order to train and validate the performance of the prediction model, four new data sets are created (X_{PC}, Y_{PC}, $X_{PC_{test}}$, Y_{test}). The data are separated into a training set (75% of the records) and a test set (25% of the records).

The dataset (DTI) is created in order to proceed to the learning by *Teacher Forcing* of the model. As explained in Section 2.4 and shown in Fig. 9, this last set corresponds to the Y_{PC} dataset from which the last value has been removed and "<BOS>" has been added at the beginning of the sequence.

This transformation translates the sequence to be predicted by one

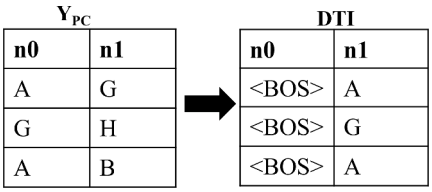

Fig. 9. Generating the DTI set from Y_{PC}.

increment while maintaining the $|n_{out}|$ size of the output vector.

Finally, the X_{train}, Y_{train} and X_{test} datasets are derived from the encoding and formatting of the X_{PC}, Y_{PC}, $X_{PC_{test}}$ datasets so that they can be used as input to RNN models.

To be fed to the recurrent neural network (RNN), the categorical data has to be encoded as numerical ones. The encoding of categorical data can be performed according to the *Label encoding* or *One hot encoding* process. In the former, each unique activity is assigned a unique value. In the latter, each activity is assigned a binary vector the size of the activity dictionary, where a 1 is placed at the location that represents the activity in question. A conversion dictionary, such as the one used in Dupuis et al. [17], has to be created.

The choice of *One hot encoding* allows a matrix M in 3 dimensions to be generated for each dataset: the number of records, the size of the sequence and the number of unique activities. The schematic representation of the data encoding process for RNNs is proposed in Fig. 10.

The encoding and formatting process are also used on the Y_{test} and *DTI* datasets. Thus, he eight data sets (X_{PC}, Y_{PC}, $X_{PC_{test}}$, X_{train}, Y_{train}, X_{test}, Y_{test} and *DTI*) are now created and prepared. Therefore, the creation of the prediction model can begin.

3.2. Step 2: Creation of the RNN model

The data are now ready to be used in the prediction model. In Fig. 4 (step 2) the prediction model is created based on an Seq2Seq architecture using LSTM cells. The Seq2Seq architecture is chosen as it is the architecture dedicated to the task of sequence prediction from sequence input [26]. The LSTM cells are chosen since their internal structure limits the vanishing and exploding gradient problem, hence the training of the network will be eased [43].

As presented in Section 2.1, RNN models using a Seq2Seq architecture for sequence prediction rely on an "encoder-decoder" set linked by a context vector. In the case of a Seq2Seq architecture using LSTM cells, the context vector represents the internal states of the encoder's LSTM cells once the entire input sequence has been evaluated. The creation of the RNN model for Seq2Seq prediction consists of setting the parameters for the layers that it is composed of. As mentioned by Schrimpf et al. [49] "The process of designing neural architectures requires expert knowledge and extensive trial and error". This applied to the definition of Seq2Seq-LSTM hyperparameters.

The first layer of the auto-encoding model is the **input layer**. As the *Teacher Forcing* method will be used to train the model (Section 2.4), the use of two sets of input data, namely X_{train} for encoder training and *DTI* for decoder training, is necessary. Thus, the input layer is composed of two blocks. The size of the input data has to be specified in order to maintain consistency throughout the model. The first block, allowing the integration of the set X_{train}, is sized to process matrices of size $None \times |n_{in}| \times n_{activities}$. While the second one, allowing the integration of the set *DTI*, is sized to handle matrices of size $None \times |n_{out}| \times n_{activities}$. The *None* dimension is used to consider the variability in the first dimension of the data matrix. Indeed, the number of sequences cannot be fixed in each data set since this number changes depending on the example considered.

The internal structure of the **encoder** and its operation have been explained in a general way in Section 2.1. Although the principle of operation of LSTM cells is relatively complex, its implementation is relatively fast. The LSTM layer is constructed by determining the number of neurons ($n_{neurons}$) that it is composed of. This value is a hyperparameter chosen by the user. Furthermore, the LSTM layer has to allow for the extraction of the C_t and h_t states in order to generate the context vector used to link the encoder to the decoder. This is possible thanks to the parameterizable state return feature on the LSTM layers. Finally, it is important to specify the origin of the layer's input data. Thus, the LSTM layer is linked to the first block of the input layer, namely, the one processing the X_{train} dataset. The output of the encoder is the context vector of dimension $None \times n_{neurons}$, a synthetic representation of the n_{in} sequence.

Concerning the **decoder**, the LSTM layer is built using the information from the encoder. The number of neurons present on the LSTM layer of the decoder will be the same as the one defined for the encoder, i.e. $n_{neurons}$. Moreover, the states of the cells composing the LSTM layer of the decoder are initialized using the context vector from the final states of the encoder. This initialization allows the information collected by the encoder to be linked to the decoder. The output of the LSTM layer is used as input to the densely connected layer using the activation function *SoftMax*. This layer is composed of $n_{activities}$ neurons. This last layer proposes the probability vector of the occurrence of activities at a time *t*.

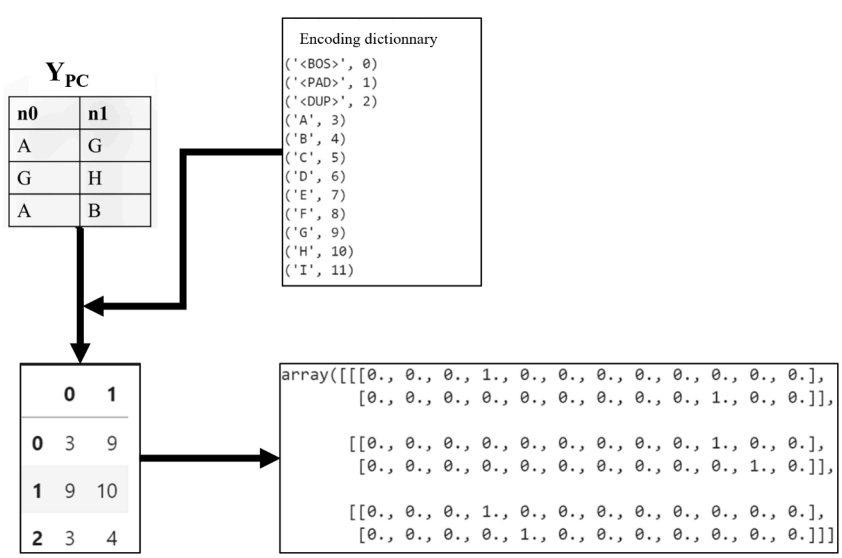

Fig. 10. Data encoding process used on the Y_{PC} dataset for RNNs models.

