Otto-Friedrich-University of Bamberg
Applied Computer Science

Seminar: AI - past, present, future

Differentiable neural computers - from inspiration to implementation

by

Katharina Weitz

28th February 2017

advised by
Professor Dr. Ute Schmid
Abstract

Information storage is an important research topic in both psychology and computer science. On one side, biological models of the human brain, especially the hippocampus, and psychological models of the working-memory of humans, inspired research in neural networks. On the other side, developments in computer science itself, like the long short-term memory (LSTM) and the Neural Turing Machine (NTM) lead to new inventions. One of these is the differentiable neural computer with an external memory (DNC) which combines psychological and biological aspects, as well as technical innovations. First experiments with traversal, shortest path and inference tasks showed that the DNC improved accuracy in comparison with LSTM and NTM a lot.
# Contents

1 Introduction 3

2 Inspiration and implementation of a neural network with external memory 3
   2.1 Inspiration .................................................. 3
       2.1.1 Hippocampus ......................................... 3
       2.1.2 Working memory model of the human mind ....... 4
   2.2 Implementation ............................................... 5
       2.2.1 Long short-term memory .............................. 5
       2.2.2 Neural Turing Machine ............................... 6

3 Differentiable neural computer with external memory 7
   3.1 Architecture ............................................... 7
   3.2 Interaction of read/write heads and memory .......... 9
   3.3 Applications ............................................... 10
   3.4 Results of LSTM, NTM & DNC ......................... 12

4 Summary 14

References 17
1 Introduction

The concept of memory is one of the most important concepts in psychological research. It influences a range of research fields like learning, performance, attention, perception and decision making. The capacity of working memory in humans is limited (Miller, 1956), which has a huge influence on performance in cognitive tasks. In computer science, invention of recurrent neural networks (RNNs) is inspired by psychology and biology. RNNs have the ability to learn and perform complicated transformations (Graves et al., 2014). In general, it was proven that RNNs are Turing-Complete (Siegelmann and Sontag, 1995). Therefore they are able to solve every calculation a computer is able to solve, but only if RNNs have the ability to store and allocate information in memory, like in a von-Neumann architecture in conventional computers (von Neumann, 1945). In practical, the combination of RNNs with memory is not as simple as it may seem. Recently, Graves et al. (2016) introduced a differentiable neural computer (DNC) which solves this gap between theory and practice. In the following, the inspiration for the DNC by psychological and biological models is described. After that, the first implementations of RNNs with memory is explained. Finally, the DNC itself will be described and the first performance results of the DNC in different tasks are shown.

2 Inspiration and implementation of a neural network with external memory

2.1 Inspiration

In this section the biological and psychological inspiration for neural network with the ability to memorize information is described. On biological side, a part of the human brain, the hippocampus is introduced. On psychological side, the working memory model of Atkinson and Shiffrin (1968) is explained.

2.1.1 Hippocampus

The working memory in the human mind is a neurocognitive system which gives us the ability to maintain and manipulate information for a short time (Güntürkün, 2012). The hippocampus is a part of the human forebrain, lying below the cerebral cortex (Gluck and Myers, 2001) (see Figure 1). It is located in the inner side of the temporal lobe (see figure 2).
The hippocampus plays a critical role in human memory and learning (Gluck and Myers, 2001), especially for transferring information from the short-term memory into the long-term memory. The hippocampal system has the ability to learn new information rapidly by using controlled processes to bind information together (Hazy et al., 2006). Much of the knowledge about the relationship between learning processes and the hippocampus was gained from people who had damages in the hippocampal part of their brains (Gluck and Myers, 2001).

2.1.2 Working memory model of the human mind

There are different theories about the sequence of controlled processes in the hippocampus. In psychology, the model of working memory explains the short-term manipulation of information (Graves et al., 2014). The multi-store model of Atkinson and Shiffrin (1968) describes, how storing of information in the human mind works. Three parts are important in their model: sensory register, short-term store and long-term store.

First, information is detected by the sense organs and then enters the sensory register. There, a stimulus, often a visual one, is registered and the information is stored. The
second part is the short-term store. If the human attends to the information, it enters the short-term store. Atkinson and Shiffrin (1968) called this store the ”working memory” of the human. To transfer the information to the long-term store, a rehearsal in the short-term store is needed; rehearsal, in this context, means the repetition of information. If the rehearsal does not occur, information in the short-term memory can only be stored for a period of about 15-30 seconds. After that time, information is forgotten, lost from the short-term memory by displacement or decay. The third part is the long-term memory. Here, information can be stored for longer time and the storage is relatively permanent.

