



Reservoir computing based on cellular automata (ReCA) to solve the 20 QA bAbI Tasks from Facebook

* Mrwan MARGEM 

Department of Control, College of Electronic Technology -
Tripoli, Libya
m.margem@cet.edu.ly

*Corresponding Author: * Mrwan MARGEM

Keyword

*Reservoir
Computing;
Cellular
Automata;
ReCA;
Sequence
Learning;
Long-Term
Dependencies;
20 QA bAbI
Tasks*

Abstract

Reservoir computing based on cellular automata (ReCA) constructs a bridge between automata computational theory and recurrent neural architectures. Sequence learning is an essential capability for a wide collection of intelligence tasks such as language, continuous vision, and symbolic manipulation in a knowledge base, etc. Therefore, in this study, our model (ReCA) has been developed to solve the 20 QA bAbI tasks from Facebook. These tasks are very hard and require an understanding of the meaning of a text and the ability to reason over relevant facts. ReCA could solve most of the bAbI tasks; 15 out of 20 had 100% accuracy, and 2 tasks were above 90%, whilst 3 tasks were less than 90%, using only the CA evolution states at the last time step as a feature vector to predict the output (**LAST** method). The CA evolution states at all time steps (**ALL** method) can also be used, which improves the ReCA accuracy with a large feature space. ReCA results were competitive with the state-of-the-art methods in 2025, such as MT-DNS, and even ReCA outperforms NeuroLogicX, even though these methods are more complicated compared with ReCA.

Received : 12/08/2026

Accepted : 25/08/2026

DOI:

<https://doi.org/10.64943/jkc.2026.040310>

نموذج الحوسبة الخزّانية القائمة على الأوتوماتا الخلوية (ReCA) لحل مهام الأسئلة و الأجوبة (bAbI) من فيسبوك

* مروان مرغم 

قسم التحكم، كلية التقنية الإلكترونية-طرابلس، ليبيا

m.margem@cet.edu.ly

*الباحث المرسل:	* مروان مرغم
الكلمة المفتاحية	الملخص
الحوسبة الخزّانية؛ الأتمتة الخلوية؛ ReCA؛ تعلم التسلسل؛ الترابطات طويلة المدى؛ 20 مهمة أسئلة وأجوبة bAbI.	تُشكل الحوسبة الخزّانية القائمة على الأوتوماتا الخلوية (ReCA) جسراً بين نظرية الحوسبة الأوتوماتية والشبكات العصبية المتكررة. يُعدّ تعلم التسلسل الزمني قدرةً أساسيةً لمجموعة واسعة من مهام الذكاء الاصطناعي، مثل اللغة، ورؤية الحاسوب، والمعالجات الرمزية، وغيرها. ولذلك، في هذه الدراسة، تم تطوير النموذج (ReCA) لحل 20 مهمة من مهام الأسئلة و الأجوبة (bAbI) من مجموعة فيسبوك البحثية. تتسم هذه المهام بصعوبتها الشديدة، إذ تتطلب فهماً لمعنى النص، والقدرة على الاستدلال على الحقائق ذات الصلة. استطاع نموذج ReCA حل معظم هذه المهام (bAbI)؛ حيث حقق دقة 100% في 15 مهمة من أصل 20، وتجاوزت الدقة 90% في مهمتين، بينما كانت أقل من 90% في 3 مهام؛ وذلك باستخدام حالات تطور الأوتوماتا الخلوية في الخطوة الزمنية الأخيرة فقط كمتجه مميز للتنبؤ بالمخرجات (طريقة LAST). كما يمكن أيضاً استخدام حالات تطور الأتمتة الخلوية في جميع الخطوات الزمنية (طريقة ALL)؛ مما يُحسن دقة ReCA مع زيادة حجم متجه الميزات. وقد كانت نتائج ReCA منافسة لأحدث الطرق في عام 2025، مثل MT-DNS، بل وتفوقت ReCA على NeuroLogicX، على الرغم من أن هذه الطرق أكثر تعقيداً من ReCA. تاريخ الإستقبال: 2026/08/12 تاريخ القبول: 2026/08/25

DOI: <https://doi.org/10.64943/jkc.2026.040310>

1 Introduction

Reservoir Computing leverages untrained, fixed dynamical systems to project input data into high-dimensional feature space. While, Cellular Automata offer a structured, spatially distributed framework for the reservoir [1-2]. The merging of Cellular Automata into the Reservoir Computing paradigm is particularly advantageous due to its capacity for highly efficient, parallelizable hardware implementation [3].

Also, the calculation of higher-order statistics and their representation in a distributed manner, even if it is a very sparse network with identical binary units, such as a cellular automaton, will provide enough computation for sequence learning [4].

The evaluation of such an architecture requires standardized robust benchmarks, such as QA Facebook's bAbI tasks, which comprise 20 distinct challenges designed to test reasoning capabilities in a question-answering system. These tasks evaluate a model's proficiency in logical deduction and information retrieval, providing a rigorous metric for

assessing how efficiently reservoir substrates manage long-term dependencies within sequential inputs [5-6].

Next, we review reservoir computing, cellular automata, and ReCA, then provide the 20 QA bAbI tasks, learning methods and results in the following sections.

1.1 Reservoir Computing

Many real-life problems in artificial intelligence (AI) require the system to remember previous inputs. Recurrent Neural Networks (RNNs) are powerful tools of machine learning with memory. In RNNs, the input presented to the network is stored in a distributed fashion after it is transformed into hidden layer activities through a highly nonlinear function. Therefore, they employ very powerful hierarchical computation as well as distributed representation, which make them excellent tools for sequence learning tasks. Unfortunately, RNNs are difficult to train properly due to the inherent difficulty of learning optimal representations tailored for long-term dependencies [7-8] and convergence issues [9]. There are some approaches to avoid these issues, such as emulating the recurrent computation in a feedforward neural network framework [10] or utilizing a reservoir of coupled oscillators [11, 12]. It is worth noting that RC systems, which rely on dynamically evolving feature maps, learn faster than fully supervised methods trained via stochastic gradient optimization while achieving comparable accuracy [13].

RC models avoid the shortcomings of conventional training methods in RNNs by setting up RNNs in the following way [14, 15]:

1. The recurrent neural network parameters W in Figure 1.1 are randomly created and remain unchanged during training. This RNN is called the reservoir. It is excited by the input signal and maintains in its state a nonlinear transformation of the input history.
2. The desired output signal is generated as a linear combination of the neurons' signals from the input-excited reservoir. This linear combination is learned by linear regression to calculate the values of W_{out} parameters in Figure 1.1, using the teacher signal as a target.

In the RC approach, due to rich dynamics already provided by the reservoir, there is no need to train many recurrent layers, and learning takes place only at the output (read-out) stage. This simplification enables the use of recurrent neural networks in complicated tasks that require memory for long-range statistical relationships (spatially and temporally).

ESNs have been adopted in our work due to their simplicity and effectiveness, also they have been successfully applied in multiple tasks, e.g., dynamic pattern classification [16], time-series prediction and noise modelling [17], reinforcement learning [18], speech recognition [19], language modelling [20], human action recognition [21], finally handwriting recognition and movement detection [22]. In [22], the authors call ESNs reservoir computing networks (RCNs).

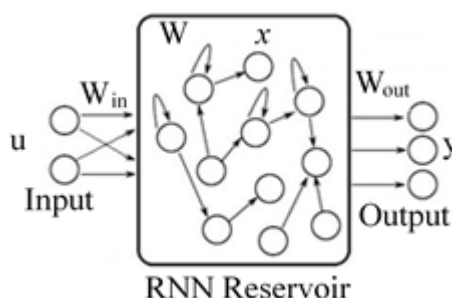


Figure 1.1 Reservoir computing paradigm. The input is projected randomly in an *RNN* (reservoir), and the evolution states of the reservoir $x(n)$ are harvested to calculate the output weight matrix W_{out} , which is then used to predict the output.

1.1.1 Physical Reservoir

Some high dimensional dynamic physical systems can be considered as a computational model [23], which led researchers to exploit these systems as physical reservoirs in Figure 1.2. The physical reservoir, of course, should verify the requirements of a conventional reservoir that are mentioned in Section 1.1. خطأ! لم يتم العثور على مصدر المرجع. Using physical systems as a reservoir starts in 2003 by water waves in a bucket [24], then the genetic regulatory network of the *Escherichia coli* bacterium [25-26], optoelectronics [27-29], random Boolean networks (RBNs) [30-32], carbon nanotubes [33-34], coupled oscillators; chemical [35], phase [36], mechanical [37], superconducting quantum computing devices [38], and very recently memristors [39-44]. The paper [45] is a review of the memristor-based implementation of Reservoir Computing. Finally, as in our model (ReCA), using different types of CA as a reservoir [1, 3-6, 10, 46-54]. Other types of physical reservoirs can be found in the papers [55, 56-60]. Also, many unconventional computing methods in [61] can be used as physical reservoirs.

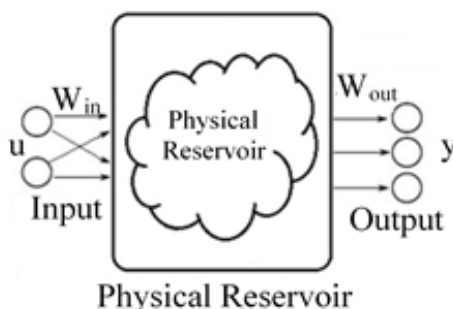


Figure 1.2 Physical Reservoir: The *RNN* reservoir in Figure 1.1 is replaced by a high dimensional dynamic physical system.

The paper [62] intends to elucidate the parallel progress of mathematical theory, algorithm design and experimental realizations of reservoir computing, and identify emerging opportunities as well as existing challenges for large-scale industrial adoption of reservoir computing. [63] covers the progress in neuromorphic devices for implementing physical reservoir computing, emphasizing device-level innovations that address the challenges of low-latency, energy-efficient, multimodal physical reservoir computing implementations. Neuromorphic computing, hyperdimensional computing, and in-memory computing are all computing paradigms that can be considered subfields of brain-inspired computing

[64]. In next-generation reservoir computing (NGRC), the small amounts of measurement noise during training act as an effective regularizer, improving long-term autonomous stability compared to standard regression alone [65].