As presented in Section 2.1, the main difference between the encoder and the decoder is the use of the vector predicted at time t (s_t) as the cell input for the prediction at time $t+1$ (s_{t+1}). This feature allows an input of dimension $None \times n_{neurons}$ to be transformed into an output matrix of size $n_{out} \times n_{neurons}$. This is possible thanks to the parameterizable sequence feedback feature on the LSTM layers.

In order to train the model according to the *Teacher Forcing* method, the *DTI* dataset has to be integrated with the decoder. This integration can be done by linking the second block of the input layer to the decoder.

Fig. 11 represents the architecture of the recurrent neural network obtained as a result of step 2 of the methodology. This neural network has to now be trained in order to predict future sequences according to past sequences.

3.3. Step 3: Model training

Once the model is created, it needs to be trained using the X_{train}, Y_{train}, and *DTI* datasets. The aim of the learning phase is to learn the weights allowing the loss function of the training set to be minimized. To do so, a loss function and an optimizer have to be defined.

During the training, each record of X_{train} is used as input to the model. At each time step, the dense layer with the softmax activation function produces a vector of size $1 \times n_{activities}$ which can be considered as the probability vector of activity occurrence for that specific time step. As explained by Jiang et al. [50], the training aims to maximize the probability of the token at each time step. This can be expressed mathematically by Eq. (7).

$$\max P(Y|X) = \max \prod_{t=1}^{|Y|} P(y_t|y_1 \ldots y_{t-1}, X) \tag{7}$$

Where $y_1 \ldots y_{t-1}$ *are previously generated tokens*

This maximization can also be viewed as minimizing the loss between the prediction and the ground truth. To calculate this loss in a Seq2Seq RNN context, the cross-entropy loss function is generally used [50,51] since it is used for multiclass prediction problems. The cross-entropy equation is proposed in Eq. (8) [50].

$$Cross\ Entropy(y_t) = -\sum_{i=1}^{n_{activities}} \delta_i(y_t) \log(P(c_i|y_1 \ldots y_{t-1}, X) \tag{8}$$

Where :

$$\{c_1, \ldots, c_N\} \ : \ Search\ space\ in\ y_t$$

$$\delta_i(y_t) = \begin{cases} 1\ if\ y_t = c_i \\ Else\ 0 \end{cases}$$

$$P(c_i|.) = \ Probability\ of\ the\ candidate\ token\ c_i\ calculated\ by\ the\ softmax\ function$$

Therefore, once the n_{out} probability vectors have been generated, the $\hat{y}$ matrix of size $n_{out} \times n_{activities}$ is compared to the ground truth of the Y_{train} dataset using the cross entropy loss (CE) function. It is important to note that in order to use the cross-entropy loss function, the Y_{train} dataset has to first be encoded using the one hot encoding technique.

As shown in Eqs. (7) and (8), the RNN model uses the prediction of the previous time steps as input for the prediction of the next time step. At the beginning of the learning process, the model is not able to provide consistent predictions. This leads to error propagation and complicates the convergence of the model, which eventually leads to slow learning. The method *Teacher Forcing* (Section 2.4) is used to address this problem. It helps the model learning process by using the ground truth of the previous time step as input for the prediction in the training phase. Thus, during the training phase, the *DTI* dataset as well as the X_{train} dataset are used as the input to the model, while the prediction obtained is compared to the Y_{train} dataset using the CE loss function. The loss function is combined with an optimizer to update the weights. For this purpose, the Adam optimizer based on the stochastic gradient is used [52]. The associated learning rate is a hyper parameter.

In addition, the *Early Stopping* method is used to avoid overfitting by

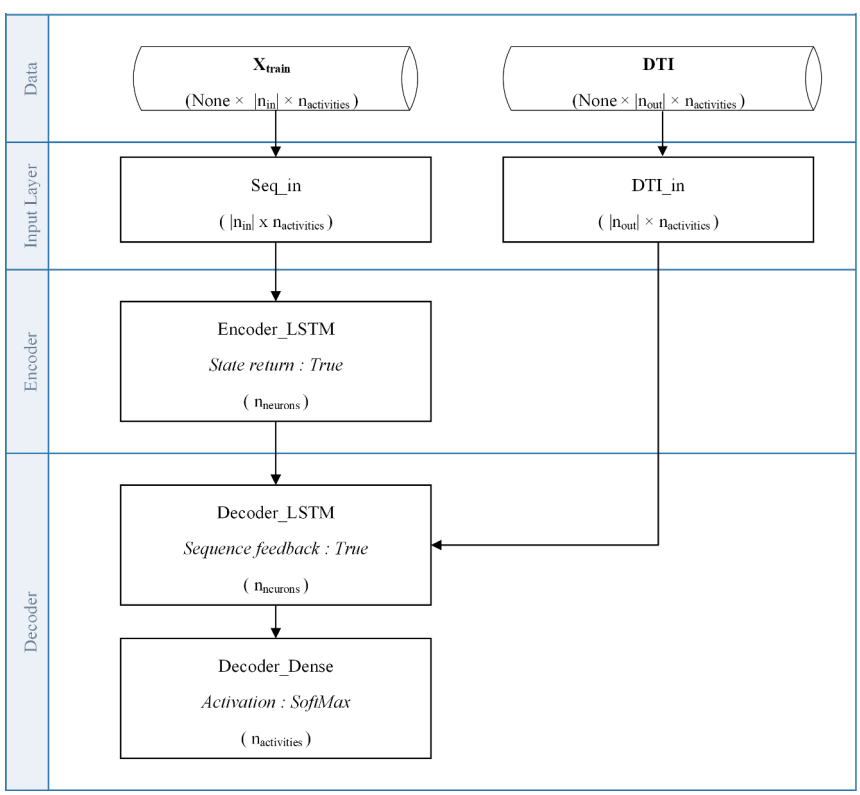

Fig. 11. Architecture of the RNN.

stopping the learning based on the monitoring of the model performance. 25% of the training set is used as a validation set to monitor the learning progress and to define a stopping criterion. Here, the stopping criterion is the accuracy on the validation set and is calculated at each epoch, i.e. at each exhaustion of the training inputs [53]. Learning is stopped when the accuracy of the validation set does not improve over a certain number of iterations, called patience. This avoids the stagnation of a local optimum. Using the *Early stopping* method determines the number of epochs required to train the model and thus eliminates a hyper parameter that needs to be set manually.

