# TEMPORAL ATTENTION MODULES FOR MEMORY–AUGMENTED NEURAL NETWORKS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

We introduce two temporal attention modules which can be plugged into traditional memory–augmented recurrent neural networks to improve their performance in natural language processing tasks. The temporal attention modules provide new inductive biases allowing the models to compute attention distributions over the different time steps of input sequences. The values of these attention distributions can be inspected to identify the sequence's elements that the model considered relevant during the inference. Using the Entity Network (Henaff et al., 2016) as the model backbone, experiments were made on the dataset bAbI tasks, a set of QA tasks. Due to the addition of the temporal attention modules, the performance metric increased 26% when the temporal attention was supervised, and 13,5% when it wasn't. Moreover, the usage of temporal attention modules proved useful at resolving reasoning tasks that the original model was unable to solve.

## 1 INTRODUCTION

Recent advances in deep learning rarely consider explainability and interpretability as first–class citizens (Freitas, 2014; Guidotti et al., 2018). A possible cause of this phenomenon is the existence of a trade–off between explainability and learning capacity: decision trees are models that are easily interpreted due to their graphic nature, but they have a low learning capacity compared to deep learning models, which in turn have poor or non–existent innate explainability attributes.

This work seeks to push this trade–off by adding explainability attributes to deep learning models, without sacrificing performance on the task at hand. In particular, we'll design two modules that implement inductive biases capable of increasing the performance of a model, while adding interpretability attributes absent in the original model. We'll enrich the Entity Network (Henaff et al., 2016), a memory–augmented recurrent neural network, applied to natural language processing (NLP) tasks.

This work introduces two classes of temporal attention modules that can be used to add explainability attributes to the predictions of memory–augmented models. The designed modules will be evaluated in the NLP task known as question answering (QA). The temporal attention mechanisms allow the model to learn to identify which elements of the input factual sequence are important to answer a given query.

In essence, temporal attention allows the model to more easily infer answers by removing noise associated to elements of the sequence of facts that are irrelevant to the query. It also adds interpretability attributes to the original models. Interpretability is achieved by looking at the attention over time values: elements of the sequence of facts with high scores were relevant to answer the query, while those with small scores were not considered important by the model.

Two families of temporal attention modules will be designed and implemented. The first class consists of *pre–hoc* modules, that is, they calculate the temporal attention distributions before performing the answer prediction step. On the other hand, the second family is the *post–hoc* modules, that is, they execute *post–hoc* the calculation of temporal attention after having done the answer inference. Both classes of modules have advantages and disadvantages that will be deeply analyzed throughout the paper.

# 2 RELATED WORK

## 2.1 ATTENTION MECHANISMS

Sutskever et al. (2014) proposed the sequence–to–sequence architecture in an attempt to solve tasks that can be framed as mappings between pairs of source–target sequences. Many tasks can be adjusted to this framework: machine translation, summarization, question answering, among others. A limitation of this work is that it relies in an fixed–size intermediate vector which is an information bottleneck in the case of complex tasks with long sequences.

In Bahdanau et al. (2014) attention mechanisms are introduced that do not use a single intermediate vector of fixed dimensionality. Instead, they enable the model to automatically perform searches on relevant representations in the source sequence conditioned on the current state of the output sequence.

Pointer networks (Vinyals et al., 2015) have used the previously exposed notions of attention in an extractive approach. Unlike Bahdanau et al. (2014), where attention is used to build a mixture of the hidden states of the origin sequence, in Vinyals et al. (2015) attention is used as the model output by pointing directly to the origin sequence, enabling predictions to be made in tasks where the output vocabulary is not known *a priori*.

## 2.2 MEMORY-AUGMENTED NEURAL NETWORKS

An approach adopted to facilitate the learning of recurrent models has been the enrichment of these networks by the usage of external memories (Graves et al., 2014; 2016; Weston et al., 2015; Sukhbaatar et al., 2015). External memories are a set of vectors that are usually accessed through location or content-based addressing mechanisms. In essence, this family of models is inspired by classic computer architectures such as Von Neumann's (Von Neumann, 1993), where in addition to memory registers present in the processing and control units, there are external memory units that store data and instructions in the long term, or Turing's machine, where a finite automata is enriched with an infinite memory tape.

An instance of a memory–augmented neural network is the Entity Network (Henaff et al., 2016). Due to the modular nature of the Entity Network and the semantic content it learns to store in the external memory vectors $(w, h)$, it has been used as a starting point to design new modules that add attributes and capabilities that the original model does not have. For example, Bansal et al. (2017) modified the dynamic memory module to incorporate relational reasoning capabilities between entities.

## 2.3 QUESTION ANSWERING

The resolution of the WikiQA (Yang et al., 2015) and SQuAD (Rajpurkar et al., 2018) datasets has focused much of the QA–task recent efforts of the NLP community (Devlin et al., 2018; Yang et al., 2019). However, for the purposes of this work, both datasets present two issues. First, it is not possible to easily analyze the reasoning capacities that the evaluated models have, since only the correctness of the responses predicted by the model is measured. Second, an important goal of this work is to add the ability to point to the relevant information that the model considers when making inferences, to articulate an explanation that the end user uses to understand the behavior of the model. The datasets presented above lack information regarding which are the supporting facts in the context paragraphs, so it is difficult to measure the degree of success of this goal.

Considering the previous point, a dataset that is interesting is the bAbI tasks (Weston et al., 2016). It consists of twenty QA tasks that allow different cognitive reasoning skills to be tested. Each task consists of 1,000 training examples and 1,000 text test examples. Each instance of the dataset consists of: a sequence of facts, a question, an answer and the supporting facts, which consist of a subset of facts with the minimum information needed to answer the query. The construction of the tasks is done synthetically, by simulating a world where different entities interact with each other.