1.2 Cellular Automata

Cellular Automata (CA) are discrete dynamical systems with sparse connections. A cellular automaton is an array of cells evolving synchronously according to an identical interaction rule. The evolution of a cell is dependent on the previous states of a surrounding neighborhood of cells, as shown in Figure 1.3 and illustrated in [66-67]. The dimension of cellular automata can have an integer value; thus, it is a grid in general, but one or two dimensions are utilized in most of the studies. CA has either null or periodic boundary. In a null boundary configuration, the boundary cells are assumed to have a logic 0, but in a periodic boundary, the grid is folded. Therefore, in a single-dimensional CA, the rightmost cell becomes the neighbor of the leftmost one. The rules applied to each cell can be either identical or different; these two different configurations are termed uniform and hybrid CA, respectively [68] (in hybrid CA, the rules change in space, not in time), but if the rules change in time (in certain iterations), the configuration is termed multilayer CA.

Elementary Cellular Automaton (ECA) is a one-dimensional CA with binary cells (1 or 0), evolving according to a uniform and non-changing rule. It is the simplest cellular automaton configuration, which will be used exclusively in our work.

The most commonly used is the visual representation, where *zero* is represented by a white square and *one* by a black square [67], as illustrated in Figure 1.3 for rule 150.

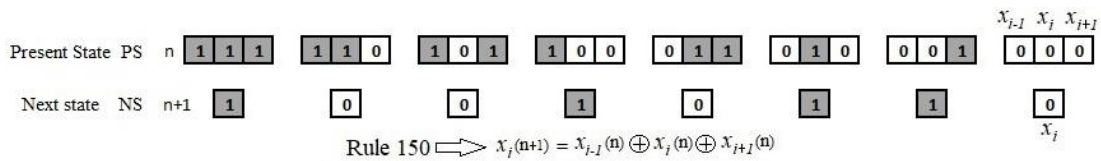


Figure 1.3 ECA Rule 150: All possible combinations for one iteration of the center cell x_i for rule 150 and its Boolean expression. The binary string $(10010110)_2$ produces the rule number $(150)_{10}$ [50].

The rules of ECA are classified according to their behavior (Wolfram classes) [67]. Starting from random initial cell values, CA state evolution will show a certain behavior: Class I (Uniform) CA states evolve to a homogeneous behavior, Class II (Periodic) CA states evolve periodically, Class III (Chaotic) CA states evolve chaotically (without any defined pattern), and Class IV (Complex or edge of chaos) can show all these evolution patterns unpredictably. Despite its simplicity, the fourth class is capable of universal computation, i.e. it can simulate a Turing machine [69-71]. CA are also useful for parallel computation and can be easily implemented using hardware such as field-programmable gate arrays (FPGAs) [72] and graphics processing units (GPUs) [73].

2 Reservoir Computing Based on Cellular Automata (ReCA)

The ReCA framework consists of three stages, as shown in Figure 2.1. The encoding stage translates the input $\mathbf{u}$ into the initial states of the CA reservoir. In the reservoir stage, the ECA rules are applied for a fixed period of iterations I , in order to evolve the CA initial states. The CA states in the reservoir are concatenated to produce a feature vector that will be used in the read-out stage using Linear Regression to predict the output $\mathbf{y}$. The details of the method are given below, step by step.

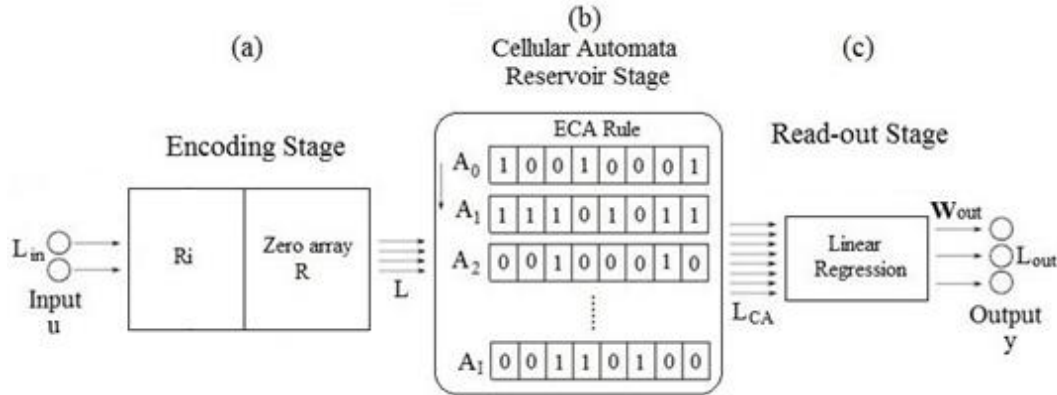


Figure 0.1 ReCA framework stages indicating vector lengths for each stage: **(a)**

Encoding Stage composed of (R_i) , which expands each input bit by R_i bits and adding (Zero Array) R -dimensional buffers for both sides of the input. **(b)** CA Reservoir Stage: The L -dimensional output of the Encoding Stage is projected onto a CA reservoir, which evolves it using a certain ECA rule in multiple time iterations I . **(c)** Read-out Stage: The reservoir output (L_{CA} -dimensional feature vector) is trained by Linear regression to find the output weights of $\mathbf{W}_{out}$ matrix, which is used to predict the output $\mathbf{y}$ [4].

2.1 Encoding Stage

The encoding stage can be divided into two subroutines, as shown in Figure 0.1(a): R_i and Zero Array R .

2.1.1 Utilizing Buffers (Zero Array R)

In the encoding stage, the input $\mathbf{u}$ is translated into the initial states of a cellular automaton reservoir. For handling a sequence of inputs, each having a length L_{in} , the size of the input to the reservoir needs to be expanded. An array of zeros with length R is added to both sides of the original input to hold the activity of the reservoir corresponding to previous input time steps. Then, the expanded input to the cellular automata reservoir becomes of length $L = L_{in} + 2R$, as shown in Figure 2.2.

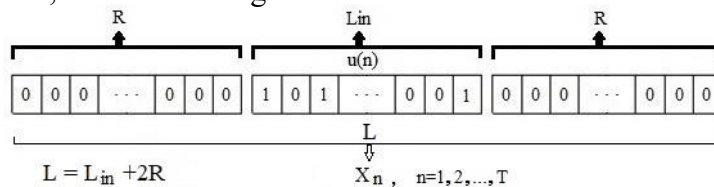


Figure 0.2 Two R -dimensional zero vectors (buffers) are added to both sides of the original input $\mathbf{u}(n)$ to produce the L -dimensional initial state $\mathbf{X}_n$ that will be evolved using ECA rule in the reservoir stage [50].

The buffer length $R=I \times T$ (I is the number of CA iterations, and T is the sequence length of the input) to guarantee that CA states have conserved all the input sequences. But, in some experiments, to reduce the time and space complexity, an expansion ratio $f \in [0, 1]$ has been introduced as follows $R = f(I \times T)$, to adjust the dimension of the buffers, thus the size of the reservoir (complexity) decreases with the value of f .

2.1.2 Reducing Interference R_i

To improve the ReCA accuracy, the interference between nonzero elements in the reservoir should be reduced. The interference is due to the cancellation of cell activity when two nonzero cells collide during additive cellular automaton rules such as rule 90¹. In order to avoid it, each bit of the input can be represented by R_i bits. At the insertion of a new time step, the location of the nonzero will rotate right one bit to reduce the interference between nonzero bits that have the *same* location in consecutive time steps (for more detail, see [4]). Thus, the length of the CA reservoir input will be expanded to $L=L_{in} \times R_i + 2R$. The value of R_i should be chosen carefully to reduce the interference between nonzero bits that have *different* locations in consecutive time steps. Hence, the input dimension is artificially expanded to increase the feature space size and reduce the interference between nonzero bits in the reservoir.

2.2 Cellular Automata Reservoir Stage

After the data is processed in the encoding stage as the initial states of a cellular automaton, it is passed on a CA reservoir (instead of an ESN as in [11]) for computation. The dynamics of CA provide the necessary projection of the input data onto an expressive and discriminative space, as shown in Figure 2.1(b). It was previously shown that the cellular automata reservoir holds a distributed representation of high order attribute statistics [46]. Thus, the sequence of inputs at each time step is processed to extract input statistics, which are represented in a distributed manner as in recurrent neural networks. However, different from recurrent neural networks, for each input, we can allocate a distinct memory slot in the reservoir space and avoid interference between different time steps altogether (at the cost of increased feature space). This is due to the fact that additive cellular automata rules propagate the cell activities of the previous time steps in a predictable manner, creating "empty spots" for injection of new input in the sequence.

The cellular automaton is initialized with the first time step input of the sequence (X_1) that has been obtained from the encoding stage; $A_0^{(1)} = X_1$ (with size of L), where the subscript 0 denotes the initial state and increases to I , which is the number of CA iterations in the reservoir, and the superscript (1) denotes the number of time steps (from 1 to T), where T is the input sequence length. Then, the cellular automata evolution of $A_0^{(1)}$ is computed using a pre-specified ECA rule CAr up to I iterations as illustrated in Figure 2.1(b) and the following equations:

$$\text{Boolean expression of rule 90: } x_i(n+1) = x_{i-1}(n) \oplus x_{i+1}(n).^1$$

$$A_1^{(1)} = CAr(A_0^{(1)}) \quad (0.1)$$

$$A_2^{(1)} = CAr(A_1^{(1)}) \quad (2.2)$$

$\vdots$

$$A_I^{(1)} = CAr(A_{I-1}^{(1)}) \quad (2.3)$$

The evolution of CA states is concatenated to obtain a single state vector as in equation (2.4), with length of $L_{CA} = I \times L$, that will be used as a feature vector to predict the output at the 1st time step:

$$A^{(1)} = [A_1^{(1)} \ A_2^{(1)} \ \dots \ A_I^{(1)}] \quad (2.4)$$

At the 2nd time step, the input X_2 will be inserted into the reservoir state vector. Applying an insertion function, which is in our model, an XOR operation² to the last state vector $A_I^{(1)}$ with the input vector at the second time step X_2 . Then the new initial state vector of the cellular automaton at time step $n = 2$ will be as:

$$A_0^{(2)} = A_I^{(1)} \oplus X_2 \quad (0.5)$$

where $\oplus$ is a bitwise XOR.