The output of the trained model is a $\widehat{Y}$ matrix of size ($n_{out} \times n_{activities}$) composed of estimates of the probability of occurrence of each activity at each time step. This matrix will be used in the prediction phase to propose the most likely observed scenarios from the RNN model using an heuristic algorithm called *BeamSearch*. This algorithm explores the graph of event sequences in a more restricted way.

3.4. Step 4: Creation of the conditional probability tables

In the case of sequence prediction from sequences, the present and future states of the system can be defined as the succession of $|n_{in}|$ activities of the considered history and $|n_{out}|$ activities to be predicted, respectively. Thus X_{PC} and Y_{PC} are used to create $|n_{in}|$ conditional probability matrices PC such that:

$$PC_{z_{(|i|\times|j|)}} = \begin{bmatrix} w_{11} & \cdots & w_{1j} \\ \vdots & \ddots & \vdots \\ w_{i1} & \cdots & w_{ij} \end{bmatrix} \tag{9}$$

Where :

$$w_{ij} = P(j|i)$$
$$\sum_{j=1}^{|j|} w_{ij} = 1 \ \forall i$$

With :

$$\begin{aligned} i &= \textit{Unique sequence of size } |n_{in}| \textit{ in } X_{PC_z} \\ j &= \textit{Unique sequence of size } |n_{out}| \textit{ in } Y_{PC_z} \\ z &\in [1; |n_{in}|] \end{aligned}$$

The objective is to determine from a history already observed in the data set the probability that an activity (or sequence of activities) will occur. The larger the size of the considered history $|n_{in}|$, the more accurate the prediction will be. But, the larger $|n_{in}|$ is, the lower the probability of having already observed a specific sequence in the training set and thus the less robust the model will be. Indeed, if a sequence is never observed in the training set but appears in the test set, the conditioned probability tables are unusable.

$|n_{in}|$ conditioned probability tables of size i, j are created with $i = |n_{out}|$ and $j \in [|n_{in}|, |n_{in}| - 1, \ldots, 1]$. The conditioned probability model used corresponds to a set of $|n_{in}|$ tables such that:

$$PC = \{PC_{|n_{in}|}; PC_{|n_{in}|-1}; \ldots; PC_1\}$$

Fig. 12 represents the generation of conditioned probability tables considering the sets X_{PC} and Y_{PC} as training data.

Based on the example proposed in Fig. 12, if a sequence $[C, Z, A]$ appears during the test of the model, then the model will predict - with a probability of 50% - the appearance of sequences $[G, H]$ and $[B, E]$. These values are obtained using the PC_1 table since the sequences $[C, Z, A]$ (in table PC_3) and $[Z, A]$ (in table PC_4) have never been encountered in the proposed training sets. It is the same for the other combinations.

The PC tables are coupled with the RNN model's prediction in the prediction step in order to improve its performance.

3.5. Step 5: Prediction of the most probable sequences used from n+1 to n+x

The RNN model generates crop sequences from a context vector and a set of possibly feasible activities. Although the RNN model relies on historical data for its training, it is possible that the predicted sequences are a combination of activities that are not present in the training set. Assuming that a sequence already observed in the data set is more plausible than the appearance of a new sequence for the same given context, predictions proposing a new combination of activities for a given context should be penalized without being removed from the set of possible sequences.

The confrontation of the results obtained with the RNN model with those obtained with the PCs allows plausible results to be proposed, while remaining open to the discovery of new sequences. The forecasting step has three phases: (1) the forecast by inference from the RNN model, (2) the forecast from the conditioned probability (CP) model, and (3) the integration of the results of the two models (Fig. 13).

Forecast by inference from the RNN model.

The *BeamSearch* algorithm is a heuristic that explores the graph of event succession possibilities in a restricted way. Indeed, when it explores the graph, the algorithm considers only a restricted set of child nodes, thus maximizing the global probability of realizing a sequence. The number of options considered, called *Beam Width*, is a user-defined hyperparameter. The use of the *Teacher Forcing* method is frequently associated with the *BeamSearch* [47]. Park et al. [36] uses this association to define the trajectories most likely to be taken by drivers. This algorithm is used on the RNN models in order to propose the *Beam Width* most probable sequences provided by the RNN model.

To do so, the prediction by inference from the RNN model consists of

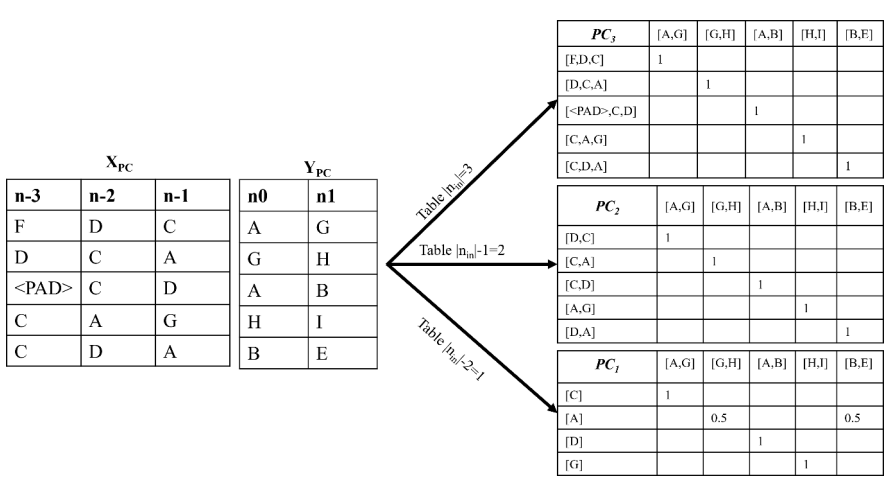

Fig. 12. Creation of $|n_{in}|$ conditional probability tables based on the sets X_{PC} and Y_{PC}.