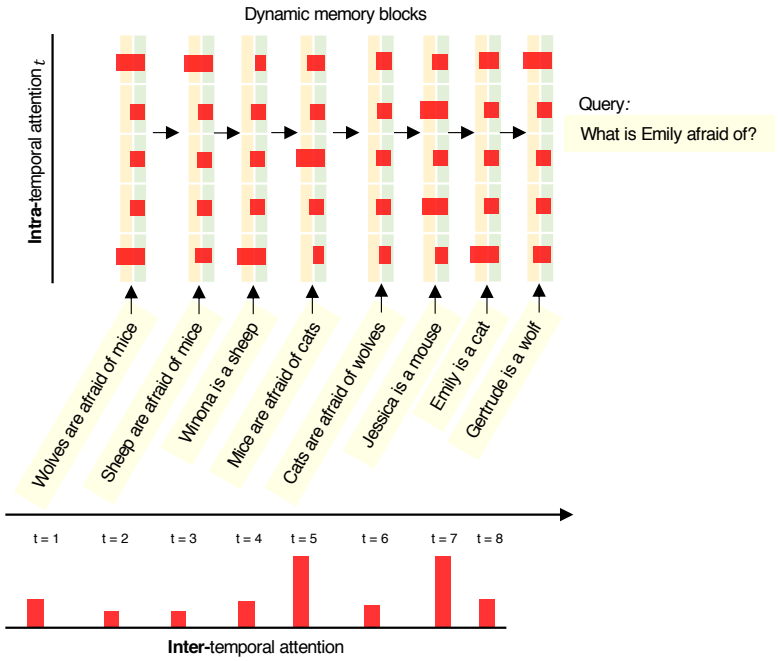


Figure 1: Intra– and inter–temporal attention. The relevance of each input sequence element can be assessed by inspecting their associate intertemporal attention value.

## 3 Temporal Attention Modules

The Entity Network (Henaff et al., 2016) will be enriched with the modules introduced in this work. It is a memory–augmented recurrent neural network whose original operations are described in the appendix. This paper extends the operations in the original output module.

### 3.1 Intra– and Inter–temporal Attention

Before we introduce the proposed modules, we will define the concepts of intra– and inter–temporal attention, illustrated in Figure 1. Let $\{s_1, \ldots, s_L\}$ be a sequence of vectors representing facts with the information needed to answer a query $q$. Furthermore, let $h_{tj}$ be the hidden states of the $j$-th memory block after processing $s_t$, and $w_j$ the $j$-th memory block key.

Intra–temporal attention is defined as some function $f$ such that intra–attention$_{tj} = f(h_{tj}, w_j, q)$. The function $f$ produces a distribution of attention over the dynamic memory blocks. This intra–temporal attention allows to put emphasis on those blocks of memory relevant to answer the query $q$, for each timestep $t$.

On the other hand, inter–temporal attention is defined as some function $g$ such that inter–attention$_{tj} = g(v_t, q)$. The argument $v_t$ is a summary of the dynamic memory blocks. $v_t$ is calculated as the sum of the memory blocks weighted by the intra–temporal attention values at timestep $t$. The inter–temporal attention can be interpreted as a distribution of attention over the elements of the input sequence. This attention can be supervised during training by incorporating it into the loss function and considering the relevant facts of the story as the ground truth. In addition, inter–temporal attention can be interpreted as an explanation of the predictions made by the model, because it will point to the time steps that the model considered relevant to answer the query $q$.

### 3.2 Pre–hoc Temporal Attention Module

The *pre–hoc* temporal attention module, illustrated in Figure 2, performs the temporal attention computation before making the answer prediction. It should be noted that the original model only uses the $h_j$ memories stored after processing the entire sequence. The *pre–hoc* module leverages

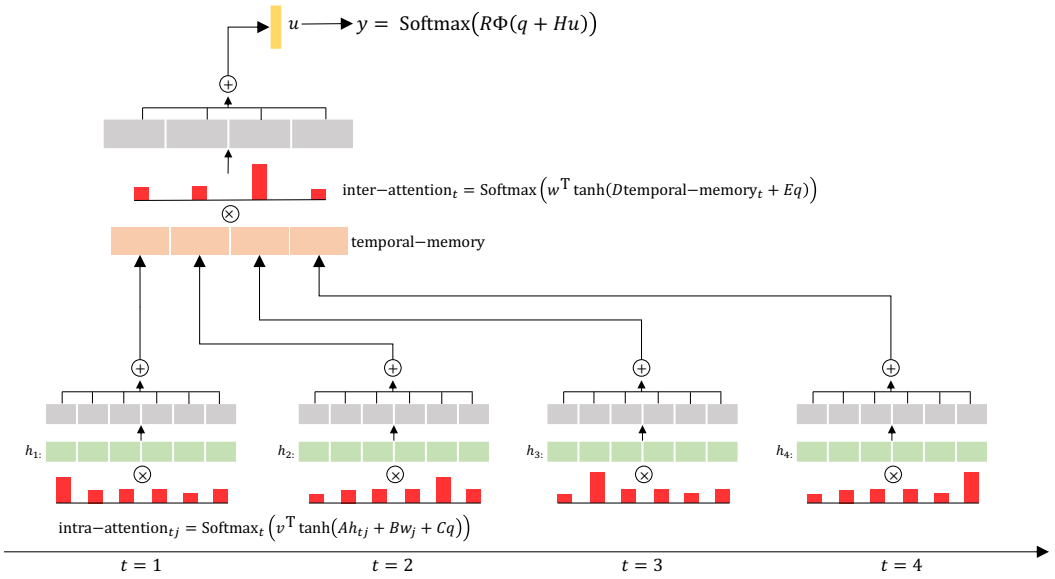


Figure 2: *Pre–hoc* temporal attention module. This module completely replaces the original output module.