Then, the cellular automaton is evolved for I steps to obtain $A^{(2)}$ as in equation (2.6).

$$A^{(2)} = [A_1^{(2)} \ A_2^{(2)} \ \dots \ A_I^{(2)}] \quad (2.6)$$

² XOR computes the correlation, which provides a lossless merging of two binary numbers: it outputs 1 if the content of input cells is different and 0 if the input cells are identical.

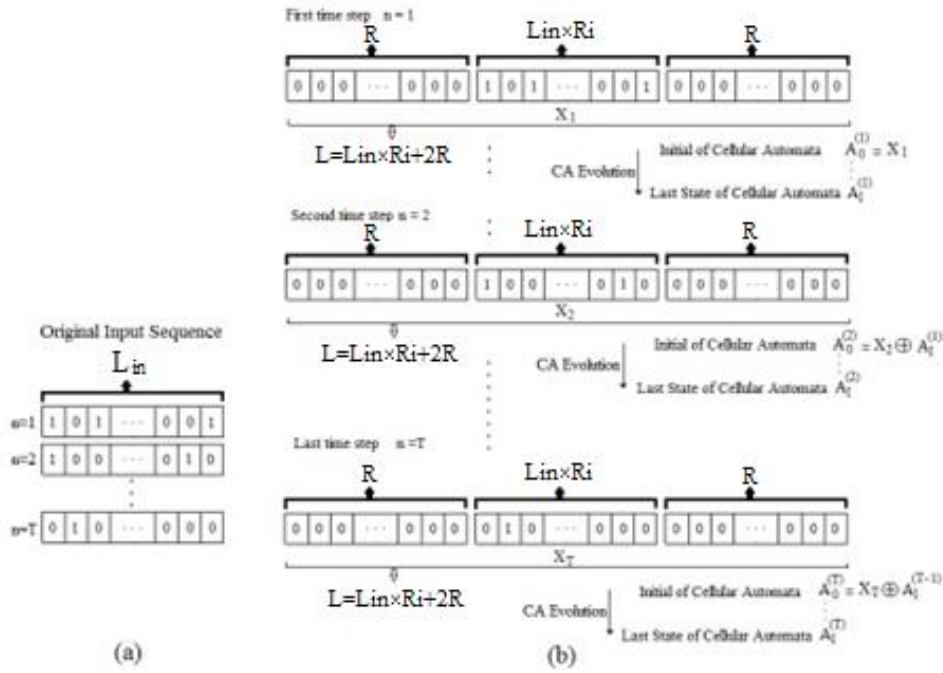


Figure 2.3 (a) Original input sequence before the encoding stage. (b) Encoding stage and

CA Reservoir stage: Adding a zero array with length R to obtain the reservoir input sequence from X_1 to X_T . Then, the cellular automaton is initialized with the first-time step input of the sequence, so $A_0^{(1)} = X_1$ (with size of L). The CA evolution states $A_i^{(n)}$ (i changes from 1 to I) will be used as a feature space to estimate the output $y(n)$ using linear regression in the read-out stage [4].

$A^{(2)}$ is used for estimation at the 2nd time step. This procedure is continued until the end of the sequence at $n = T$, when X_T is inserted into the reservoir and obtaining $A^{(T)}$ that will be used to predict the output at the last time step; the details are depicted in Figure 2.3.

Note: Any feature vector $A^{(n)}$ memorizes the input information from 1st time step up to its time step n , i.e., $A^{(T)}$ memorizes the whole input information.

2.3 Read-out stage

In this stage, the output states of the CA reservoir are used as a feature vector (with dimension of L_{CA}) in linear regression to compute the weight values of $\mathbf{W}_{out}$ matrix, which is used to predict the output; thus, this output will be a vector with dimension of L_{out} . Therefore, the output $y(T)$ can be expressed as in (2.7):

$$\mathbf{y}(T) = \mathbf{W}_{out} \cdot \mathbf{A}^{(T)} \quad (0.6)$$

where $\mathbf{A}^{(T)}$ is the feature vector of the last time step with dimension of $I \cdot L$, and the size of the regression parameters matrix $\mathbf{W}_{out}$ is $L_{out} \times IL$. This method will be called **LAST** because ReCA uses the CA evolution states only at the *last* time step.

The feature space can be increased by choosing the CA evolution states at *all* time steps from $n=1$ to $n=T$. Then, all of these states are concatenated to produce a large single feature vector $\mathbf{A}^{(ALL)}$ (with dimension of $L_{CA} = I \times L \times T$). This method will be called

ALL and used to predict the output $\mathbf{y}(T)$. **ALL** method can be considered as an extra memory resource in RNNs, as in [74-77].

3 The Facebook 20 QA bAbI Tasks

The 20 QA bAbI tasks from Facebook [78] are complex relational reasoning tasks in natural language processing (NLP); they require an understanding of the meaning of a text and the ability to reason over relevant facts.

Most tasks in NLP can be considered as a QA problem: high level tasks like machine translation (What is the translation into French?); sequence modeling tasks like named entity recognition (What are the named entity tags in this sentence?) or part-of-speech tagging (What are the part-of-speech tags?); classification problems like sentiment analysis (What is the sentiment?); even multi-sentence joint classification problems like coreference resolution (Who does 'their' refer to?) [79].

The 20 QA bAbI tasks are a synthetic question-answering dataset from the bAbI project of Facebook AI Research, which is organized towards the goal of automatic text understanding and reasoning. It contains 20 tasks; each of them is composed of a set of sentences³ (story), a question related to some of those sentences, and followed by an answer, which is mostly a single word (in some tasks, the answer is a set of words). In most cases, only a subset of facts (sentences) is relevant to the given question (called *supporting* facts (SF), which are also included in the training set). All the tasks are noiseless, and a human able to read that language can potentially achieve 100% accuracy. Two versions of the data are available: the 1st has 1K training examples per task, and the 2nd has 10K examples per task, while the testing set is 1K examples for both versions [78]. The questions in these tasks are quite hard; they not only require lots of knowledge in natural sciences but also abilities to make inferences, generalize the concepts, apply the general ideas to the examples and so on. These 20 tasks can be summarized into 11 groups:

- 1- One, Two, or Three supporting facts (Task 1, 2 and 3): In order to answer the question, one, two, or three supporting sentences have to be used as demonstrated in Figure 3.1.

The sentences are as independent from others as possible.³

Category 1: Single Supporting Fact 01: Mary moved to the bathroom. 02: John went to the hallway. 03: Where is Mary? bathroom 1 04: Daniel went back to the hallway. 05: Sandra moved to the garden. 06: Where is Daniel? hallway 4	Category 3: Three Supporting Facts 01: Sandra went back to the hallway. 02: Daniel took the apple. 03: John travelled to the kitchen. 04: Daniel travelled to the bedroom. 05: Daniel got the football there. 06: Daniel went to the hallway. 07: Where was the apple before the hallway? bedroom 2 6 4 08: Mary went back to the bedroom. 09: Daniel discarded the football. 10: Daniel got the football. 11: Mary went to the garden. 12: Daniel travelled to the office. 13: Daniel went back to the bedroom. 14: Where was the football before the bedroom? office 10 13 12 15: Daniel went back to the hallway. 16: Mary went back to the bathroom. 17: Daniel dropped the apple. 18: Sandra journeyed to the kitchen. 19: Where was the apple before the office? hallway 17 12 6
Category 2: Two Supporting Facts 01: Mary went to the kitchen. 02: Sandra journeyed to the office. 03: Mary got the football there. 04: Mary travelled to the garden. 05: Where is the football? garden 3 4 06: John travelled to the office. 07: Sandra moved to the garden. 08: Where is the football? garden 3 4 09: Mary dropped the football. 10: Mary journeyed to the kitchen. 11: Where is the football? garden 9 4	

Figure 3.1 One, Two, or Three SFs tasks: Story (black), question (blue), answer (red), and indices of the supporting facts (green) [80].

- 2- Two or Three argument relations (Task 4 and 5): The ability to differentiate and recognize subjects and objects is necessary to answer the task, as illustrated in Figure 3.2.

Category 4: Two Argument Relations 01: The office is north of the kitchen. 02: The garden is south of the kitchen. 03: What is north of the kitchen? office 1 01: The kitchen is west of the garden. 02: The hallway is west of the kitchen. 03: What is the garden east of? kitchen 1	Category 5: Three Argument Relations 01: Jeff took the milk there. 02: Jeff gave the milk to Bill. 03: Who did Jeff give the milk to? Bill 2 04: Daniel travelled to the office. 05: Daniel journeyed to the hallway. 06: Who received the milk? Bill 2 07: Bill went to the kitchen. 08: Fred grabbed the apple there. 09: What did Jeff give to Bill? milk 2
---	--

Figure 3.2 Two or Three argument relation tasks: the colors as in Figure 3.1 [80].

- 3- Yes/No Questions (Task 6): to test the ability of the model to answer true/false type questions as demonstrated in Figure 3.3.

Category 6: Yes/No Questions 01: Daniel went back to the hallway. 02: John got the apple there. 03: Is Daniel in the hallway? yes 1 04: John dropped the apple. 05: Mary got the apple there. 06: Is Daniel in the hallway? yes 1 07: Daniel moved to the bedroom. 08: Sandra travelled to the hallway. 09: Is Daniel in the hallway? no 7
--

Figure 3.3 Yes/No Question Task: the colors as in Figure 3.1 [80].