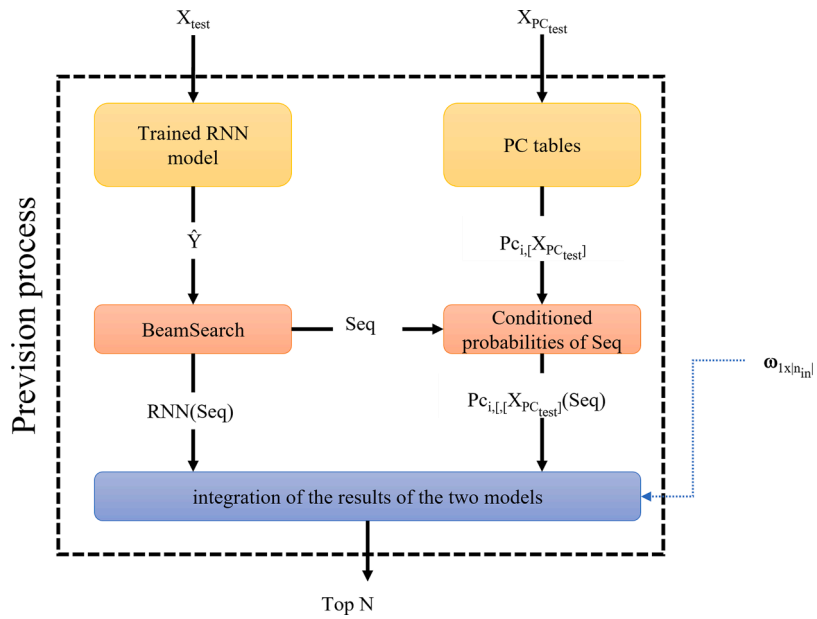

Fig. 13. Detailed forecasting process.

using the RNN model previously trained on the X_{test} dataset. The X_{test} data are encoded into a context vector by the model encoder. The decoding phase uses only the predicted values at time t to predict the value at time $t+1$ since the actual observed values (set *DTI*) are not available in the test phase. To predict the first item in the n_{out} sequence, the context vector is associated with an empty vector of dimension $n_{activities}$. The decoder outputs a probability vector of the culture exploited at time $t+1$. This probability vector is then associated with the context vector and used in the decoder to output the probability vector of the culture exploited at time $t+2$. This process continues until the n_{out} probability vectors are obtained. The matrix of predicted sequences of size $n_{out} \times n_{activities}$ is considered as the output of the model. It is on this matrix that the BeamSearch algorithm is applied. As mentioned above, the BeamSearch algorithm aims to maximize the overall probability of a sequence being completed. For this purpose, each combination of activity, call **Seq**, is scored using the log-likelihood function as defined in Eq. (10).

$$RNN(Seq) = \sum_{x_i \in Seq} \log(\hat{Y}(x_i)) \tag{10}$$

This score is called the $RNN(Seq)$ and it provides an order to the sequences. The *BeamWidth* most probable sequences **Seq** are those with the maximum $RNN(seq)$. As a reminder, the *BeamWidth* is an hyper-parameter defining the number of options considered. Those *BeamWidth* sequences and their respective $RNN(Seq)$ scores are then extracted to be integrated with the CP model results.

Fig. 14 illustrates the forecast by inference from the RNN model with the input sequence [DCA] and $BeamWidth = 3$.

First, the [DCA] sequence is encoded as a One-Hot matrix, which is then fed to the trained RNN model. As a result, the matrix is produced

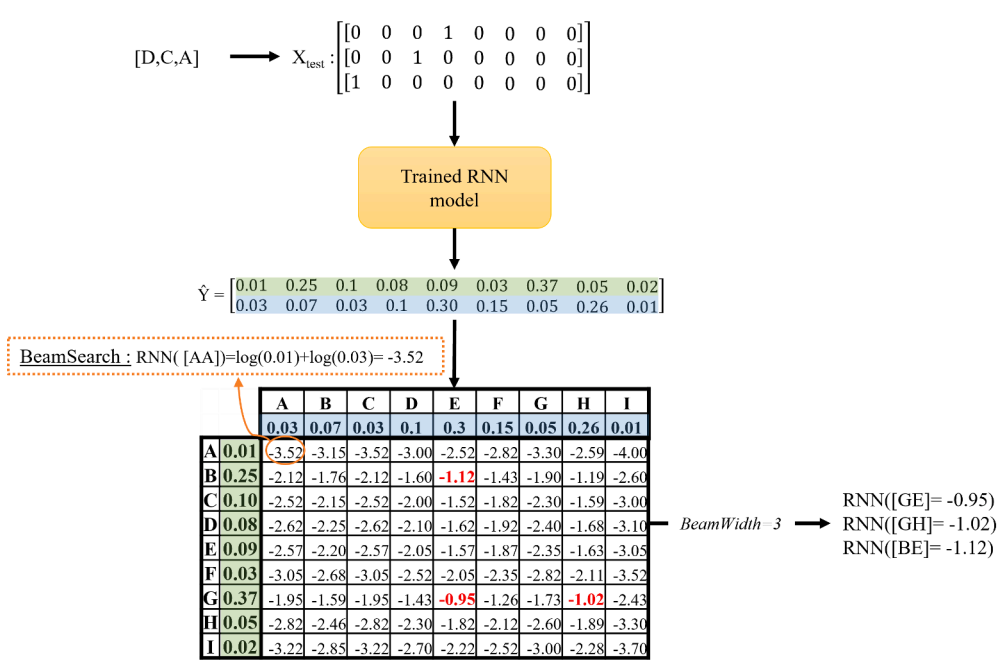

Fig. 14. Example of RNN prediction process with BeamSearch for the input sequence [D,C,A].

and the log-likelihood of each output sequence is calculated. The sequences [GE], [GH] and [GE] are the 3 sequences with the highest RNN (Seq). Therefore, they are extracted to be used in the prediction of the conditioned probability (CP) model and the calculation of the overall score.

Forecast from the conditioned probability (CP) model.

The prediction from the PC model uses the $X_{PC_{test}}$ dataset with the PC tables to obtain the probability vector of occurrence of each sequence to be predicted. For a given historical sequence present in the $X_{PC_{test}}$ dataset, each possible output sequence and their associated probabilities of occurrence are stored in a dictionary called $PC_i[X_{PC_{test}}]$ where $i \in [1, |n_{in}|]$. Fig. 15 represents the generation of the $|n_{in}| = 3$ dictionaries of probability, given the input sequence [DCA] and the PC tables presented in Fig. 12.

Then, the sequences **Seq** proposed by the RNN model are used to extract the pertinent probability of occurrence $PC_i[X_{PC_{test}}](Seq)$. If the sequence *Seq* is not represented in the dictionaries, the associated probabilities are set to 0. Then, the logarithmic function is applied to the $PC_i[X_{PC_{test}}](Seq)$ probabilities in order to be integrated with the $RNN(Seq)$ results when calculating the overall score of the *Seq* sequence. Since the logarithm of the zero value is $-\infty$, a substitute value, called k, is use as a penalty value for improbable sequences.