all $h_{tj}$ memory values. This module has a great impact on the QA task predictions because its operations are immediately upstream of the QA operations. The *pre–hoc* temporal attention module completely replaces the Entity Network's output module. It implements the following operations:

$$\text{intra–alignment}_{tj} = v^{\text{T}}\tanh(Ah_{tj} + Bw_j + Cq) \tag{1}$$

$$\text{intra–attention}_{tj} = \text{Softmax}_t(\text{intra–alignment}_{tj}) \tag{2}$$

$$\text{temporal–memory}_t = \sum_j \text{intra–attention}_{tj} h_{tj} \tag{3}$$

$$\text{inter–alignment}_t = w^{\text{T}}\tanh(D\text{temporal–memory}_t + Eq) \tag{4}$$

$$\text{inter–attention}_t = \text{Softmax}(\text{inter–alignment}_t) \tag{5}$$

$$u = \sum_t \text{inter–attention}_t \text{temporal–memory}_t \tag{6}$$

$$y = \text{Softmax}(R\phi(q + Hu)) \tag{7}$$

The operation of this module is decomposed in three stages. First, the calculation of the intra–temporal attention is used to build $\text{temporal–memory}_t$ vectors that summarize the memories, conditioned on the query $q$, for each instant of time $t$. Next, the calculation of the inter–temporal attention is used to compute the vector $u$ corresponding to a summary of the entire dynamic memory, again conditioned to the query $q$. Finally, the answer prediction is performed with similar operations as the ones the original model uses.

Regarding the intra–temporal attention calculation, the module evaluates the alignment of the content of the memories $h_{tj}$ and their keys $w_j$ with the query vector $q$. These alignments are computed by using additive attention mechanisms as observed in equation 1. Then, the alignments are activated by a softmax function, which outputs an intra–temporal attention distribution over the memory blocks for each timestep $t$. The attention distribution is used to build summary vectors $\text{temporal–memory}_t$, which is shown in equation 3.

The second stage determines the inter–temporal alignments between the content of $\text{temporal–memory}_t$ and the query vector $q$. Like the intra–temporal attention mechanism, this alignment is calculated using additive attention mechanisms. Then, the alignments are activated with a softmax function building an inter–temporal attention distribution over time, as shown in equation 5.

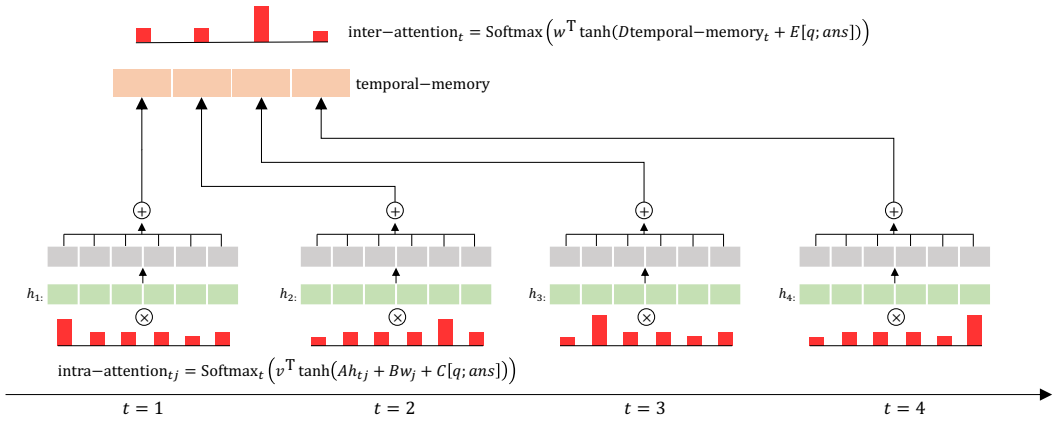


Figure 3: *Post–hoc* Temporal Attention Module. This module extends the original output module operations.

The last stage of the module is to summarize the information of the memories in a $u$ vector by summing the temporal–memory$_t$ vectors weighted by intertemporal–attention$_t$. This vector is projected into a space of dimensionality equal to the size of the vocabulary of possible answers, as observed in equation 7, to build a probability distribution of answers.

### 3.3 Post–hoc Temporal Attention Module

This module extends the original output module, leaving the operations to perform the answer prediction the same as the original model. The *post–hoc* temporal attention module, unlike the *pre–hoc* one, performs the calculation of the inter–temporal attention after predicting the question answer. The operations of this module are illustrated in Figure 3. In other words, it implements calculations that in the model computational graph that are downstream from the answer prediction operations:

$$\text{intra–alignment}_{tj} = v^{\mathrm{T}}\tanh(Ah_{tj} + Bw_j + C\text{concat}(q, ans)) \tag{8}$$

$$\text{intra–attention}_{tj} = \text{Softmax}_t(\text{intra–alignment}_{tj}) \tag{9}$$

$$\text{temporal–memory}_t = \sum_j \text{intra–attention}_{tj} h_{tj} \tag{10}$$

$$\text{inter–alignment}_t = w^{\mathrm{T}}\tanh(D\text{temporal–memory}_t + E\text{concat}(q, ans)) \tag{11}$$

$$\text{inter–attention}_t = \text{Softmax}(\text{inter–alignment}_t) \tag{12}$$

The model operations are similar to those used in the *pre–hoc* temporal attention module. Essentially, through mechanisms of additive attention over time, the model measures the relevance of the different timesteps to answer the query $q$.