An easy example to understand the theoretic concept, is the situation in which someone gives another person his or her telephone number. When the person has nothing to write the number on, he or she has to repeat it for some time, before the number can be memorized and therefore be stored in long-term memory. Without the rehearsal, the number normally would not be remembered.

2.2 Implementation

The basic structure of the human mind is the idea of a system which can store information and use it to solve tasks. The question is, whether and how this idea can be transferred to the field of neural networks. Judd (1990) described one of the biggest problems of neural networks: they are not able to generalize, which means to extrapolate learned knowledge to other domains. For this step, a neural network would need some kind of memorization. For Judd (1990) the memorization was ”intractable” for a neural network, but nowadays this perspective has changed. This section will give a short overview about two of the first attempts, the long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and the Neural Turing Machine (NTM) (Graves et al., 2014), two approaches to combine the power of a neural network with the ability to store information.

2.2.1 Long short-term memory

Before the LSTM was invented, the different approaches which tried to store information in a recurrent network, had huge error backflow problems (Hochreiter and Schmidhuber, 1997). The usage of conventional backpropagation through time as described by Williams and Zipser (1995), or real-time recurrent learning as described by Robinson and Fallside (1987), handle occurring errors during the computation, which leads to blowing up or vanishing the system. The reason for blowing up and vanishing is the temporal evolution of the error which exponentially depends on the size of the weights (Hochreiter, 1991). The higher the weights, the higher the computational costs for handling the error. The LSTM was designed to solve this backflow problem. For this, the LSTM uses memory cells and gate units. The memory cells guarantee a constant error flow. The gate units, consisting of input and output gates, control the access to the memory cells. Figure 3 shows such a memory cell with input gate \( y_i^{inj} \) and output gate \( y_i^{outj} \). The self-recurrent connection (with weight 1) indicates feedback with a delay of one time step. This idea leads to a ”perfect integrator” (Seung, 1998), which can be expressed in the following
equation: \( x(t+1) = x(t) + i(t) \), where \( i(t) \) is an input to the system. With the use of the gates, the equation can be extended to \( x(t+1) = x(t) + g(\text{context})i(t) \) (Graves et al., 2014). The \( g(\text{context}) \) is the gate which depends on a context - the input or output gate - and gives the LSTM the ability to store information selectively. This represents the attending part of the short-term memory model from Atkinson and Shiffrin (1968).

Figure 3: Architecture of a memory cell (Figure 1 from Hochreiter and Schmidhuber (1997)).

### 2.2.2 Neural Turing Machine

With the LSTM it is possible to implement a kind of internal memory like a short-term memory. Graves et al. (2014) extended this approach and invented the Neural Turing Machine (NTM). The NTM includes two important parts: a neural network (controller) which interacts with the external world via input and output vectors, and reading/writing heads which can be used by the controller to read/write information from a \( N \times M \) memory matrix. \( N \) is the number of memory locations, \( M \) the vector size of each location (see Figure 4).

Figure 4: High level illustration of the NTM architecture (Figure 1 from Graves et al. (2014)).
Reading from memory is realized by a reading vector of the form:

\[ r_t \leftarrow \sum_i w_t(i)M_t(i) \]

with \( M_t(i) \), the row-vectors in memory and \( w_t(i) \), \( N \) elements of the weighting vector \( w_t \). For writing, the NTM uses two vectors: first an erase vector \( (e_t) \) of the form

\[ \tilde{M}_t(i) \leftarrow M_{t-1}(i)[1 - w_t(i)e_t] \]

where \( M_{t-1} \) is the memory vector of the previous time step. Second, an add vector \( (a_t) \) of the form

\[ M_t(i) \leftarrow \tilde{M}_t(i) + w_t(i)a_t \]