Given the results presented in Figs. 14 and 15, the sequences [GE], [BE] and [GH] should be evaluated. If the penalty value is $k = -10$, then $PC_{3,[DCA]}([GE]) = PC_{2,[CA]}([GE]) = PC_{1,[A]}([GE]) = -10$ since [GE] does not appear in any of the outputs of the PC tables. The same can be inferred for the $PC_{3,[DCA]}([BE])$ and $PC_{2,[CA]}([BE])$ since [BE] have never been seen following sequence [DCA] or [CA] in the training dataset. On the other hand, $PC_{3,[DCA]}([GH])$, $PC_{2,[CA]}([GH])$, $PC_{1,[A]}([GH])$ and $PC_{1,[A]}([BE])$ have specific values since they appear in the PC dictionaries. More precisely, $PC_{3,[DCA]}([GH]) = PC_{2,[CA]}([GH]) = log(1) = 0$ and $PC_{1,[A]}([GH]) = PC_{1,[A]}([BE]) = log(0.5) = -0.3$.

Those results are then used in the last step of the prediction phase, the computation of the general score of each sequence by the integration of the results of the two models.

Integration of the results of the two models.

The integration of the results of the two models is done with a linear function (Eq. (11)).

$$\begin{aligned} Score(Seq) &= RNN(Seq) + \omega_{|n_{in}|} PC_{|n_{in}|}(Seq) + \omega_{|n_{in}-1|} PC_{|n_{in}|-1}(Seq) + \ldots \\ &\quad + \omega_1 PC_1(Seq) \\ &= RNN(Seq) + \sum_{k=1}^{|n_{in}|} \omega_k PC_k(Seq) \end{aligned} \tag{11}$$

With ω_k the coefficient of importance associated to the PC_k table.

For a historical sequence H of given size $|n_{in}|$, the RNN model proposes *BeamWidth* possible sequences (*Seq*) with the associated score RNN(*Seq*). For each of these sequences, the score RNN(*Seq*) proposed by the RNN model is summed to each $PC_i(Seq)$ weighted by its corresponding ω_i from the ω vector. Eq. (11) allows the predicted sequences already observed in the dataset to be valued, without eliminating the new sequences produced by the RNN model.

The use of a weighted sum allows the importance given to each of the conditioned probability tables to be adjusted. As seen previously, the longer the history considered, the greater the accuracy of the forecast. Thus, the weights ω associated with the tables that consider the longest history should be the most important.

Once the set of predicted sequences has been evaluated, the sequences are ranked in descending order of *Score*. The sequences with the highest score are the sequences with the highest probability of occurrence based on the historical sequence H provided and the RNN and PC models.

Fig. 16 summarizes the example of the prediction process with H= [D,C,A], $\omega = [0.1, 0.05, 0.01]$ using the PC tables presented in Fig. 12.

The output of the prediction is an ordered list of size N of the sequences most likely to be observed following sequence H. The closer the $Score(Seq)$ is to 0, the higher the certainty of the model's prediction that can be considered. Evaluation of the performance and applicability of the methodology should be undertaken using a real case study as proposed in Dupuis et al. [54].

4. Discussions

The proposed methodology predicts the most probable crops to be grown in a field from year n+1 to year n+x, taking into consideration the producer's habits. Not only is one probable crop rotation proposed, but an ordered list of crop rotations is also provided, with each list having a certain probability of occurrence. This gives options to a farmer who can choose the list of crop rotations he/she prefers, taking into

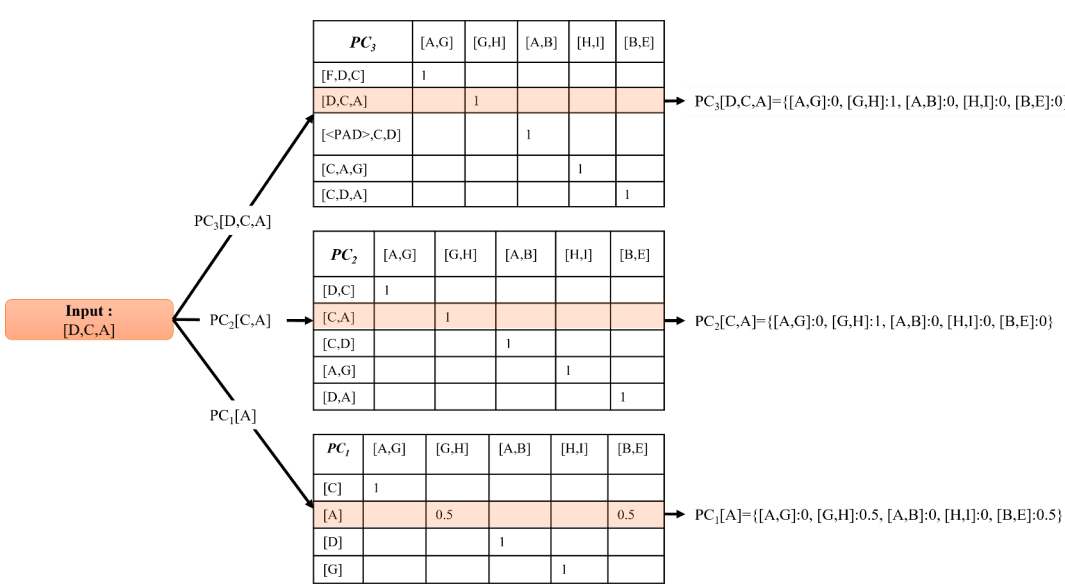

Fig. 15. Example of a PC dictionary generation for the input sequence [D,C,A].