Another important difference, related to the previous point, is that when calculating the intra– and inter–temporal alignments, the *post–hoc* temporal attention module is able to consider the information of the predicted answer $ans$. This way, the task of predicting the temporal attention distribution is enriched.

By using the *post–hoc* temporal attention module, it is expected to add explainable attributes and improve performance on the QA task. The values of the inter–temporal attention are interpreted as the probability that a piece of information is relevant to answer the query $q$. On the other hand, by performing the task of identifying supporting facts, we enrich the learning of better internal representations enhancing the model performance in the QA task.

## 4 Experiments

The modules were trained and evaluated on the bAbI tasks dataset (Weston et al., 2016). More information about the dataset and its structure can be found in Appendix B.

A loss function $\mathcal{L}_{\text{qa}}(D, \Theta)$ of cross entropy is used to supervise the QA task On the other hand, a loss function $\mathcal{L}_{\text{sf}}(D, \Theta)$ of binary cross entropy is used to supervise the task of identifying the supporting facts. In both terms, $D$ is the bAbI task dataset and $\Theta$ are the weights that parameterize the model.

To facilitate the learning process, the terms of the $\mathcal{L}_{\text{sf}}(D, \Theta)$ loss function associated with supporting facts are weighted by a $\frac{n_{\text{neg}}}{n_{\text{pos}}}$ scalar, where $n_{\text{neg}}$ and $n_{\text{pos}}$ are the number of non–relevant and relevant facts, respectively.

The complete loss function of the models is a weighted sum of the loss terms of each task defined as

$$\mathcal{L}_{\text{joint}}(D, \Theta) = \lambda_{\text{qa}} \mathcal{L}_{\text{qa}}(D, \Theta) + \lambda_{\text{sf}} \mathcal{L}_{\text{sf}}(D, \Theta)\,. \tag{13}$$

Where $\lambda_{\text{qa}}$ and $\lambda_{\text{sf}}$ are scalar hyperparameters that weigh the importance of the QA tasks and supporting facts identification, respectively. Both hyperparameters are equal to 1, i.e., the terms are considered with equal importance, unless stated differently. The rest of the model's hyperparameters values are in the appendix.

A multitasking training configuration is observed, where two tasks are trained in parallel.

### 4.1 Pre–hoc Temporal Attention Module

During the end–to–end training, the *pre–hoc* temporal attention module learns to perform both tasks simultaneously. It is interesting to evaluate the interaction between the tasks of QA and supporting facts identification. For this purpose, the following experiments will be performed:

1. Train the model with a loss function with values of $\lambda_{\text{qa}} = 1$ and $\lambda_{\text{sf}} = 0$. In this setting, we evaluate the model's ability to learn to answer questions and identify supporting facts without supervision in the latter task. This setting will be referred to as the weak *pre–hoc* module.
2. Train the model with a loss function with $\lambda_{\text{qa}} = \lambda_{\text{sf}} = 1$, i.e., supervision is given to both tasks.

The results of the model on the task of identifying the supporting facts of a story can be seen in Table 1. It can be seen that the weak *pre–hoc* module reaches an F1 score equal to $48\%$. This is an important result, because even though we did not directly supervise the supporting facts task, the model is capable of learning to identify part of them solely from the QA task supervision. By adding supervision in the task of identifying supporting facts, the F1 score increases by $11.4\%$ percentage points to $59.4\%$.

When evaluating the performance of the model in the QA task, the results reported in Table 1 are observed. The average error of $25.6\%$ achieved by the weak *pre–hoc* module shows that even without supervision of supporting facts it is possible to improve the performance in the QA task. A hypothesis explaining this behavior is that due to the addition of the temporal attention module, the complete model has better inductive biases to solve these tasks than the bare model backbone. On the other hand, incorporating the additional supervision results in an average error of $21.9\%$.

### 4.2 Post–hoc Temporal Attention Module

Unlike the *pre–hoc* temporal attention module, the *post–hoc* module operations to identify the supporting facts of a story are downstream of the operations used to solve the QA task. Therefore, the supporting facts identification task will be conditioned by the predicted answer of the QA task.

The results of the QA and identification of relevant facts tasks are also found in Table 1.

It can be seen that in the QA task an average error equal to $24.8\%$ is reached, corresponding to a decrease of $4.8$ percentage points with respect to the metric reached by the original Entity Network. The intuition behind this increase in performance is that supervising the supporting facts identification task induces the model to learn representations that are richer in semantics, allowing it to improve performance on the upstream QA task. However, the *post–hoc* module does not achieve the same performance as the *pre–hoc* module in the task of answering questions.

Table 1: Results of the temporal attention modules on the bAbI tasks test split. EntNet is the Entity Network. Weak *pre–hoc* is the weakly supervised *pre–hoc* temporal attention module.