- 4- Counting and Lists/Sets (Task 7 and 8): Task 7 tests the ability of the model to perform counting operations. Task 8 tests the ability to produce a list of single-word answers as demonstrated in Figure 3.4.

Category 7: Counting 01: Mary took the apple there. 02: John travelled to the office. 03: How many objects is Mary carrying? one 1 04: Mary travelled to the bathroom. 05: Sandra went back to the bedroom. 06: How many objects is Mary carrying? one 1 07: Mary got the football there. 08: Mary went to the office. 09: How many objects is Mary carrying? two 1 7 10: Mary passed the apple to John. 11: Mary left the football. 12: How many objects is Mary carrying? none 1 7 10 11	Category 8: List/Sets 01: Mary took the milk there. 02: Mary went to the office. 03: What is Mary carrying? milk 1 04: Mary took the apple there. 05: Sandra journeyed to the bedroom. 06: What is Mary carrying? milk,apple 1 4 07: Mary put down the milk. 08: Mary discarded the apple. 09: What is Mary carrying? nothing 1 7 4 8
---	---

Figure 3.4 Counting and Lists/Sets: the colors as in Figure 3.1 [80].

- 5- Simple Negation and Indefinite Knowledge (Task 9 and 10): Task 9 is similar to Task 6, but with the possibility of negative supporting facts. Task 10 tests if the model can describe the uncertainty as demonstrated in Figure 3.5.

Category 9: Simple Negation 01: Sandra travelled to the garden. 02: Sandra is no longer in the garden. 03: Is Sandra in the garden? no 2 04: Sandra is in the garden. 05: Sandra journeyed to the hallway. 06: Is Sandra in the hallway? yes 5	Category 10: Indefinite Knowledge 01: Julie travelled to the kitchen. 02: Bill is either in the school or the office. 03: Is Bill in the office? maybe 2 04: Bill went back to the bedroom. 05: Bill travelled to the kitchen. 06: Is Bill in the kitchen? yes 5
---	---

Figure 3.1 [80].

- 6- Basic Coreference, Conjunctions and Compound Coreference (Task 11, 12 and 13): Task 11 tests the basic type of coreference to detect the single subject. Task 12 tests referring to multiple subjects in a single sentence. Task 13 tests referring to multiple subjects in multiple sentences as demonstrated in Figure 3.6.

Category 11: Basic Coreference 01: Mary went back to the bathroom. 02: After that she went to the bedroom. 03: Where is Mary? bedroom 1 2 04: Daniel moved to the office. 05: Afterwards he moved to the hallway. 06: Where is Daniel? hallway 4 5	Category 12: Conjunction 01: Daniel and Sandra went back to the kitchen. 02: Daniel and John went back to the hallway. 03: Where is Daniel? hallway 2 04: Daniel and John moved to the bathroom. 05: Sandra and Mary travelled to the office. 06: Where is Daniel? bathroom 4
---	--

Category 13: Compound Coreference 01: Mary and Daniel went to the bathroom. 02: Then they journeyed to the hallway. 03: Where is Daniel? hallway 1 2 04: Sandra and John moved to the kitchen. 05: Then they moved to the hallway. 06: Where is John? hallway 4 5
--

Figure 3.6 Basic Coreference, Conjunctions, and Compound Coreference: the colors as in Figure 3.1 [80].

- 7- Time Reasoning: (Task 14): tests the understanding of time expressions within the story as illustrated in Figure 3.7.

Category 14: Time Manipulation	
01: Yesterday Julie went back to the park.	
02: Julie went to the bedroom this morning.	
03: Bill journeyed to the cinema yesterday.	
04: This morning Bill went back to the park.	
05: Where was Bill before the park?	cinema 4 3
06: This evening Julie went to the school.	
07: This afternoon Julie went back to the park.	
08: Where was Julie before the bedroom?	park 2 1

Figure 3.7 Time Reasoning: the colors as in Figure 3.1 [80].

- 8- Basic Deduction and Induction (Task 15 and 16): Task 15 tests basic deduction. Task 16 tests basic induction as demonstrated in Figure 3.8.

Category 15: Basic Deduction		Category 16: Basic Induction	
01: Mice are afraid of cats.		01: Bernhard is a lion.	
02: Emily is a mouse.		02: Julius is a lion.	
03: Wolves are afraid of mice.		03: Lily is a lion.	
04: Cats are afraid of sheep.		04: Bernhard is green.	
05: Winona is a cat.		05: Lily is green.	
06: Sheep are afraid of wolves.		06: Brian is a lion.	
07: Jessica is a mouse.		07: Greg is a swan.	
08: Gertrude is a sheep.		08: Greg is gray.	
09: What is Jessica afraid of?	cat 7 1	09: Julius is yellow.	
10: What is Emily afraid of?	cat 2 1	10: What color is Brian?	green 6 3 5
11: What is Jessica afraid of?	cat 7 1		
12: What is Winona afraid of?	sheep 5 4		

Figure 3.8 Basic Deduction and Induction: the colors as in Figure 3.1 [80].

- 9- Positional and Size Reasoning (Task 17 and 18): Task 17 tests the spatial reasoning about the relative positions. Task 18 tests the understanding of the relative size of objects as illustrated in Figure 3.9.

Category 17: Positional Reasoning		Category 18: Reasoning about Size	
01: The triangle is above the pink rectangle.		01: The suitcase is bigger than the container.	
02: The blue square is to the left of the triangle.		02: The container fits inside the box.	
03: Is the pink rectangle to the right of the blue square?	yes 1 2	03: The chest is bigger than the chocolate.	
		04: The suitcase fits inside the box.	
01: The red sphere is below the yellow square.		05: The chest fits inside the box.	
02: The red sphere is above the blue square.		06: Does the chocolate fit in the box?	yes 5 3
03: Is the blue square below the yellow square?	yes 2 1	07: Does the chocolate fit in the box?	yes 5 3
		08: Does the box fit in the container?	no 1 4
		09: Is the box bigger than the chocolate?	yes 5 3
		10: Does the box fit in the chocolate?	no 3 5

Figure 3.9 Positional and Size Reasoning: the colors as in Figure 3.1 [80].

- 10- Path Finding (Task 19): To test the ability to find the path between locations as demonstrated in Figure 3.10.

Category 19: Path Finding
 01: The bedroom is south of the hallway.
 02: The bathroom is east of the office.
 03: The kitchen is west of the garden.
 04: The garden is south of the office.
 05: The office is south of the bedroom.
 06: How do you go from the garden to the bedroom? n.n 4 5

Figure 3.10 Path Finding: the colors as in Figure 3.1 [80].

11- Agents Motivations (Task 20): tests *why* the agent performs an action and *why* the action has been done.

Category 20: Reasoning about Motivations
 01: Sumit is bored.
 02: Where will Sumit go? garden 1
 03: Yann is hungry.
 04: Where will Yann go? kitchen 3
 05: Yann went back to the kitchen.
 06: Why did Yann go to the kitchen? hungry 3
 07: Sumit journeyed to the garden.
 08: Why did Sumit go to the garden? bored 1
 09: Yann picked up the apple there.
 10: Why did Yann get the apple? hungry 3
 11: Sumit grabbed the football there.
 12: Why did Sumit get the football? bored 1

Figure 3.11 Agents' Motivations: the colors as in Figure 3.1 [80].

4 Related Work

The 20 QA bAbI tasks have been studied within the context of the Memory Network (MemNN) model [74,78], which consists of four learnable modules: the I-module encodes the input into feature representation, the G-module updates relevant memory slots, the O-module performs inferences to compute output features given the input representation and the current memory, and finally the R-module decodes the output feature-based representation to the final response. Since the proposal of the basic MemNN [74] model, the Adaptive/Nonlinear MemNN [78], Dynamic Memory Networks (DMN)⁴ [79], and End-to-End Memory Networks (MemN2N) [81] models have been developed by varying certain parts of these modules. The difference for MemN2N is that the indices of supporting facts are no longer provided in the dataset (weakly supervised), while in the other models the indices of supporting facts are used in the training process (strongly supervised). Hence, the MemN2N model must deduce for itself at training and test time which sentences are relevant and which are not, making MemN2N harder and more generally applicable in realistic settings. MemN2N can also be considered as an extension of RNNsearch [82] to the case where multiple computational steps (hops) are performed per output symbol [81]. Within the context of Deep Learning (DL), Neural Machine Translation (NMT) and Neural Turing Machine (NTM) have been proposed to solve the 20 bAbI tasks [83]. Neural Reasoner (NR) is another approach using a deep architecture, which is a framework for neural network-based reasoning over natural language sentences. NR has two essential specifications:

⁴ In DMN Global Vectors for Word Representation (GloVe) [122] has been used instead of the random numbers for word representation in the other methods.

1. An interaction-pooling mechanism which allows NR to examine multiple facts, and 2. a deep architecture, allowing it to deal with the complicated logical relations in reasoning tasks [84], in which only the most difficult tasks (task 17 Positional Reasoning and task 19 Path Finding) from the 20 tasks have been tested and gave superior results for both tasks in weakly supervised. Finally, in [80], the authors propose a vector-space model inspired by Tensor Product Representation (TPR) [85], which achieves the best results for all 20 bAbI tasks, but by using the *language principles* in the preprocessing of the 20 QA bAbI dataset. Human intervention, e.g., using the *language principles*, is not allowed in the QA bAbI tasks from Facebook [78], because the target of the Facebook research team is to produce a language-independent model.

Table 4.1 lists the best results obtained from different models that were used to solve question-answering bAbI tasks and indicates that using supporting facts (Strongly supervised) increases the accuracy due to the extra information provided by the task. The most difficult tasks for all models are task 17 (Positional Reasoning) and 19 (Path Finding) because they require a general search algorithm to be built into the inference procedure [78], only TPR model in [80] could solve both tasks due to using external resources, not in the training data (the language principles that have been used in the TPR model).