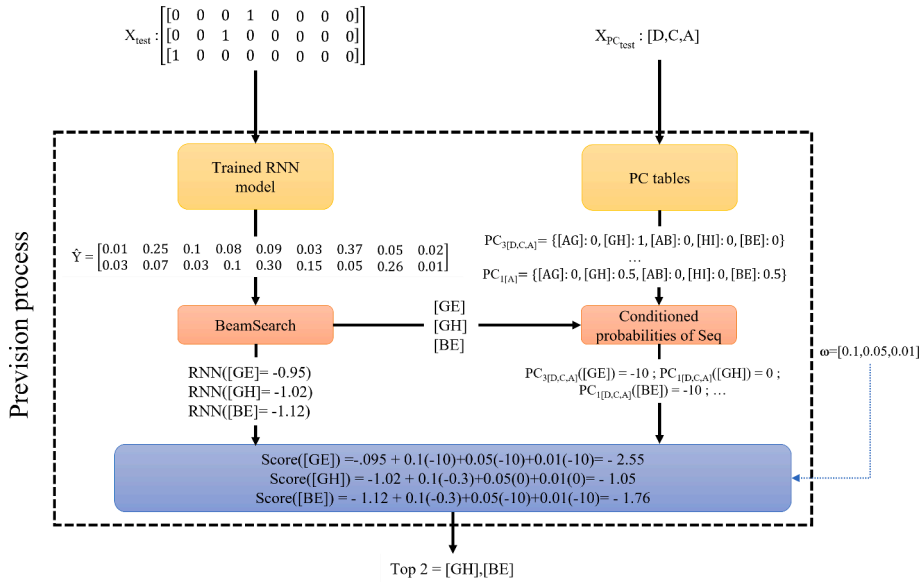

Fig. 16. Example of the forecasting process with the historical sequence [D,C,A].

consideration implications on future possible crops to grow.

Based on LSTMs neural networks (RNN), the Seq2Seq-LSTM model proposed remains to be explored. The data preparation may generate important losses of information that would be of interest to quantify, and the treatment of missing data should be studied in more detail. It would also be relevant to evaluate the distribution of sequences excluded due to missing data to ensure that no exploitation pattern is excluded from the dataset. This would be equivalent to testing the randomness of missing data in the dataset. The hypothesis of a random distribution of missing data supported in this study has to be validated.

The optimization of hyperparameters and the choice of training methods should be further investigated in order to improve the performance of the model. The actual trial-and-error tuning of hyperparameters is a tedious task that does not ensure the optimality or near-optimality of the result. Defining a standardized tuning methodology for the Seq2Seq-LSTM architecture should be considered as an excellent research opportunity [49] and could positively influence the results obtained in this project.

During the training phase, the Cross-Entropy (CE) loss function is used along with the *Teacher Forcing* method. This choice leads to a well-known bias during training called *Exposure Bias* [47,51]. Indeed, the systematic use of the observed data (DTI_t) during the training of the model can lead to an accumulation of prediction errors during the test phase since the model will never have been confronted with its own prediction errors [47]. This is why other methods such as *Professor Learning* or *scheduled sampling* [47,51] have emerged. It will be of interest to test those other kinds of teaching methods to compare and improve the performance of the prediction model. Also, using the mixed cross-entropy loss function (mixed CE) defined by Li and Lu [51] instead of the general CE loss function would relax the mapping process to a one-to-many mapping process instead of one-to-one, as CE does. These changes would probably have a beneficial effect on the performance of the model, improving the training of the RNN and thus its rate of good predictions.

Finally, the quality of the database used limits the quality of the forecasts proposed by the model. Indeed, the structure and management of the database are responsible for the traceability of crop histories, particularly in the case of split fields. The past history of the split field must be accessible since it contains relevant information about cropping sequences which should not be neglected. Evaluation of the performance and applicability of the methodology should be undertaken using a real case study as proposed in Dupuis et al. [54].

5. Conclusion

In a context of sustainable agriculture, the forecasting of future crops in the medium term is necessary for the planning and development of coherent fertilization plans, and to limit the environmental impact of agricultural activities. This study focuses on predicting the intentions of agricultural producers in the medium term with regards to a crop rotation history in a field.

It is based on the use of recurrent neural networks (RNN); more specifically, LSTMs and the Seq2Seq architecture specialized in the prediction of sequences from sequences. The Seq2Seq-LSTM model is associated with a conditioned probability (CP) model to refine the prediction results by prioritizing sequences already observed in the training set.

This methodology is a step towards more sustainable agriculture, since it will allow the development of medium-term sustainable fertilization plans that meet the needs of crops and limit the environmental impact of agricultural activities. Each crop rotation scenario considered in the medium to long-term could be evaluated according to the economic and environmental criteria considered by the producer via a growth model such as APSIM [55]. This knowledge will reduce the amount of fertilizer and herbicides used. Hence, it will allow producers to choose the strategy that offers the best compromise between mid-term profitability and environmental impact for farmers and producers.

Research axes such as the optimization of hyperparameters and the use of different training methods should be explored. An evaluation of the performance of the tool resulting from the methodology is proposed by Dupuis et al. [54].

Declaration of competing interests

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

Data availability

The data that has been used is confidential.

Acknowledgments

We would like to thank our industrial partner for their collaboration in the project as well as CRSNG and FRQNT for their financial support.