| | | **Accuracy error** | | | **F1 score** | | |
|---|---|---|---|---|---|---|---|
| **#** | **Task name** | **EntNet** | **Weak *pre–hoc*** | ***Pre–hoc*** | ***Post–hoc*** | **Weak *pre–hoc*** | ***Pre–hoc*** | ***Post–hoc*** |
| 1 | QA with single supporting fact | 0.7% | 0.0% | 0.3% | 0.0% | 100.0% | 91.4% | 96.3% |
| 2 | QA with two supporting facts | 56.4% | 69.5% | 70.3% | 64.4% | 4.4% | 12.2% | 18.3% |
| 3 | QA with three supporting facts | 69.7% | 74.5% | 67.3% | 75.9% | 0.1% | 3.5% | 0.3% |
| 4 | Two argument relations | 1.4% | 0.2% | 0.0% | 2.1% | 75.6% | 100.0% | 98.0% |
| 5 | Three argument relations | 4.6% | 20.6% | 19.8% | 10.7% | 55.3% | 59.3% | 81.8% |
| 6 | Yes/No questions | 30.0% | 23.5% | 9.5% | 11.6% | 45.0% | 82.0% | 82.5% |
| 7 | Counting | 22.3% | 20.3% | 17.1% | 16.7% | 36.9% | 56.8% | 57.6% |
| 8 | Lists/Sets | 19.2% | 7.9% | 6.5% | 4.5% | 51.6% | 58.0% | 66.3% |
| 9 | Simple negation | 31.5% | 15.9% | 8.1% | 4.1% | 54.6% | 78.1% | 84.8% |
| 10 | Indefinite knowledge | 15.6% | 25.6% | 15.8% | 5.9% | 61.8% | 78.9% | 81.5% |
| 11 | Basic coreference | 8.0% | 5.3% | 0.4% | 6.9% | 61.4% | 56.1% | 53.0% |
| 12 | Conjunction | 0.8% | 0.0% | 0.5% | 0.5% | 100.0% | 86.8% | 90.0% |
| 13 | Compound coreference | 9.0% | 7.5% | 0.2% | 6.0% | 59.9% | 53.4% | 52.2% |
| 14 | Time manipulation | 62.9% | 23.0% | 22.6% | 32.2% | 29.4% | 43.6% | 55.7% |
| 15 | Basic deduction | 57.8% | 29.4% | 18.1% | 60.5% | 47.1% | 64.0% | 66.7% |
| 16 | Basic induction | 53.2% | 50.2% | 52.0% | 52.7% | 6.7% | 25.6% | 49.9% |
| 17 | Positional reasoning | 46.4% | 41.0% | 41.0% | 40.9% | 69.7% | 82.1% | 66.7% |
| 18 | Reasoning about size | 8.8% | 9.2% | 4.5% | 9.5% | 2.7% | 34.8% | 41.7% |
| 19 | Path finding | 90.4% | 88.7% | 83.1% | 89.8% | 0.0% | 21.2% | 25.7% |
| 20 | Motivations | 2.6% | 0.0% | 0.0% | 1.0% | 97.7% | 100.0% | 100.0% |
| **Mean** | | 29.6% | 25.6% | 21.9% | 24.8% | 48.0% | 59.4% | 63.4% |

The *post–hoc* module achieves an F1 score equal to $63.4\%$ in the supporting facts identification task. In this task, the *post–hoc* module has better performance to any version of the*pre–hoc* module.

In other words, if a practitioner wants to prioritize the model's ability to identify supporting facts in a story, the *post–hoc* module is the most convenient. On the other hand, if the performance in the task of answering questions is more important, the *pre–hoc* module seems to be the most suitable choice.

### 4.3 Supporting Facts Prediction

We are not aware of other work leveraging the bAbI tasks supporting facts features and reporting metrics on the task of predicting them that would enable quantitatively analysis and comparison. Nevertheless, it is possible to do a qualitative analysis by observing a sample of dataset records and the predictions made by the model.

The analyzed examples are records of the test dataset. The best weak *pre–hoc* temporal attention module was used to process the records, so the only supervision the model received during training was the ground truth answer. Facts with a red background are those marked in the dataset as supporting facts.

An example of the bAbI task #1 which has only one supporting fact in throughout its records is observed in Figure 4a. It can be seen how the temporal attention module focus its attention on the last element of the story, which are the supporting fact labeled in the data set.

An interesting aspect of the behavior of the model is that it focus mass of the inter–temporal attention distribution on elements of the story that, although weren't labeled as supporting facts, involve the query main entity. This can be clearly seen in the examples in Figures 4a and 4b. The above is an example of the explainable attributes that the temporal attention modules add to the model, since it's possible to check that the model is putting greater emphasis to the information that should be considered.

| Fact | |
|---|---|
| John traveled to the hallway. | 0,00 |
| Mary journeyed to the bathroom. | 0,00 |
| Daniel went back to the bathroom. | 0,00 |
| John moved to the bedroom. | 0,00 |
| John went to the hallway. | 0,00 |
| Sandra journeyed to the kitchen. | 0,00 |
| Sandra traveled to the hallway. | 0,05 |
| John went to the garden. | 0,00 |
| Sandra went back to the bathroom. | 0,26 |
| Sandra moved to the kitchen. | 0,69 |
| **Where is Sandra?** | *Ground truth: kitchen.* |
| **Prediction:** kitchen. | |

(a) bAbI task #1 "QA with single supporting fact".

| Fact | |
|---|---|
| Mary got the milk there. | 0,00 |
| John moved to the bedroom. | 0,00 |
| Sandra went back to the kitchen. | 0,01 |
| Mary traveled to the hallway. | 0,00 |
| John got the football there. | 0,00 |
| John went to the hallway. | 0,78 |
| John put down the football. | 0,00 |
| Mary went to the garden. | 0,04 |
| John went to the kitchen. | 0,15 |
| Sandra traveled to the hallway. | 0,03 |
| **Where is the football?** | *Ground truth: hallway.* |
| **Prediction:** bedroom. | |

(b) bAbI task #2 "QA with two supporting facts".

Figure 4: Sample bAbI task records with its predicted inter–temporal attention values in the second column

## 5 Conclusions and Discussion

This work successfully shows that the temporal attention modules to a recurrent model with external memory increases performance in QA tasks and, simultaneously, adds interpretability attributes previously unavailable in the original model. In the best configuration, an average accuracy error equal to 21,9% is observed, which corresponds to a 26% improvement compared to the error obtained by the base model called the Entity Network.