Table 4.1 The 20 QA bAbI tasks results: *Facebook* team work on MemNN [78] and MemN2N [81], MitaMind Lab works on DMN [79], *Microsoft* team work on reasoning in vector space (TPR model) [80], *IBM* team work on NMT and NTM [83], and Noah's Ark Lab, *Huawei* Technologies team work on NR [84]. LSTM results are obtained from [78] using the LSTM created in [8].

Task No	Strongly supervised					Weakly supervised			
	MemNN	DMN	NMT	NTM	TPR	MemN2N	NR	NMT	LSTM
1	100%	100%	100%	100%	100%	100%		98.2%	50%
2	100%	98.2%	99.6%	100%	100%	91.7%		41.3%	20%
3	100%	95.2%	99.5%	100%	100%	59.7%		33.4%	20%
4	100%	100%	97.5%	100%	100%	97.2%		97.8%	61%
5	98%	99.3%	90.6%	73.7%	99.8%	86.9%		90.3%	70%
6	100%	100%	99.8%	100%	100%	92.4%		84.6%	48%
7	85%	96.9%	96.6%	100%	100%	82.7%		82.4%	49%
8	91%	96.5%	92.7%	98%	100%	90%		70.8%	45%
9	100%	100%	99.7%	100%	100%	86.8%		89.3%	64%
10	98%	97.5%	96.8%	85.9%	100%	84.9%		73.5%	44%
11	100%	99.9%	100%	100%	100%	99.1%		99.8%	62%
12	100%	100%	100%	100%	100%	99.8%		99.4%	74%
13	100%	99.8%	100%	100%	100%	99.6%		99.7%	94%
14	99%	100%	97.5%	100%	100%	99.3%		44.4%	27%
15	100%	100%	92.7%	100%	100%	100%		42.9%	21%
16	100%	99.4%	88.1%	100%	99.5%	98.7%		42.7%	23%
17	65%	59.6%	58%	61.2%	100%	49%	66.4%	64.6%	51%
18	95%	95.3%	91.8%	93%	100%	88.9%		90.9%	52%

19	36%	34.5%	29.7%	100%	100%	17.2%	17.3%	9.3%	8%
20	100%	100%	93.3%	100%	100%	100%		91.6%	91%
Mean accuracy	93.3%	93.6%	91.2%	95.6%	99.97%	86.1%		72.3%	48.7%
Failed Tasks Accuracy < 90%	3	2	3	3	0	8		12	18

In [87], the proposed framework to solve the 20 bAbI tasks is built by embedding questions and answers based on bidirectional long short-term memory (biLSTM) attentive models. The similarity between questions and answers has been measured by semantic and cosine similarity. The (biLSTM) model achieves (85.7% mean accuracy) [87]. The memory module (MN) is made up of two GRUs (Gated Recurrent Units) [88]; this method will be called MN(GRU) in Table 4.2. The researchers in [89] built a neural network model, S2G-TextCNN, that followed the sequence-input grid-output architecture by varying the grid decoder according to target modalities of problems. They compared their model (S2G-TextCNN) with Universal Transformer (UT) [90] and LSTM seq2seq model (LSTM) [91]. SPARTQA model has been proposed in [92] to solve only task 17 (Positional-Reasoning); they reach 100% accuracy after 1000 training set, but by using the *language principles* in their model, as in [80].

Table 4.2 The 20 QA bAbI tasks results for the models (continue): LSTM-s2s [91], UT [90], S2G-TextCNN [89], MN(GRU) [88], DNC [76], MT-DNC [94], NeuroLogicX [93], and biLSTM [87].

Task No	LSTM-s2s	UT	S2G-TextCNN	MN(GRU)	DNC	MT-DNC	NeuroLogicX	biLSTM
1	100.0%	100.0%	100.0%	99.90%	78.4%	100.0%	96.8%	
2	52.6%	45.0%	68.8%	54.60%	40.3%	99.3%	93.4%	
3	54.1%	32.1%	68.5%	44.20%	44.0%	96.5%	92.5%	
4	99.9%	100.0%	100.0%	99.70%	98.9%	100.0%		
5	99.2%	94.5%	99.0%	91.10%	97.5%	99.4%		
6	99.5%	99.9%	100.0%	90.20%	85.6%	100.0%		

7	98.2%	96.0%	100.0%	85.50%	83.2%	99.2%		
8	99.8%	97.7%	98.2%	77.50%	88.6%	99.8%		
9	100.0%	100.0%	100.0%	88.30%	84.0%	100.0%		
10	99.7%	100.0%	100.0%	96.20%	79.0%	100.0%		
11	100.0%	99.9%	100.0%	98.50%	91.0%	100.0%		
12	100.0%	100.0%	100.0%	97.60%	88.7%	100.0%		
13	100.0%	100.0%	100.0%	96.60%	93.3%	100.0%		
14	79.4%	95.6%	92.7%	56.30%	81.5%	100.0%		
15	65.2%	81.5%	100.0%	66.10%	52.7%	100.0%		
16	47.9%	46.4%	48.3%	67.80%	44.5%	50.1%		
17	58.9%	59.0%	68.6%	70.20%	59.6%	98.3%		
18	91.4%	90.9%	96.2%	92.80%	94.0%	100.0%		
19	9.1%	20.9%	64.9%	37.30%	0.0%	98.8%		
20	98.2%	98.6%	100.0%	98.20%	99.9%	100.0%		
Mean Accuracy	82.7%	82.9%	90.3%	80.43%	74.2%	97.1%		85.70%
Failed Tasks Accuracy < 90%	7	7	5	10	14	1		

Note: The results of MT-DNC in [94] are ranges; we choose the lower value in the Table.

In [93], the author proposed a modular neurosymbolic framework called (NeuroLogicX), he evaluated his model using only the first three tasks of 20 bAbI tasks as shown in Table 4.2. Finally, A brain-inspired Memory Transformation based Differentiable Neural Computer (MT-DNC) has been proposed in [94] to solve the 20 bAbI tasks. Where, Differentiable Neural Computer (DNC) is an improved version of the Neural Turing Machine (NTM) in [75], using external memory to enhance the ability of neural networks [76, 95-96]. The model (MT-DNC) consists of three layers: Controller Layer, Memory Layer, and Linear Layer. The total number of parameters in (MT-DNC) is 1,267,337 [94]. The whole results are listed in Table 4.2.

5 Training Methods

The 20 QA bAbI Tasks dataset can be trained in three ways:

- 1- Using only supporting facts (sentences) and removing all distractor facts.
- 2- Using all facts (the whole story) with the indices of supporting facts; this method is called (strongly supervised).
- 3- Using only all facts (the whole story) without the indices of supporting facts; this method is called (weakly supervised).

In this work, we adopt the first approach to check whether ReCA can solve these tasks or not⁵.

Figure 5.1 shows the original story of task 3 as an example, while Figure 5.2 presents task3 using only supporting facts, and the question is inserted at the end of the story.

```
Task 3 (Original)
1 Mary moved to the bathroom.
2 Sandra journeyed to the bedroom.
3 Mary got the football there.
4 John went back to the bedroom.
5 Mary journeyed to the office.
6 John journeyed to the office.
7 John journeyed to the bathroom.
8 Mary journeyed to the bathroom.
9 Sandra went back to the garden.
10 Daniel journeyed to the office.
11 Mary dropped the football.
12 John moved to the bedroom.
Where was the football before the bathroom? office 11 8 5
```

Figure 5.1 The original story of task3 where supporting facts(blue), question (red), and answer (green) and indices of supporting facts (bold blue).

```
Task 3 (Only Supporting Facts)
Mary journeyed to the office.
Mary journeyed to the bathroom.
Mary dropped the football.
Where was the football before the bathroom?
```

Figure 5.2 Using only supporting facts in task 3 in Figure 5.1, this dataset will be used for training and testing.

⁵ Because, if ReCA could not solve bAbI tasks using only Supporting facts, a fortiori it cannot solve them using strongly or weakly supervised

Figure 5.3 demonstrates how to convert the story and question to a matrix using random numbers to represent all words.

Input								T
Office	2	3	4	5	6	0	0	1
	2	3	4	5	7	0	0	2
	2	8	5	9	0	0	0	3
	10	11	5	9	12	5	7	4

Figure 5.3 Input and output of task3 (Only supporting facts): Converting the story and question in the Figure to a matrix after representing each word by a number. Thus, the input matrix is 4×7 (4-time steps, i.e., T=4), and the output is labeled by **6** (means office).

The number **1** was not used because it is reserved to represent the space between sentences.

Then, the decimal numbers in the input matrix and the output number in Figure 5.3 are binarized using one-hot encoding. The binarized matrix is the input sequence **u** that will be used in ReCA as explained in previous sections. The target is that ReCA can capture the pattern of the input sequence to predict the output (answer) correctly; thus, our model is language-independent, i.e., this model can be used for any language.

6 Results

After preparing the dataset as in the previous section, ReCA is implemented as in Section 2, where I is the number of CA evolution iterations in the reservoir and Rule No is the number of ECA rule has been used to achieve the best results.

ReCA could solve 15 tasks with 100% accuracy and 2 tasks above 90%, whilst 3 tasks less than 90%. Most tasks achieve the best results with rule 90; only 2 tasks use rules 60 and 150. The options ALL and LAST are almost the same as listed in Table 6.1.

For comparison, the IBM research team in [83] have used only SFs as in our work. The results of their two methods, Neural Machine Translator NMT and Neural Turing Machine NTM, and the results of MN(GRU) [88], S2G-TextCNN [89], and MT-DNC [94] are listed in Table 6.2. Table 6.2 shows that ReCA achieves very good results, outperforming MN(GRU) and S2G-TextCNN.In terms of NMT, ReCA outperforms in 4 tasks (4, 5, 14, and 18) and falls behind in 4 tasks (8, 10, 17, and 19). For NTM and MT-DNC, they outperform ReCA a little bit, but with very complicated models. For the recent method in 2025, NeuroLogicX [93], ReCA could solve the first three tasks with 100% accuracy, but NeuroLogicX solved these tasks with around 90% accuracy.

The low ReCA mean accuracy is due to very poor performance on tasks 17 and 19 compared with other methods, which had a dramatic impact on its mean accuracy, as shown in Table 6.2.