is used. The erase vector only removes the content from the memory when both the weighting \( w_t \) and the erase element \( e_t \) are one. Otherwise the content of the memory stays unchanged. The weights for the reading and writing operations are computed by two addressing mechanisms: a so-called content-based addressing and a location-based addressing (Graves et al., 2014). In Content-based addressing information with similar content are stored close together. Location-based addressing is the conventional type of saving data. Here, information is stored at different locations in the memory. Graves et al. (2014) described the example of storing the content of a \( x \) and \( y \) variable for a calculation as a typical location-based addressing. For calculation (e.g. \( x + y \)) the information of the variables are important but not that \( x \) and \( y \) are stored at the same location in memory. Both address types are important, but also complementary. There are three types of complementation: weightings can be chosen by the content-based addressing without involving location-based addressing. It is also possible that the weighting calculated by the content-based addressing system can be chosen and then be relocated. The third type is a rotated weighting from the previous time step without involving the content-based addressing. Rotating is useful when a weighting is not changing the location. A shift weighting \( s_t \) then takes care to shift the focus of the weighting to the next location.

3 Differentiable neural computer with external memory

In this section, a new machine learning model of an artificial neural network, the ”differentiable neural computer” (DNC) (Graves et al., 2016) is introduced. This model combines the ideas of neural networks with the idea of memory to write and read from.

3.1 Architecture

The architecture of the model depends on three key components, the neural network itself, the read/write heads and the external memory (see Figure 5).
Neural Network. The controller can be any neural network. In the work of Graves et al. (2016) a variant of the LSTM architecture is used as controller. The controller interacts with the external world via input and output vectors. The difference between this controller and conventional neural networks is, that the controller also interacts with a memory matrix. At every time step $t$, the controller receives an input vector $x_t \in \mathbb{R}^X$ from the environment and sends an output vector $y_t \in \mathbb{R}^Y$. The output vector can be used for a predictive distribution (supervised learning) or an action distribution (reinforcement learning). The interaction with external memory is performed through a set of $R$ read vectors, sent from the memory matrix $M_{t-1} \in \mathbb{R}^{N \times W}$ from the previous time step.

Writing and Reading Heads. To get access to memory, the DNC uses similar mechanisms like the NTM does. These allow the DNC to define distributions (locations) in the $N \times W$ memory matrix $M$. Every location has a weighting. A weighting of a location is a vector of non-negative numbers and represents the degree to which each location is involved in a read or write operation. The read operation needs a read vector $r$ which is returned by a read weighting $w^r$ in the form

$$r^i_t = M^T_t w^r_i$$

It holds the information about how much each location of the memory is involved in a read operation. Similarly, the write operation uses a write weighting $w^w$. The write operation happens in two steps: First, it erases the data on the specific location with an erase vector $e$ and then adds new data with a write vector $v$. The reading/writing heads use three various forms of differentiable attention which are called content lookup, transition record and allocating memory.

External memory. The external memory in the DNC is important to store information over long time. Graves et al. (2016) call the memory ”external” because the controller is independent of the memory. Like the NTM, the memory of the DNC is used by content-based addressing and location-based addressing. For all content lookup
operations the content-based addressing is done with the function:

\[ C(M, k, \beta)[i] = \frac{\exp D(k, M[i], \beta)}{\sum_j \exp D(k, M[j], \beta)} \]

where \( k \in \mathbb{R}^W \) is a lookup key, \( \beta \in [1, \infty) \) is a scalar which represents the strength of the key and \( D \) stands for the cosine similarity:

\[ D(u, v) = \frac{u \cdot v}{||u|| ||v||} \]

In contrast to the NTM, the DNC controller is able to allocate and release memory dynamically. For this, a linked list is used. The linked list makes it possible to remove or add addresses. The controller sends out a set of free gates \( f_t^i \) for every read head which decides whether a read location can be freed. The retention vector \( \psi_t \in [0, 1]^N \) shows how much the location will not be released by freed gates. It is defined as

\[ \psi_t = \prod_{i=1}^{R} (1 - f_t^i w_t^{r,i}) \]

The retention vector \( \psi_t \) is important to calculate the usage vector

\[ u_t = (u_{t-1} - w_t^{w,i} - u_{t-1} \odot w_t^{w,i}) \odot \psi_t \]

The usage vector tells the controller if the memory reached its capacity. Then the controller have to free used locations first before allocate memory.

Another improvement in comparison to the NTM is the temporal memory linkage. The memory system described so far does not have the ability to know anything about the order of written or read information in the memory. For applications where a sequence of information should be recorded and retrieved in order (e.g. path finding tasks) this capability is useful. The transition record with \( L[i,j] \) makes this possible.