Although it is not possible to carry out a quantitative comparison on the performance obtained in the task of identifying supporting facts, which is what adds the component of explainability to the model, the qualitative analysis of the results obtained is also satisfactory. The model is able to learn to point to the supporting facts of a given story and question even without explicit supervision.

A future line of work is to validate the benefits of the temporal attention modules with other instances of memory–augmented neural networks. Due to the modular nature of the proposed architectures, it is possible to integrate them into models such as the Memory Network (Sukhbaatar et al., 2015) or the Differentiable Neural Computer (Graves et al., 2016), among others.

During the development of this work, much of the progress done in the NLP community has been due to the application of large models with hundreds of millions of parameters in massive datasets (Devlin et al., 2018; Yang et al., 2019). In comparison, the complete models presented in this work have 4 orders of magnitude less complexity in terms of the number of parameters. An interesting line of research in the future is to quantify the impact on model performance that would be produced by the effect of scaling the number of parameters of the model. If this line is continued, there are more complex datasets that may be more convenient to use (Yang et al., 2018).

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, sep 2014. URL http://arxiv.org/abs/1409.0473.

Trapit Bansal, Arvind Neelakantan, and Andrew McCallum. RelNet: End-to-End Modeling of Entities \& Relations. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, jun 2017. URL http://arxiv.org/abs/1706.07179.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. oct 2018. URL http://arxiv.org/abs/1810.04805.

Alex A. Freitas. Comprehensible Classification Models - a position paper. *ACM SIGKDD Explorations Newsletter*, 15(1):1–10, 2014. ISSN 19310145. doi: 10.1145/2594473.2594475.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural Turing Machines. 2014. URL http://arxiv.org/abs/1410.5401.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, Adrià Puigdomènech Badia, Karl Moritz Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, Christopher Summerfield, Phil Blunsom, Koray Kavukcuoglu, and Demis Hassabis. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626): 471–476, 2016. ISSN 14764687. doi: 10.1038/nature20101.

Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A Survey of Methods for Explaining Black Box Models. *ACM Computing Surveys*, 51(5):1–42, aug 2018. ISSN 03600300. doi: 10.1145/3236009. URL http://arxiv.org/abs/1802.01933.

Zellig S. Harris. Distributional Structure. *WORD*, 10(2-3):146–162, aug 1954. ISSN 0043-7956. doi: 10.1080/00437956.1954.11659520. URL http://www.tandfonline.com/doi/full/10.1080/00437956.1954.11659520.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1026–1034, 2015. ISBN 9781467383912. doi: 10.1109/ICCV.2015.123.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. Tracking the World State with Recurrent Entity Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1612.03969.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1412.6980.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know What You Don't Know: Unanswerable Questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pp. 784–789, 2018. doi: 10.18653/v1/p18-2124.

Leslie N. Smith. Cyclical learning rates for training neural networks. In *Proceedings - 2017 IEEE Winter Conference on Applications of Computer Vision, WACV 2017*, number April, pp. 464–472, 2017. ISBN 9781509048229. doi: 10.1109/WACV.2017.58.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-To-End Memory Networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, mar 2015. URL http://arxiv.org/abs/1503.08895.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to Sequence Learning with Neural Networks. 2014. URL http://arxiv.org/abs/1409.3215.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer Networks. 2015. URL http://arxiv.org/abs/1506.03134.

John Von Neumann. First draft of a report on the EDVAC. *IEEE Annals of the History of Computing*, 15(4):27–75, 1993. ISSN 1058-6180. doi: 10.1109/85.238389. URL http://ieeexplore.ieee.org/document/238389/.

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory Networks. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1410.3916.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M. Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, feb 2016. URL http://arxiv.org/abs/1502.05698.

Yi Yang, Wen-tau Yih, and Christopher Meek. WikiQA: A Challenge Dataset for Open-Domain Question Answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 2013–2018, 2015.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, Stroudsburg, PA, USA, sep 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259. URL http://arxiv.org/abs/1809.09600http://aclweb.org/anthology/D18-1259.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. XLNet: Generalized Autoregressive Pretraining for Language Understanding. jun 2019. URL http://arxiv.org/abs/1906.08237.

# A Entity Network

The Entity Network (Henaff et al., 2016) is a recurrent model with an external memory capable of solving question answering and dialogue tasks, among others. This model introduces inductive biases to identify entities in the input domain, in order to track relevant properties of them.

The model is decomposed into three modules: one input module, one dynamic memory and finally an output module. The new modules proposed in this work extend the output module, adding explainability attributes. The original modules in a question answering setting will be described below to contextualize the reader.

## A.1 Input Module

Under the QA setting, the Entity Network's input module receives a sequence of vectors $\{e_1, \ldots, e_L\}$, where $e_i \in \mathbb{R}^e$ is the $i$-th token embedding of a sentence and $L$ is the sentence length. The input module is responsible of building a fixed dimensionality vector $s_t$ for each time instant $t$. This vector $s_t$ is calculated in the following way.

$$s_t = \sum_i f_i e_i \qquad\qquad s_t, f_i, e_i \in \mathbb{R}^e \tag{14}$$

The multiplicative masks $\{f_1, \ldots, f_L\}$ are trainable weights of the model. This encoding method is a generalization of a bag of words encoding (Harris, 1954), which occurs when the multiplicative masks weights are exactly 1 in all their entries. This encoding method provides flexibility to differently assess the importance of each token according to its location within a sentence.