Table 6.1 ReCA accuracy for all 20 QA bAbI tasks using **ALL** and **LAST** options, where I is the number of CA iterations in the reservoir.

Task	Method	Rule No	I	Accuracy
1	ALL/LAST	90	1	100%
2	ALL/LAST	90	2	100%
3	ALL/LAST	90	1	100%
4	ALL/LAST	90	1	100%
5	ALL/LAST	90	56	100%
6	ALL/LAST	90	12	100%
7	ALL/LAST	60/150	17/4	77%/76%
8	ALL/LAST	90	4	91%/90%
9	ALL/LAST	90	8	100%
10	ALL/LAST	150	39	96%
11	ALL/LAST	90	2	100%
12	ALL/LAST	90	2	100%
13	ALL/LAST	90	3	100%
14	ALL/LAST	90	1/6	100%/90%
15	ALL/LAST	90	2	100%
16	ALL/LAST	90	2	100%
17	ALL/LAST	60/90	128/99	43%/44%
18	ALL/LAST	90	22/17	100%
19	ALL/LAST	90	56/84	38%/39%
20	ALL/LAST	90	17	100%

Table 6.2 Comparison between ReCA (ALL, LAST) and the other models (NMT, NTM, MN(GRU), S2G-TextCNN, and MT-DNC).

Task No	ReCA		NMT	NTM	MN(GRU)	S2G-TextCNN	MT-DNC
	ALL	LAST					
1	100%	100%	100%	100%	99.9%	100.0%	100.0%
2	100%	100%	100%	100%	54.6%	68.8%	99.3%
3	100%	100%	100%	100%	44.2%	68.5%	96.5%
4	100%	100%	99.1%	100%	99.7%	100.0%	100.0%
5	100%	100%	99.3%	79.2%	91.1%	99.0%	99.4%
6	100%	100%	100%	100%	90.2%	100.0%	100.0%
7	77%	76%	68.5%	100%	85.5%	100.0%	99.2%
8	91%	90%	99%	100%	77.5%	98.2%	99.8%
9	100%	100%	100%	100%	88.3%	100.0%	100.0%
10	96%	96%	98.9%	94.6%	96.2%	100.0%	100.0%
11	100%	100%	100%	100%	98.5%	100.0%	100.0%
12	100%	100%	100%	100%	97.6%	100.0%	100.0%
13	100%	100%	100%	100%	96.6%	100.0%	100.0%
14	100%	90%	99.8%	100%	56.3%	92.7%	100.0%

15	100%	100%	100%	100%	66.1%	100.0%	100.0%
16	100%	100%	100%	100%	67.8%	48.3%	50.1%
17	43%	44%	64.2%	69.3%	70.2%	68.6%	98.3%
18	100%	100%	97.8%	93%	92.8%	96.2%	100.0%
Task No	ReCA		NMT	NTM	MN(GRU)	S2G-TextCNN	MT-DNC
	ALL	LAST					
19	38%	39%	80.7%	100%	37.3%	64.9%	98.8%
20	100%	100%	100%	100%	98.2%	100.0%	100.0%
Mean Accuracy	92.25%	91.75%	95.37%	96.81%	80.43%	90.3%	97.1%
Failed Tasks Accuracy < 90%	3	3	3	2	10	5	1

6.1 Discussions

ReCA has been used to solve the 20 QA bAbI tasks from Facebook. These tasks are very hard and complex natural language processing (NLP) tasks and require an understanding of the meaning of a text and the ability to reason over relevant facts.

ReCA could solve most of the bAbI tasks; 15 out of 20 have 100% accuracy, and 2 tasks are above 90%, whilst 3 tasks are less than 90%. Most tasks achieve best results with rule 90; only 2 tasks used rules 60 and 150. Hence, all the successful rules are linear (additive) to maximize one-to-one correspondence between the input sequence and the reservoir activity due to the sequence, i.e., reducing the unwanted interference. The options **ALL** and **LAST** are almost the same because the sequence is not long in most tasks. The harder tasks need a greater number of iterations to achieve high accuracy, i.e., to improve computational power [7, 85]. The most difficult tasks for ReCA are 17 (Positional Reasoning) and 19 (Path Finding), with a large difference, which reduces the mean accuracy of ReCA dramatically. Tasks 17 and 19 are the hardest tasks for most models, as explained in Section 4. Task 7 (Counting) can also be considered a hard task for ReCA (accuracy 77%) because the answer is not a word in the story and also needs mathematical operations (adding and subtracting). We should note that the other methods are very complicated compared with our model ReCA, as explained in Section 4.

7 Conclusion

ReCA has been used to solve the 20 QA bAbI tasks from Facebook. These tasks are very hard, complex natural language processing (NLP) tasks, and require an understanding of the meaning of a text and the ability to reason over relevant facts. Using only supporting facts, ReCA could solve most of the bAbI tasks; 15 out of 20 have 100% accuracy, and 2 tasks are above 90%, whilst 3 tasks are below 90%. The options **ALL** and **LAST** are almost the same because the sequence is not long in these tasks. ReCA outperforms most models that are used to solve the 20 bAbI tasks, including the state-of-the-art methods, such as NeuroLogicX, and it is very close to NTM and MT-DNC mode, although these models are very complicated compared with ReCA.

In addition, the usage of cellular automata in the reservoir computing paradigm greatly simplifies the architecture, makes the computation more transparent for analysis, and

provides enough computation sequence learning tasks. Furthermore, the reservoir in ReCA can be implemented using ordinary logic gates or Field programmable gate arrays FPGAs, resulting in reduced complexity in space, time and power consumption.

8 Future Work

These are some points to enrich the ReCA research field:

- Using ECA with memory (ECAM) in the reservoir to benefit from the effect of large distributed representation provided by ECAM rules, and even using 2-dimensional CA, e.g., the game of life, which has universal computational properties.
- Other training methods can be used in the read-out stage rather than linear regression, e.g. logistic regression, support vector machines SVMs or ... to augment the model computational power.
- Adding Attention mechanisms to allow ReCA to focus on a specific part of the input (supporting facts), enabling ReCA to solve weakly supervised bAbI tasks.
- Using LLMs to solve the 20 QA bAbI tasks and compare their results with ReCA results.
- Translate the 20 QA bAbI tasks into Arabic to create an Arabic QA dataset, then solve them using ReCA.

References

- [1] Glover, T. E., Osipov, E., & Nichele, S. (2024). *On when is Reservoir Computing with Cellular Automata Beneficial?* arXiv.
<https://doi.org/10.48550/arXiv.2407.09501>
- [2] Atiya, A. F., & Parlos, A. G., (2000). New results on recurrent network training: Unifying the algorithms and accelerating convergence, *IEEE Transactions on Neural Networks*, 11, 697-709.
- [3] Glover, T. E., Lind, P., Yazidi, A., Osipov, E., & Nichele, S. (2023). Investigating rules and parameters of reservoir computing with elementary cellular automata, with a criticism of rule 90 and the five-bit memory benchmark. *Complex Systems*, 32(3), 309–351.
<https://doi.org/10.25088/ComplexSystems.32.3.309>
- [4] Margem, M., & Yilmaz, O., (2016). How much computation and distributedness is needed in sequence learning tasks? In *Artificial General Intelligence, AGI-16, Lecture Notes in Computer Science*, Springer, 9782, 274-283.
- [5] Uragami, D., & Gunji, Y.-P. (2022). Universal criticality in reservoir computing using asynchronous cellular automata. *Complex Systems*, 31(1), 103–121.
<https://doi.org/10.25088/ComplexSystems.31.1.103>
- [6] Pontes-Filho, S., Nichele, S., & Lepperød, M. (2025). Reservoir computing with evolved critical neural cellular automata. *Proceedings of the Artificial Life Conference 2025*. MIT Press. <https://doi.org/10.1162/isal.a.832>
- [7] Bengio, Y., Simard, P., & Frasconi, P., (1994). Learning Long-Term Dependencies with Gradient Descent Is Difficult, *IEEE Transactions on Neural Networks*, 5(2), 157-166. doi:10.1109/72.279181.
- [8] Hochreiter, S., & Schmidhuber, J., (1997). Long short-term memory Neural computation, 9(8), 1735-1780.
- [9] Doya, K., (1992). Bifurcations in the learning of recurrent neural networks, In *IEEE International Symposium on Circuits and Systems*, 2777–2780.