### 3.2 Interaction of read/write heads and memory

To give the read and write heads the ability to interact with the memory, three mechanisms are used: content lookup, transition record and allocating memory.

**Content lookup.** Content lookup answers the question: "Where should I write and read?" Content lookup is used by read and write heads for associative recall and for changing an existing vector in memory by finding already written locations to edit them. The write and read heads get a key vector, calculated by the controller, which is compared to the content of each location in the memory according to a similarity measure, the cosine similarity. The similarity scores determine a weighting that can be used by the read and write heads.

**Transition record.** The Transition record answers the question: "Where can I find the information?" The transition record gives the DNC the native ability to recover
sequences in the order in which it wrote them, which means that the read heads can read out locations either forwards or backwards in the order they were written. For this, the reading and writing heads have to record transitions between continuously written locations in an $N \times N$ temporal link matrix $L$, where $L[i, j]$ is close to 1 if $i$ is the next location written after $j$ and close to 0 otherwise. For every weighting $w$ the operation $Lw$ shifts the focus forwards, defined by

$$f_t^i = L_i w_{t-1}^r$$

The operation $L^\top w$ shifts the focus backwards, defined by

$$b_t^i = L_i^\top w_{t-1}^r$$

**Allocating memory.** The allocating memory mechanism answers the question: “Which part of the memory is used/not used?” The third form of attention allocates memory for writing and is a new feature in the DNC architecture; the NTM architecture does not have this. The usage vector provides which locations have been used so far and is represented by a number between 0 and 1 and a weighting that finds out unused locations. The idea is that the usage can be increased automatically with each writing to a location and can be decreased after each reading. It allows the controller to release memory that is no longer required. This allocating mechanism is independent of the size and contents of the memory. Hence, the DNC can be trained to solve a task using one size of memory and if a certain threshold is reached, the system can expand the memory without retraining.

### 3.3 Applications

To investigate the performance of the DNC, Graves et al. (2016) tested it against the LSTM and the NTM in three different kind of tasks: synthetic question answering, graph experiments and block puzzle experiment.

**Synthetic question answering experiments.** In the first experiment - a logical reasoning task - Graves et al. (2016) used the bAbI dataset which is publicly available and includes 20 tasks for testing text understanding (Weston et al., 2015). The dataset includes short stories in natural language and questions about them. Every story was read as a distinct sequence and was presented to the controller in form of word-vectors (one word at each time step). Every sentence in the sequence was divided by a full stop character, every question ended with a question mark character and as many dash characters as there were words in the answer. Graves et al. (2016) illustrate the following example:

”mary journeyed to the kitchen. mary moved to the bedroom. john went back to the hallway. john picked up the milk there. what is john carrying? - john travelled to the garden. john journeyed to the bedroom. what is john carrying? - mary travelled to the bathroom. john took the apple there. what is john carrying? - -”
The answers of this example are: "milk", "milk", "milk, apple". 10% of the used stories were excluded into a validation set to test the trained network. The rest of the stories were used in random order to train the network.

**Graph experiments.** The second task is a supervised learning task performing synthetic reasoning on randomly generated graphs. The controller was trained with graphs and queries whose complexity increased before the test. Graves et al. (2016) used three kinds of tasks: traversal, shortest path and inference.

For the traversal task and the shortest path task, they used a map of the London Underground and triples with a source label (from), a destination label (to) and an edge label (edge) (see Figure 6). In the traversal task, seven step traversals were tested (random walks from random start nodes) with given source and edge labels but without a destination (e.g. "BondSt., Central", see Figure 6). The input triples for the rest of the task include only edge labels. In the shortest path walk, a start and an end node were given to the controller. After that, the controller had a 10-time-step planning phase to determine the shortest path. In the answering phase, the controller created a sequence of triples corresponding to a path. By contrast to the traversal task, the controller got input triples depending on the decision of the previous time step.

For the inference task, a family tree representation was used. The researchers pre-defined 400 relation labels that stood as abbreviations for sequences of up to five connected edge labels. The controller was given a start node and a relation. In the planning phase, the network had to infer the relation sequence from its label and perform an implicit traversal along that sequence to reach the destination. The relation sequences were never presented to the network. It had to conclude the relations from the queries and targets.
The last task, a reinforcement task, used the Winograd’s SHRDLU inspired block puzzle (Winograd, 1971). The idea of block puzzles is a world with a number of movable blocks in it, in which an agent can execute instructions of a user. As a view of the world, a "board" (a $S \times S$ grid) was given as input to the agent. He had the ability to move top blocks from one stack to another. For the block puzzle experiment, two DNC networks were used: a policy network for selecting an action and a value network for the estimation of a reward. After generating a start configuration of the block world, the agent was given different goals. Each goal was represented by block pairs. After all goals were presented, a single goal label was chosen and the agent had the mission to reach that goal.