## A.2 Dynamic Memory Module

The dynamic memory module receives the $s_t$ vectors, process them sequentially and updates the corresponding memory blocks. The dynamic memory consists of pairs $(w_j, h_j), j \in \{1, \ldots, M\}$, where $w_j$ and $h_j$ is a key–value pair corresponding to the $j$-th memory and $M$ is the number of memory blocks. The $w_j$ keys, which are trainable weights, remain frozen while a sequence is processed. On the other hand, the $h_j$ values are updated at each timestep, according to a behavior

inspired by a gated recurrent network. The equations used in each recurrent step to update the dynamic memory module are described below.

$$g_j = \sigma(s_t^{\mathrm{T}} w_j + s_t^{\mathrm{T}} h_j + b_j) \qquad g_j, b_j \in \mathbb{R}; w_j, h_j \in \mathbb{R}^e \tag{15}$$

$$h_j^{\sim} = \phi(U h_j + V w_j + W s_t + b_h) \qquad h_j^{\sim}, b_h \in \mathbb{R}^e; U, V, W \in \mathbb{R}^{e \times e} \tag{16}$$

$$h_j = h_j + g_j h_j^{\sim} \qquad h_j \in \mathbb{R}^e \tag{17}$$

$$h_j = \frac{h_j}{\|h_j\|} \qquad h_j \in \mathbb{R}^e \tag{18}$$

$\sigma$ and $\phi$ are the sigmoid and PReLU (He et al., 2015) activation functions, respectively. The value of $g_j$ is a scalar that acts as a gate mechanism and determines how much the $j$-th memory should be updated at the current step. The similarity of the input vector $s_t$ and the vectors $w_j$ and $h_j$ determine how much should the gate close. Next, a candidate vector $h_j^{\sim}$ is computed according to a set of linear transformations of $h_j$, $w_j$ and $s_t$ which are added together and activated by $\phi$. The candidate vector $h_j^{\sim}$ is used to update the memory value $h_j$ by weighting all its components by the gate scalar $g_j$. Finally, the updated vector $h_j$ is normalized inside the unitary sphere by dividing it by its L2 norm, which allows forgetting non-relevant information.

### A.3 Output Module

The original output module computes, through attention mechanisms over the resulting memories, the answer to some query represented by $q$. A query is a sequence of token embeddings $e_1, \ldots, e_L$ that is processed through a mechanism similar to the input module. Each embedding $e_i$ is weighted by multiplicative masks $\{f_1, \ldots, f_L\}$ and then summed to compute the query representation $q$. It's important to note that these multiplicative masks are different from those used in the input module.

Once the representation of the $q$ query is calculated, it's possible to predict the answer to it using the information stored in the memory blocks.

$$p_j = \text{Softmax}(q^{\mathrm{T}} h_j) \qquad p_j \in \mathbb{R}; q \in \mathbb{R}^e \tag{19}$$

$$u = \sum_j p_j h_j \qquad u \in \mathbb{R}^e \tag{20}$$

$$y = \text{Softmax}(R\phi(q + Hu)) \qquad y \in \mathbb{R}^V; R \in \mathbb{R}^{V \times e}; H \in \mathbb{R}^{e \times e} \tag{21}$$

The attention scores $p_j$ are computed through a dot–product attention mechanism between the representation of the query $q$ and the memory value $h_j$. Then, a vector $u$ is calculated, which is a summary of the dynamic memory, focusing on those blocks that are relevant to answer the query. Finally, a softmax distribution is computed over the answer vocabulary conditioned on the query representation $q$.

This paper extends the operations in the original output module. The proposed enhancements implement temporal attention modules, adding explainability attributes and improving the model performance at the same time.

## B bAbI Tasks

Both modules will be evaluated in the bAbI tasks dataset. The bAbI tasks are a collection of twenty question answering tasks, where each one of them a different type of cognitive reasoning capability. The dataset is constructed synthetically, simulating entities that interact with each other altering the fictional world state where they live.

Each task consists of several training examples. Each training example is composed of:

- a story or sequence of facts $\{\text{fact}_1, \ldots, \text{fact}_T\}$, where each fact is a sequence of natural language tokens $\{x_1, \ldots, x_{L_f}\}$,

1 The suitcase is bigger than the container.
2 The container fits inside the box.
3 The chest is bigger than the chocolate.
4 The suitcase fits inside the box.
5 The chest fits inside the box.
*Query:* Does the chocolate fit in the box?
*Answer:* Yes

Figure 5: bAbI task #18 training record example. It's composed by a facts sequence $\{\text{fact}_1, \ldots, \text{fact}_5\}$, a query $q$ and the answer $ans$ to that question. The supporting facts have a red background color.

- a question $q$ structured as a sequence of natural language tokens $\{x_1, \ldots, x_{L_q}\}$,
- an natural language answer $ans$ to the question, and
- the set of relevant facts supporting–facts, which are the smallest subset of facts with sufficient information to answer the question.

An example of a training record of the dataset is in Figure 5.

Models designed to solve the bAbI tasks can be classified into two training strategies: a weakly supervised one, where only the ground truth answer is provided as supervision, and another strongly supervised, where both the ground truth answer and the subset of supporting facts are given as supervision. The pre hoc temporal attention module is able to operate on both weakly and strongly supervised settings. On the other hand, the post hoc temporal attention module operates exclusively in the strongly supervised mode.

The bAbI tasks are available in two versions: the *1k* one with 1,000 training examples and the *10k* one with 10,000 training examples for each task. Both versions have test datasets with the same number of instances as the training split. Generally, the *1k* version of the dataset is considered to be more difficult to solve than the *10k* one because it offers an order of magnitude less training examples to learn the different types of reasoning needed to solve the tasks.