References

[1] J. Pretty, W. Sutherland, J. Ashby, et al., The top 100 questions of importance to the future of global agriculture, Int. J. Agric.Sustain. 8 (4) (2010) 219–236, https://doi.org/10.3763/ijas.2010.0534.
[2] D. Tilman, C. Balzer, J. Hill, B. Befort, Global food demand and the sustainable intensification of agriculture, Proc. Natl. Acad. Sci. 108 (50) (2011) 20260–20264, https://doi.org/10.1073/pnas.1116437108.https://www.pnas.org/content/108/50/20260.full.pdf
[3] FAO, Competition for Natural resources, FAO, Rome, Italy, 2017, p. 32. http://www.fao.org/3/i6583e/i6583e.pdf.
[4] H. Godfray, J. Beddington, et al., Food security: the challenge of feeding 9 billion people, Science 327 (5967) (2010) 812–818, https://doi.org/10.1126/science.1185383.https://science.sciencemag.org/content/327/5967/812. https://science.sciencemag.org/content/327/5967/812.full.pdf
[5] Y. Ma, Seed coating with beneficial microorganisms for precision agriculture, Biotechnol. Adv. 37 (7) (2019) 107423.https://www.sciencedirect.com/science/article/pii/S0734975019301235.
[6] V. Papanastasis, M. Arianoutsou, K. Papanatasis, Environnemental concervation in classical greece, J. Biol. Res. Thessaloniki 14 (2010) 123–135.
[7] M. Tariq, H. Ali, N. Hussain, W. Nasim, M. Mubeen, S. Ahmad, M. Hasanuzzaman, Fundamentals of crop rotation in agronomic management, in: M. Hasanuzzaman (Ed.), Agronomic Crops Vol. 1, Springer Singapore, Singapore, 2019, pp. 545–559, https://doi.org/10.1007/978-981-32-9151-5_24.Production Technologies
[8] C. Sheaffer, K. Moncada, Introduction to Agronomy: Food, Crops and Environment, second ed., Cengage Learning, 2011, pp. 340–367.
[9] Z. Cui, Y. Liu, Z. Huang, H. He, G. Wu, Potential of artificial grasslands in crop rotation for improving farmland soil quality, Land Degradation Dev. 30 (18) (2019) 2187–2196, https://doi.org/10.1002/ldr.3415.https://onlinelibrary.wiley.com/doi/abs/10.1002/ldr.3415.
[10] W. Haneveld, A. Stegeman, Crop succession requirements in agricultural production planning, Eur. J. Oper. Res. 166 (2) (2005) 406–429, https://doi.org/10.1016/j.ejor.2004.03.009.https://www.sciencedirect.com/science/article/pii/S0377221704002358.
[11] N. Detlefsen, A. Jensen, Modelling optimal crop sequences using network flows, Agric. Syst. 94 (2) (2007) 566–572, https://doi.org/10.1016/j.agsy.2007.02.002. https://www.sciencedirect.com/science/article/pii/S0308521X07000108.
[12] S. Dogliotti, W. Rossing, M. van Ittersum, ROTAT, a tool for systematically generating crop rotations, Eur. J. Agron. 19 (2) (2003) 239–250, https://doi.org/10.1016/S1161-0301(02)00047-3.https://www.sciencedirect.com/science/article/pii/S1161030102000473.
[13] J. Bachinger, P. Zander, Rotor, a tool for generating and evaluating crop rotations for organic farming systems, Eur. J. Agron. 26 (2) (2007) 130–143, https://doi.org/10.1016/j.eja.2006.09.002.https://www.sciencedirect.com/science/article/pii/S1161030106001249.
[14] J. Aurbacher, S. Dabbert, Generating crop sequences in land-use models using maximum entropy and Markov chains, Agric. Syst 104 (6) (2011) 470–479, https://doi.org/10.1016/j.agsy.2011.03.004.https://www.sciencedirect.com/science/article/pii/S0308521X11000424.
[15] F. Le Ber, M. Benot, C. Schott, J.-F. Mari, C. Mignolet, Studying crop sequences with CarrotAge, a HMM-based data mining software, Ecol. Modell. 191 (1) (2006) 170–185, https://doi.org/10.1016/j.ecolmodel.2005.08.031.https://www.sciencedirect.com/science/article/pii/S0304380005003844.
[16] J. Osman, J. Inglada, J. Dejoux, Assessment of a Markov logic model of crop rotations for early crop mapping, Comput. Electron. Agric. 113 (2015) 234–243, https://doi.org/10.1016/j.compag.2015.02.015.https://www.sciencedirect.com/science/article/pii/S0168169915000575.
[17] A. Dupuis, C. Dadouchi, B. Agard, Predict crop rotations using process mining techniques and Markov principals, Comput. Electron. Agric. 194 (2022) 106686, https://doi.org/10.1016/j.compag.2022.106686.
[18] C. Zhang, L. Di, L. Lin, L. Guo, Machine-learned prediction of annual crop planting in the u.s. corn belt based on historical crop planting maps, Comput. Electron. Agric. 166 (2019) 104989, https://doi.org/10.1016/j.compag.2019.104989. https://www.sciencedirect.com/science/article/pii/S0168169919309482.
[19] R. Yaramasu, V. Bandaru, K. Pnvr, Pre-season crop type mapping using deep neural networks, Comput. Electron. Agric. 176 (2020) 105664, https://doi.org/10.1016/j.compag.2020.105664.https://www.sciencedirect.com/science/article/pii/S0168169920307742.
[20] N. Kussul, M. Lavreniuk, S. Skakun, A. Shelestov, Deep learning classification of land cover and crop types using remote sensing data, IEEE Geosci. Remote Sens. Lett. 14 (5) (2017).
[21] S. Stein, H.-H. Steinmann, J. Isselstein, Linking arable crop occurrence with site conditions by the use of highly resolved spatial data, Land 8 (4) (2019), https://doi.org/10.3390/land8040065.https://www.mdpi.com/2073-445X/8/4/65.
[22] S. Stein, H.-H. Steinmann, Identifying crop rotation practice by the typification of crop sequence patterns for arable farming systems a case study from central europe, Eur. J. Agron. 92 (2018) 30–40, https://doi.org/10.1016/j.eja.2017.09.010.https://www.sciencedirect.com/science/article/pii/S1161030117301405.
[23] CRAAQ, Guide de reference en fertilisation, 2 actualise edition, Centre de rfrence en agriculture et agroalimentaire du Qubec, 2015.
[24] D. Reeves, Principales of crop rotation. Crops Residue Management, first ed., CRC press, 1994, p. 136, https://doi.org/10.1201/9781351071246.
[25] MAPAQ, Plan Strategique 2019–2023, Gouvernement du Qubec, 2020, p. 24.https://cdn-contenu.quebec.ca/cdn-contenu/adm/min/agriculture-pecheries-alimentation/publications-adm/plan-strategique/PL_plan-strategique2019-2023_MAPAQ.pdf?1611950991.
[26] J. Brownlee, Long Short-term Memory Networks with Python: Develop Sequence Prediction Models with Deep Learning, v1.5 ed., Jason Brownlee, 2017.https://books.google.fr/books?id=ONpdswEACAAJ.
[27] F. Hongxiao, T. Fengyun, Bidirectional grid long short-term memory (BiGridLSTM): a method to address context-sensitivity and vanishing gradient, Algorithms 11 (11) (2018) 172, https://doi.org/10.3390/a11110172.https://www.mdpi.com/1999-4893/11/11/172.
[28] M. Nabipour, P. Nayyeri, H. Jabani, S. Shahab, A. Mosavi, Predicting stock market trends using machine learning and deep learning algorithms via continuous and binary data; a comparative analysis, IEEE Access 8 (2020) 150199–150212, https://doi.org/10.1109/ACCESS.2020.3015966.
[29] A. Ozcanli, F. Yaprakdal, M. Baysal, Deep learning methods and applications for electrical power systems: a comprehensive review, Int. J. Energy Res. 44 (9) (2020) 7136–7157, https://doi.org/10.1002/er.5331.https://onlinelibrary.wiley.com/doi/abs/10.1002/er.5331. https://onlinelibrary.wiley.com/doi/pdf/10.1002/er.5331
[30] Z. Nazi, A. Biswas, M. Rayhan, T. Azad Abir, Classification of ECG signals by dot residual LSTM network with data augmentation for anomaly detection. 2019 22nd International Conference on Computer and Information Technology (ICCIT), Dacca, Bangladesh, 2019, pp. 1–5, https://doi.org/10.1109/ICCIT48885.2019.9038287.
[31] X. Hu, S. Yuan, F. Xu, Y. Leng, K. Yuan, Q. Yuan, Scalp EEG classification using deep Bi-LSTM network for seizure detection, Comput. Biol. Med. 124 (2020) 103919, https://doi.org/10.1016/j.compbiomed.2020.103919.https://www.sciencedirect.com/science/article/pii/S0010482520302614.
[32] D. Li, J. Qian, Text sentiment analysis based on long short-term memory. 2016 First IEEE International Conference on Computer Communication and the Internet (ICCCI), 2016, pp. 471–475, https://doi.org/10.1109/CCI.2016.7778967.Wuhan, Chine
[33] F. Long, K. Zhou, W. Ou, Sentiment analysis of text based on bidirectional LSTM with multi-head attention, IEEE Access 7 (2019) 141960–141969, https://doi.org/10.1109/ACCESS.2019.2942614.
[34] J. Jurgovsky, M. Granitzer, K. Ziegler, S. Calabretto, P. Portier, L. He-Guelton, O. Caelen, Sequence classification for credit-card fraud detection, Expert Syst. Appl. 100 (2018) 234–245, https://doi.org/10.1016/j.eswa.2018.01.037.https://www.sciencedirect.com/science/article/pii/S0957417418300435.
[35] M. Wang, T. Lin, K. Jhan, S. Wu, Abnormal event detection, identification and isolation in nuclear power plants using LSTM networks, Prog. Nucl. Energy 140 (2021) 103928, https://doi.org/10.1016/j.pnucene.2021.103928.https://www.sciencedirect.com/science/article/pii/S0149197021002882.
[36] S. Park, B. Kim, C. Kang, C. Chung, J. Choi, Sequence-to-sequence prediction of vehicle trajectory via LSTM encoder-decoder architecture. 2018 IEEE Intelligent Vehicles Symposium (IV), 2018, pp. 1672–1678, https://doi.org/10.1109/IVS.2018.8500658.
[37] K. Arunkumar, D. Kalaga, C. Kumar, M. Kawaji, T. Brenza, Forecasting of COVID-19 using deep layer recurrent neural networks (RNNs) with gated recurrent units (GRUs) and long short-term memory (LSTM) cells, Chaos Solitons Fractals 146 (2021) 110861, https://doi.org/10.1016/j.chaos.2021.110861.https://www.sciencedirect.com/science/article/pii/S0960077921002149.
[38] I. Sutskever, O. Vinyals, Q. Le, Sequence to sequence learning with neural networks. Advances in Neural Information Processing Systems (NIPS 2014), Montreal, Canada, 2014, https://doi.org/10.48550/ARXIV.1409.3215.https://proceedings.neurips.cc/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf.
[39] M. Dua, R. Yadav, D. Mamgai, S. Brodiya, An improved RNN-LSTM based novel approach for sheet music generation, Procedia Comput. Sci. 171 (2020) 465–474, https://doi.org/10.1016/j.procs.2020.04.049.
[40] R. Nallapati, B. Zhou, C. dos Santos, C. Gülehre, B. Xiang, Abstractive text summarization using sequence-to-sequence RNNs and beyond. The SIGNLL Conference on Computational Natural Language Learning (CoNLL), 2016.Berlin, Allemagne
[41] R. Pascanu, T. Mikolov, Y. Bengio, On the difficulty of training recurrent neural networks. Proceedings of the 30th International Conference on Machine Learning (ICML) Vol. 28, 2013, pp. 1310–1318.Atlanta, USA
[42] Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE Trans. Neural Netw. 5 (2) (1994) 157–166, https://doi.org/10.1109/72.279181.
[43] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[44] C. Wang, L. Ma, R. Li, T. Durrani, H. Zhang, Exploring trajectory prediction through machine learning methods, IEEE Access 7 (2019) 101441–101452, https://doi.org/10.1109/ACCESS.2019.2929430.
[45] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, in: Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 2014, p. 1724–1734. https://emnlp2014.org/papers/emnlp2014-proceedings.pdf.