-
- [10] Margem, M., & Gedik, O. S., (2019). Reservoir Computing Based on Cellular Automata (ReCA) in Sequence Learning, *Journal of Cellular Automata*. 14(1-2), 153-170.
- [11] Jaeger, H., (2001). The 'Echo State' Approach to Analyzing and Training Recurrent Neural Networks—with an Erratum Note, GMD Technical Report, 148:34, Bonn, Germany: German National Research Center for Information Technology.
- [12] Maass W., Natschläger T., & Markram H., (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations, *Neural computation*., 14(11), 2531–2560.
- [13] Cisneros, H., Sivic, J., & Mikolov, T. (2022). Benchmarking learning efficiency in deep reservoir computing. *Proceedings of the 1st Conference on Lifelong Learning Agents (CoLLAs 2022)*, 532–547. <https://doi.org/10.48550/arXiv.2210.02549>
- [14] Lukoševičius, M., & Jaeger, H., (2009). Reservoir computing approaches to recurrent neural network training, *Computer Science Review*, 3(3), 127-149.
- [15] Lukoševičius, M., (201). A Practical Guide to Applying Echo State Networks, In *Neural Networks: Tricks of the Trade*, Lecture Notes in Computer Science, 659-686, Springer,
- [16] Jaeger, H., (2002). A tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the 'echo state network' approach, Technical report GMD report 159, German National Research Center for Information Technology.
- [17] Jaeger, H., & Hass, H., (2004). Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless telecommunication”, *Science*, 304, 78-80.
- [18] Bush, K., & Anderson, C., (2005). Modeling reward functions for incomplete state representations via echo state networks, In *Proceedings of the International Joint Conference on Neural Networks*, Montreal, Quebec, 2005.
- [19] Skowronski, M. D., & Harris, J. G., (2006). Minimum mean squared error time series classification using an echo state network prediction model, In *IEEE International Symposium on Circuits and Systems*, Island of Kos, Greece, 3153-3156.
- [20] Tong, M. H., Bicket, A. D., Christiansen, E. M., & Cottrell, G. W., (2007). Learning grammatical structure with echo state network, *Neural Networks*, 20, 424-432.
- [21] Soh, H., & Demiris, Y., (2012). Iterative temporal learning and prediction with the sparse online echo state gaussian process, In *International Joint Conference on Neural Networks (IJCNN)*, 1-8, IEEE.
- [22] Jalalvand, A., Van Wallendael, G., & Van de Walle, R., (2015). Real-time reservoir computing network-based systems for detection tasks on visual contents, In *7th International Conference on Computational Intelligence, Communication Systems and Networks (CICSyN)*, 146-151, IEEE.
- [23] Adamatzky, A., (2001). *Computing in nonlinear media and automata collectives*”, CRC Press.
- [24] Fernando, C., & Sojakka, S., (2003). Pattern recognition in a bucket, In *European Conference on Artificial Life (ECAL 2003)*, 588-597, Springer.
doi:10.1007/978-3-540-39432-7_63.
- [25] Dai, X., (2004). Genetic Regulatory Systems Modeled by Recurrent Neural Network, In *Proceedings, Part II, Advances in Neural Networks: International Symposium on Neural Networks (ISNN 2004)*, 519-524, Springer. doi:10.1007/978-3-540-28648-6_83.
- [26] Jones, B., Stekel, D., Rowe, J., & Fernando, C., (2007). Is There a Liquid State Machine in the Bacterium *Escherichia coli*?”, In *Proceedings of the IEEE Symposium on Artificial Life 2007 (ALIFE'07)*, 187-191, IEEE. doi:10.1109/ALIFE.2007.367795.
- [27] Paquot Y., Duport, F., Smerieri, A., Dambre, J., Schrauwen, B., Haelterman, M., & Massar, S., (2012). Optoelectronic Reservoir Computing, *Scientific Reports in Nature*, 2, 287 doi:10.1038/srep00287
- [28] Larger, L., Soriano, M. C., Brunner, D., Appeltant, L., Gutiérrez, J. M., Pesquera, L., Mirasso, C. R., & Fischer, I., (2012). Photonic Information Processing beyond Turing: An Optoelectronic

- Implementation of Reservoir Computing, *Optics Express*, 20(3), 3241-3249. doi:10.1364/OE.20.003241.
- [29] Ortín, S., Soriano, M. C., Pesquera, L., Brunner, D., San-Martín, D., Fischer, I., Mirasso C. R., & Gutiérrez, J. M., (2015). A unified framework for reservoir computing and extreme learning machines based on a single time-delayed neuron”, *Scientific reports*, 5, 14945.
- [30] Snyder, D., Goudarzi, A., & Teuscher, C., (2013). Computational Capabilities of Random Automata Networks for Reservoir Computing, *Physical Review E*, 87(4), 042808. doi:10.1103/PhysRevE.87.042808.
- [31] Jähren, R. (2022). *Comparison and benchmarking of reservoir computing using cellular automata and random boolean networks as substrates* [Master's thesis, OsloMet—Oslo Metropolitan University]. ODA Open Digital Archive.
- [32] Glover, T. E., Jähren, R., Martinuzzi, F., Lind, P. G., & Nichele, S. (2024). A sensitivity analysis of cellular automata and heterogeneous topology networks: Partially-local cellular automata and homogeneous homogeneous random boolean networks. *International Journal of Parallel, Emergent and Distributed Systems*, 39(1), 59–99. <https://doi.org/10.1080/17445760.2024.2396334>
- [33] Dale, M., Miller, J. F., Stepney, S., & Trefzer, M. A., (2016). Evolving Carbon Nanotube Reservoir Computers, In *Proceedings of International Conference on Unconventional Computation and Natural Computation (UCNC 2016)*, 49-61, Springer International Publishing. doi:10.1007/978-3-319-41312-9_5.
- [34] Dale, M., Miller, J. F., & Stepney, S., (2017). Reservoir Computing as a Model for in-Materio Computing, *Advances in Unconventional Computing: Volume 1: Theory* (Adamatzky, A., ed.), 533-571, Springer International Publishing. doi:10.1007/978-3-319-33924-5_22.
- [35] Goudarzi, A., Lakin, M.R., & Stefanovic, D., (2013). DNA reservoir computing: a novel molecular computing approach, In *International Workshop on DNA Based Computers*, 76-89, Springer.
- [36] Yamane, T., Katayama, Y., Nakane, R., Tanaka, G., & Nakano, D., (2015). Wave based reservoir computing by synchronization of coupled oscillators, In *International Conference on Neural Information Processing*, 198-205, Springer.
- [37] Coulombe, J.C., York, M.C., & Sylvestre, J., (2017). Computing with networks of nonlinear mechanical oscillators, *PloS one*, 12(6), e0178663.
- [38] Suzuki, Y., Gao, Q., Pradel, K. C., Yasuoka, K., & Yamamoto, N. (2022). Natural quantum reservoir computing for temporal information processing. *Scientific Reports*, 12, Article 1353. <https://doi.org/10.1038/s41598-022-05061-w>
- [39] Daniels, R. K., Mallinson, J. B., Heywood, Z. E., Bones, P. J., Arnold, M. D., & Brown, S. A. (2022). Reservoir computing with 3D nanowire networks. *Neural Networks*, 154, 122–130. doi:10.1016/j.neunet.2022.07.001
- [40] Choi, S., Shin, J., Park, G., Eo, J. S., Jang, J., Yang, J. J., & Wang, G. (2024). 3D-integrated multilayered physical reservoir array for learning and forecasting time-series information. *Nature Communications*, 15, Article 2044. <https://doi.org/10.1038/s41467-024-46323-7>
- [41] Singh, A., Choi, S., Wang, G., Daimari, M., & Lee, B. G. (2025). Analysis and fully memristor-based reservoir computing for temporal data classification. *Neural Networks*, 182, Article 106925. <https://doi.org/10.1016/j.neunet.2024.106925>
- [42] Guo, Y., Miao, C., Wang, X., Liu, X., Ma, C., Wang, L., Duan, S., Duan, W., & Li, H. (2025). Symbolic reservoir computing within memristive crossbar arrays as a cellular automata. *Advanced Intelligent Systems*, 7(12), Article 70163. <https://doi.org/10.1002/aisy.70163>
- [43] Kim, S., Shin, D. H., Choi, W., Cheong, S., Shim, S. K., Lee, S. H., Han, J., Jang, Y. H., Son, K., Ghenzi, N., & Hwang, C. S. (2026). Spatiotemporal reservoir computing with a reconfigurable

- multifunctional memristor array. *Advanced Materials*, 38(3), Article 10635. <https://doi.org/10.1002/adma.202510635>
- [44] Ghafoor, F., Kim, H., Zhang, H., Ghafoor, B., Lee, M., Shi, T., & Kim, D. (2026). Energy-efficient reservoir computing with 10×10 crossbar array memristor for high performance multitask recognition. *Advanced Composites and Hybrid Materials*, 9, Article 56. <https://doi.org/10.1007/s42114-025-01566-w>
- [45] Park, S., Kim, H., & Jang, H. W. (2026). Device-level implementation of reservoir computing with memristors. *Advanced Intelligent Systems*, 8(5), Article 202500843. <https://doi.org/10.1002/aisy.202500843>
- [46] Yilmaz, O., (2015). Machine Learning using Cellular Automata based Feature Expansion and Reservoir Computing, *Journal of Cellular Automata*, 10(5-6), 435-472.
- [47] Nichele, S., & Gundersen, M. S., (2017). Reservoir Computing Using Non-Uniform Binary Cellular Automata, *Complex Systems*, 26(3), 225-245, Complex Systems Publications Inc.
- [48] Nichele, S., & Molund, A., (2017). Deep learning with cellular automaton-based reservoir computing, *Complex Systems*, 26(4), 319-339, Complex Systems Publications Inc.
- [49] McDonald, N., (2017). Reservoir Computing & Extreme Learning Machines using Pairs of Cellular Automata Rules, In *International Joint Conference on Neural Networks (IJCNN)*, USA, 88, 2429-2436.
- [50] Margem, M., & Gedik, O. S., (2019). Feed-forward vs. Recurrent Architecture and Local vs. Cellular Automata Distributed Representation Based Reservoir Computing in Sequence Memory Learning, under review in *Artificial Intelligence Review (AIRE)*.
- [51] Kantic, J., Legl, F. C., Stechele, W., & Hermann, J. (2024). ReLiCADA: Reservoir computing using linear cellular automata design algorithm. *Complex & Intelligent Systems*, 10(3), 3593–3616. <https://doi.org/10.1007/s40747-023-01330-x>
- [52] Barandiaran, M., & Stovold, J. (2025). Growing reservoirs with developmental graph cellular automata. *Proceedings of the Artificial Life Conference 2025*. MIT Press. <https://doi.org/10.1162/isal.a.854>
- [53] Nakada, K., Takahashi, K., Uragami, D., Alonso-Sanz, R., Adamatzky, A., & Nishiyama, Y. (2026). Generalization performance of reservoir computing implemented by asynchronously tuned elementary cellular automaton on parity task. *Artificial Life and Robotics*, 31(1), 77–82. <https://doi.org/10.1007/s10015-025-01045-x>
- [54] Glover, T. E., Lind, P., Yazidi, A., Osipov, E., & Nichele, S. (2021). The dynamical landscape of reservoir computing with elementary cellular automata. In *Proceedings of the ALIFE 2021: The 2021 Conference on Artificial Life* (p. 102). MIT Press. https://doi.org/10.1162/isal_a_00440.
- [55] Tanaka, G., Yamane, T., Héroux, J. B., Nakane, R., Kanazawa, N., Takeda, S., Numata, H., Nakano, D., & Hirose, A., (2019). Recent advances in physical reservoir computing: A review, *Neural Networks*. doi: 10.1016/j.neunet.2019.03.005.
- [56] Chen, X., Araujo, F. A., Riou, M., Torrejon, J., Ravelosona, D., Kang, W., Zhao, W., Grollier, J., & Querlioz, D. (2022). Forecasting the outcome of spintronic experiments with Neural Ordinary Differential Equations. *Nature Communications*, 13, Article 1016. <https://doi.org/10.1038/s41467-022-28571-7>
- [57] Abe, Y., Nakada, K., Hagiwara, N., Suzuki, E., Suda, K., Mochizuki, S.-i., Terasaki, Y., Sasaki, T., & Asai, T. (2024). Highly-integrable analogue reservoir circuits based on a simple cycle architecture. *Scientific Reports*, 14, Article 13798. <https://doi.org/10.1038/s41598-024-61880-z>
- [58] Kong, L.-W., Brewer, G. A., & Lai, Y.-C. (2024). Reservoir-computing based associative memory and itinerancy for complex dynamical attractors. *Nature Communications*, 15, Article 4204. <https://doi.org/10.1038/s41467-024-49190-4>
- [59] Li, X., Zhu, Q., Zhao, C., Duan, X., Zhao, B., Zhang, X., Ma, H., Sun, J., & Lin, W. (2024). Higher-order Granger reservoir computing: simultaneously achieving scalable complex structures