The augmented–modules presented in this work learn to predict both the answers and supporting facts of the bAbI tasks. The identification of those supporting facts is done using the intertemporal attention distributions. The resolution of this task provide explainable attributes to the models.

## C HYPERPARAMETERS

Each training experiment is performed during 200 epochs. At the end of each epoch, the loss function is evaluated on a validation split, built as a random sample of 10% of the training split. Finally, the model that minimizes the loss function on the validation data split is evaluated on the test subset.

The learning rate varied cyclically (Smith, 2017) between $5 \times 10^{-3}$ and $5 \times 10^{-5}$ with a period of 6 epochs. The embedding size was 100 and 20 blocks of dynamic memory were used. The linear transformations observed in equation 1, equation 4, equation 8 and equation 11 projected the inputs into a space of dimensionality equal to 50.

Parameters were initialized from a normal distribution with $\mu = 0$ and $\sigma = 1$, except for the PReLU activation function slopes and the multiplication masks which were initialized equal to 1. Gradients were clipped to have at most an L2–norm equal to 40. Finally, the model parameters were tuned using stochastic gradient descent, a batch size equal to 32 and the Adam optimizer (Kingma & Ba, 2014).

Table 2: Teacher forced *pre–hoc* temporal attention module results at the QA task on the test split. Accuracy errors are reported. Lower values are better. EntNet is the Entity Network. MemNN SS is the strongly supervised End–to–End Memory Network.

| **Task** | **Name** | **EntNet** | **MemNN SS** | ***Pre–hoc* module** |
|---|---|---|---|---|
| 1 | QA with single supporting fact | 0,7% | 0,0% | 0,00% |
| 2 | QA with two supporting facts | 56,4% | 0,0% | 0,00% |
| 3 | QA with three supporting facts | 69,7% | 0,0% | 0,00% |
| 4 | Two argument relations | 1,4% | 0,0% | 0,00% |
| 5 | Three argument relations | 4,6% | 2,0% | 0,88% |
| 6 | Yes/No questions | 30,0% | 0,0% | 0,00% |
| 7 | Counting | 22,3% | 15,0% | 9,08% |
| 8 | Lists/Sets | 19,2% | 9,0% | 6,45% |
| 9 | Simple negation | 31,5% | 0,0% | 0,00% |
| 10 | Indefinite knowledge | 15,6% | 2,0% | 2,83% |
| 11 | Basic coreference | 8,0% | 0,0% | 0,00% |
| 12 | Conjunction | 0,8% | 0,0% | 0,00% |
| 13 | Compound coreference | 9,0% | 0,0% | 0,00% |
| 14 | Time manipulation | 62,9% | 1,0% | 0,00% |
| 15 | Basic deduction | 57,8% | 0,0% | 0,00% |
| 16 | Basic induction | 53,2% | 0,0% | 0,00% |
| 17 | Positional reasoning | 46,4% | 35,0% | 39,45% |
| 18 | Reasoning about size | 8,8% | 5,0% | 7,13% |
| 19 | Path finding | 90,4% | 64,0% | 55,86% |
| 20 | Motivations | 2,6% | 0,0% | 0,00% |
| | **Mean error** | 29,6% | 6,7% | 6,1% |
| | **Failed** ($> 5\%$) | 15 | 5 | 5 |

# D Additional Experiments

## D.1 Pre–hoc Temporal Attention Module: Teacher Forced

During the teacher forcing experiments the prediction of inter–temporal attention, observed in equation 5, is not performed by the model. Instead, the inter–temporal attention is forced to be the ground truth supporting facts given by the dataset, as seen in the next equation.

$$\text{inter–attention}_t = \begin{cases} 1 & \text{if fact}_t \in \text{supporting–facts} \\ 0 & \text{if fact}_t \notin \text{supporting–facts} \end{cases} \tag{22}$$

Even though the model doesn't learn to predict the inter–temporal attention –since the ground truth values are forced–, this experiment is helpful to get an approximation of the upper bound performance in the QA task if the module learns to perfectly predict the supporting facts of a record.

The results of this experiment can be seen in table 2.

A significant performance difference is observed between weakly supervised methods, such as the Entity Network, and the strongly supervised End–to–End Memory Network, or MemNN SS, (Sukhbaatar et al., 2015) or the teacher forced *pre–hoc* module. Substantial increases in performance are observed in tasks 2, 3, 14, 15 and 16. On the other hand, there are some tasks, such as number 18, where learning to perfectly predict the inter–temporal attention doesn't seem to help in terms of performance.

Task number 18 tests the model's spatial reasoning ability. Essentially, its records depict a relationship of size between entities, e.g., "the chocolate fits inside the box", and then asks questions regarding these relationships, e.g., "do you have the box fit in the chocolate?". It makes sense that the aggregation of the temporal attention module doesn't significantly help the model's performance on this task, since it's not a task whose temporal dimension is relevant for solving it.

Finally, when comparing the *pre–hoc* module with the strongly supervised End–to–End Memory Network, an our module has an average performance of $0{,}6\%$ higher. However, there are still 5 tasks that none of the strong models are capable of solving, specifically tasks 7, 8, 17, 18 and 19. A reasonable hypothesis to explain the unsatisfactory performance of the model in these tasks is that they need types of reasoning that the model does not have. For instance, just as task 18 is about spatial reasoning, task 17 requires strong spatial awareness skills to be solved. It's a future challenge to incorporate model inductive biases in order to acquire these reasoning abilities.

After analyzing the results, it's clear that if the *pre–hoc* module is able to correctly identify the relevant facts of a story, it'll be able to solve reasoning tasks that weakly supervised counterparts are not capable of.