[46] I. Goodfellow, Y. Bengio, A. Courville, Sequence modeling: recurrentand recursive nets. DeepLearning, MIT Press, 2016, pp. 390–392.
[47] Y. Keneshloo, T. Shi, N. Ramakrishnan, C. Reddy, Deep reinforcement learning for sequence-to-sequence models, IEEE Trans. Neural Netw. Learn. Syst. 31 (7) (2020) 2469–2489, https://doi.org/10.1109/TNNLS.2019.2929141.
[48] S. Rafi, R. Das, Rnn encoder and decoder with teacher forcing attention mechanism for abstractive summarization. 2021 IEEE 18th India Council International Conference (INDICON), 2021, pp. 1–7, https://doi.org/10.1109/INDICON52576.2021.9691681.
[49] M. Schrimpf, S. Merity, J. Bradbury, R. Socher, A flexible approach to automated RNN architecture generation, CoRR (2017). abs/1712.07316.
[50] S. Jiang, P. Ren, C. Monz, M. de Rijke, Improving neural response diversity with frequency-aware cross-entropy loss. WWW '19: The World Wide Web Conference, Association for Computing Machinery, New York, NY, USA, 2019, pp. 2879–2885, https://doi.org/10.1145/3308558.3313415.
[51] H. Li, W. Lu, Mixed cross entropy loss for neural machine translation, in: M. Meila, T. Zhang (Eds.), Proceedings of the 38 th International Conference on Machine Learning, PMLR 2021, On line, July 18–24, 2021, Conference Track Proceedings, 2021.https://arxiv.org/pdf/2106.15880.pdf
[52] D. Kingma, J. Ba, Adam: a method for stochastic optimization, in: Y. Bengio, Y. LeCun (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings, 2015. abs/1412.6980.
[53] M. Nielsen, Neural Networks and Deep Learning, Determination Press, 2015, p. 23. http://neuralnetworksanddeeplearning.com/chap1.html
[54] A. Dupuis, C. Dadouchi, B. Agard, Performances of a Seq2Seq-LSTM methodology to predict crop rotations in qubec, Smart Agric. Technol. (2023).[Under review]
[55] D. Holzworth, N. Huth, P. deVoil, E. Zurcher, et al., APSIM evolution towards a new generation of agricultural systems simulation, Environ. Model. Softw. 62 (2014) 327–350, https://doi.org/10.1016/j.envsoft.2014.07.009.