- inference and accurate dynamics prediction. *Nature Communications*, 15, Article 2506. <https://doi.org/10.1038/s41467-024-46852-1>
- [60] Jin, P., Cao, R., Gan, Z., & Leng, S. (2026). Time-delayed spiking reservoir computing enables efficient time series prediction. *Advanced Intelligent Systems*, 8(7), Article 70421. <https://doi.org/10.1002/aisy.70421>
- [61] Hadaeghi, F., He, X., Jaeger, H., (2017). Unconventional Information Processing Systems, Novel Hardware: A Tour D'Horizon, IRC-Library, Information Resource Center der Jacobs University Bremen.
- [62] Yan, M., Huang, C., Bienstman, P., Tino, P., Lin, W., & Sun, J. (2024). Emerging opportunities and challenges for the future of reservoir computing. *Nature Communications*, 15, Article 2056. <https://doi.org/10.1038/s41467-024-45187-1>
- [63] Liu, J., Feng, G., Li, W., Hao, S., Han, S., Zhu, Q., Tian, B., Duan, C., & Chu, J. (2025). Physical reservoir computing for Edge AI applications. *The Innovation Materials*, 3(2), Article 100127. <https://doi.org/10.59717/j.xinn-mater.2025.100127>
- [64] Zolfagharinejad, M., Alegre-Ibarra, U., Chen, T., Kinge, S., & van der Wiel, W. G. (2024). Brain-inspired computing systems: a systematic literature review. *The European Physical Journal B*, 97, Article 93. <https://doi.org/10.1140/epjb/s10051-024-00703-6>
- [65] Cestnik, R., & Martens, E. A. (2026). Next-generation reservoir computing for dynamical inference. *Chaos*, 36(1), Article 013101. <https://doi.org/10.1063/5.0302319>
- [66] Sipper, M., Tomassini, M., & Capcarrere, M. S. (1998). Evolving asynchronous and scalable non-uniform cellular automata. In *Artificial Neural Nets and Genetic Algorithms* (pp. 66–70). Springer. https://doi.org/10.1007/978-3-7091-6492-1_15
- [67] Wolfram, S. (2002). *A new kind of science*. Wolfram Media.
- [68] Pal Chaudhuri, P., Roy Chowdhury, D., Nandi, S., & Chatterjee, S. (1997). *Additive cellular automata theory and applications* (Vol. 1). IEEE Computer Society Press.
- [69] Martínez, G., Seck-Tuoh-Mora J., & Zenil H., (2013). Computation and Universality: Class IV versus Class III Cellular Automata, *Journal of Cellular Automata*, 7(5-6), 393-430.
- [70] Martínez, G. J., Seck-Tuoh-Mora, J. C., & Zenil, H. (2013). Wolfram's classification and computation in cellular automata classes III and IV, Irreducibility and Computational Equivalence, Zenil, H. (ed.), Chapter 17, 237-259, Springer.
- [71] Cook, M., (2004). Universality in elementary cellular automata, *Complex Systems*, 15(1), 1-40.
- [72] Shackleford, B., Tanaka, M., Carter, R. J., & Snider, G. (2002). FPGA implementation of neighborhood-of-four cellular automata random number generators, In *Proceedings of the 2002 ACM/SIGDA tenth international symposium on Field-programmable gate arrays*, 106-112, ACM.
- [73] Gobron, S., Devillard, F., & Heit, B. (2007). Retina simulation using cellular automata and GPU programming", *Machine Vision and Applications*, 18(6), 331-342.
- [74] Weston, J., Chopra, S., & Bordes, A., (2015). Memory Networks, *International Conference on Learning Representations (ICLR)*, USA.
- [75] Graves, A., Wayne, G., & Danihelka, I., (2014). Neural Turing machines, *arXiv preprint arXiv:1410.540*.
- [76] Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., et al. (2016). Hybrid computing using a neural network with dynamic external memory, *Nature*, 538(7626), 471.
- [77] Collier, M., & Beel, J., (2018). *Implementing Neural Turing Machines*, *Artificial Neural Networks and Machine Learning – ICANN 2018*, Lecture Notes in Computer Science, 11141, 94-104, Springer International Publishing. [doi:10.1007/978-3-030-01424-7_10](https://doi.org/10.1007/978-3-030-01424-7_10), ISBN 9783030014230

-
- [78] Weston, J., Bordes, A., Chopra, S., Rush, A. M., van Merriënboer, B., Joulin, A., & Mikolov, T., (2015). Towards AI-Complete Question Answering: A set of Prerequisite Toy Tasks, arXiv preprint arXiv: 1502.05698v10.
- [79] Kumar, A., Ondruska, F., Iyyer, M., Bradbury, J., Gulrajani I., Zhong, V., Paulus, R., & Socher, R., (2016). Ask Me Anything: Dynamic Memory Networks for Natural Language Processing, In Proceedings of the 33rd International Conference on Machine Learning in PMLR., 48, 1378-1387.
- [80] Lee, M., He, X., Yih, W. T., Gao, J., Deng, L., & Smolensky, P., (2016). Reasoning in Vector Space: An Exploratory Study of Question Answering, arXiv preprint arXiv: 1511.06426v4.
- [81] Sukhbaatar, S., Weston, J., & Fergus, R., (2015). End-to-end memory networks, In Advances in neural information processing systems, 2440-2448.
- [82] Bahdanau, D., Cho, K., & Bengio, Y., (2014). Neural machine translation by jointly learning to align and translate, arXiv preprint arXiv:1409.0473.
- [83] Yu, Y., Zhang, W., Hang, C. W., Xiang, B., & Zhou, B., (2015). Empirical study on deep learning models for question answering, arXiv preprint arXiv:1510.07526.
- [84] Peng, B., Lu, Z., Li, H., & Wong, K. F., (2015). Towards neural network-based reasoning, arXiv preprint arXiv:1508.05508.
- [85] Smolensky, P., (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems, Artificial intelligence, 46(1-2), 159-216.
- [86] Pennington, J., Socher, R., & Manning, C., (2014). Glove: Global vectors for word representation, In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 1532-1543.
- [87] Karpagam, K., Madusudanan, K., & Saradha, A. (2020). Deep learning approaches for answer selection in question answering system for conversation agents. *ICTACT Journal on Soft Computing*, 10(2), 2057–2063. <https://doi.org/10.21917/ijsc.2020.0289>
- [88] Patel, R., Patel, P., Bhatt, C. M., & Rahevar, M. L. (2021). Question answering chatbot using memory networks. In V. Chkoniya (Ed.), *Handbook of research on applied data science and artificial intelligence in business and industry* (pp. 401–413). IGI Global Scientific Publishing. <https://doi.org/10.4018/978-1-7998-6985-6.ch019>
- [89] Kim, S., Nam, H., Kim, J., & Jung, K. (2021). Neural sequence-to-grid module for learning symbolic rules. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(9), 8163–8171. <https://doi.org/10.1609/aaai.v35i9.16994>
- [90] Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., & Kaiser, Ł. (2019). Universal transformers. *International Conference on Learning Representations (ICLR)*. <https://doi.org/10.48550/arXiv.1807.03819>
- [91] Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems* (Vol. 27, pp. 3104–3112). Curran Associates, Inc. <https://papers.nips.cc/paper/2014/hash/a14ac55a4f27472c5d894ec1c3c743d2-Abstract.html>
- [92] Mirzaee, R., Rajaby Faghihi, H., Ning, Q., & Kordjamshidi, P. (2021). SPARTQA: A textual question answering benchmark for spatial reasoning. In K. Toutanova et al. (Eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 4582–4598). Association for Computational Linguistics. <https://doi.org/10.18653/v1/2021.naacl-main.364>
- [93] Tiwari, A. P. (2025). A modular neurosymbolic framework for general-purpose reasoning: Bridging symbolic and deep learning for interpretable AI. *TechRxiv*. <https://doi.org/10.36227/techrxiv.175295182.20276969/v1>

- [94] Liang, Y., Wang, Y., Fang, H., Zhao, F., & Zeng, Y. (2025). A brain-inspired memory transformation based differentiable neural computer for reasoning-based question answering. *Frontiers in Artificial Intelligence*, 8, Article 1635932. <https://doi.org/10.3389/frai.2025.1635932>
- [95] Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., & Lillicrap, T. (2016). Meta-learning with memory-augmented neural networks. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)* (pp. 1842–1850). PMLR.
- [96] Lake, B. M., Ullman, T. D., Tenenbaum, J. B., & Gershman, S. J. (2017). Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40, Article e253. <https://doi.org/10.1017/S0140525X16001837>