3.4 Results of LSTM, NTM & DNC

To compare the DNC with the LSTM & NTM, Graves et al. (2016) use the applications described in the section before. The DNC achieved the best results in all experiments.

**Synthetic question answering experiments.** The DNC performed best in comparison with LSTM and NTM and even in comparison with best previous trained result
by a memory network, called MemN2N, which had an accuracy of 7.5% and 6 failed tasks (see Table 1).

Table 1

*Results of LSTM, NTM and DNC synthetic question answering experiments (Extracted from Graves et al. (2016))*

<table>
<thead>
<tr>
<th>Architectures</th>
<th>LSTM</th>
<th>NTM</th>
<th>DNC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error rate in %</td>
<td>25.5</td>
<td>20.1</td>
<td>4.3</td>
</tr>
<tr>
<td>Number of failed tasks</td>
<td>15</td>
<td>16</td>
<td>2</td>
</tr>
</tbody>
</table>

**Graph experiments.** As shown in Table 2 the LSTM completed the first and easiest part of the graph experiment, the traversal task, not successfully. The DNC solved all graph task experiments successfully. For the NTM no information was given in the paper.

Table 2

*Results of LSTM, NTM and DNC in graph experiments (Extracted from Graves et al. (2016))*

<table>
<thead>
<tr>
<th>Architectures</th>
<th>LSTM</th>
<th>NTM</th>
<th>DNC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completed levels</td>
<td>none</td>
<td>?</td>
<td>5</td>
</tr>
<tr>
<td>Accuracy in %</td>
<td></td>
<td>37</td>
<td>98.8</td>
</tr>
<tr>
<td>Traversal</td>
<td></td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>Shortest path</td>
<td>-</td>
<td>?</td>
<td>55.3</td>
</tr>
<tr>
<td>Interference</td>
<td>-</td>
<td>?</td>
<td>81.8</td>
</tr>
<tr>
<td>Training examples</td>
<td>2m</td>
<td>?</td>
<td>1m</td>
</tr>
</tbody>
</table>

*Note.* The average accuracy is shown.

**Block puzzle experiments.** In Figure 7 results of 20 replicated training runs with LSTM and DNC are displayed. Only the DNC finished the whole learning curriculum.
4 Summary

Biological and psychological research influenced the research on neural networks and memory usage. The working memory model of the human mind from Atkinson and Shiffrin (1968) lead to the idea, that a rehearsal of information is useful. The LSTM implemented this idea. Of course, a neural network with external memory like the DNC is not comparable to a human brain. Nevertheless, parts of it, like the usage-based memory allocation or dynamically used memory capacity, can also be found in the human brain, especially in the hippocampus (Graves et al., 2016). The DNC combines the idea of a short-term memory and a long-term memory. For the short-term memory, a LSTM as controller is used. The long-term memory is realised by a external memory. Therefore, the DNC has similar functions as the hippocampus, because the hippocampus can also handle short and long-term memory. Nevertheless, there are also borders of comparison, for example the backtracking problem which the LSTM handles successfully (Hochreiter and Schmidhuber, 1997). Here, the system is able to send error feedback backwards to nodes which are responsible for the input. In the human mind, this does not work because the connections of the brain are more complex and can not easily send feedback to specific neurons (Gluck and Myers, 2001). For the research in neural networks, the DNC is an important step to combine an external and dynamically changeable memory with a neural network effectively. This combination was not conceivable a few decades ago, as Judd (1990) wrote:

"[O]ne would hesitate to use neural networks just to memorize and store data because it is probably not economical at all".

The DNC shows, that this assumption no longer holds and the combination of neural
networks with external memory is possible and makes the DNC to a "learning machine that, without prior programming, can organise information into connected facts and use those facts to solve problems" (Graves et al., 2016).
References


George A Miller. 1956. The magical number seven, plus or minus two: some limits on our capacity for processing information. Psychological review 63, 2 (1956), 81.


