Integrating Attention mechanisms into Recurrent Highway Networks with Grouped Auxiliary Memory

Shaurya (shaurya.2022@vitstudent.ac.in),
Devika Krishna Iyer (devika.iyer2022@vitstudent.ac.in)
Vellore Institute of Technology, VIT University, Chennai.

Abstract:

Sequence to sequence tasks relate to the generation of sentences based on a certain set of input sentences. Some tasks include language translation, speech recognition, time-series forecasting, recommendation systems, and a wide range of applications in the realistic world, making it all the more essential that the output be coherent and relevant to the input. At their core, recurrent neural networks and their variants are meant to comprehend and analyze time-based information. Such networks are necessary for many real-world applications-ranging from analyzing stock prices over time to translating between different languages. By processing the information provided in each part of a sequence in context with what comes before, RNNs are effective in coding applications in which the order and the timing of events is important.
However, RNNs experience a major limitation in processing exceedingly long functional sequences. The most serious limitation is the "vanishing gradient" problem where the gradients fall into an exponential decrease, which creates a lack of information. This will hinder RNNs from learning long-term dependencies for which the loss would be extremely relevant to predict. Another limitation consists of every component of a sequence receiving an equal stream of information from the previous moment, even though each component may not be equally relevant. This inability by the model to demean and prioritize the biased and associated information will take a lot of computational time. This also brings forth the necessity to improve RNNs in processing long sequences. Efforts have been made to address these limitations, with variants like recurrent highway networks (RHNs), added mechanisms for memory, and attention techniques demonstrating conclusively that the augmentation of attention networks contributes significantly to the better performance of RHNs. The accuracy achieved was 0.5781 and perplexity achieved was 4.8876.

Keywords: Recurrent Neural Networks(RNN), Recurrent Highway Networks(RHN), Grouped Auxiliary Memory(GAM), Attention Mechanisms.

1. Introduction

A recurrent neural network is a certain category of neural network which processes sequence-based data by implementing memory to grasp information between different time steps. A feature of RNNs is the feedback loop, which allows information to stay in the current state. Both the input and the hidden state from the previous step are passed to the RNN at every time step, which in turn allows for output and the next hidden state. From here, a connection is formed temporally across the sequence into the next time step. However, a disadvantage of this is that it suffers from the vanishing gradients problem. It is again very commonly found that as training of long sequence-based data is performed, the gradients shrink or dwindle away exponentially through the network generally leading to lack of the long-term dependence feature among their observations in the data.

Recurrent Highway networks are a radical approach toward providing a solution to the problem of vanishing gradients which arises in deep RNN networks. RHNs use highway layers, allowing information to flow forward through the network with very little interference. The placement and stacking of highway layers and skip connections adequately preserve long-term dependencies throughout extended sequences while minimizing the loss of gradient. From this, it follows that RHNs stand out as a major advance to RNNs and thereafter a useful form of finding a solution to capturing and processing their dependencies over longer timescales, particularly amenable to complex sequential tasks. A fundamental innovation in the highway layers includes a gating mechanism that is characterized and never allows excess data to be directly pushed forward, serving in restraining how far and much of the non-linear transformation is given in a highway layer. It could allow selective skip transformations and propagate unaltered information along the path through the highway layers, performing effectively in preserving crucial information on long stretches that would otherwise fade away deep into the RNNs. That gave RHNs an added elegance to maintain intermediate and long-range dependencies in all areas of the RNN while not compromising either performances or stability in the network.

Another important aspect of this architecture is the dedicated addition of Grouped Auxiliary Memory (GAM) which extends the ability of the network to preserve long-term dependencies in the input by moving them to auxiliary memory which is separate from the primary recurrent state memory. Typical RNNs repeatedly update recurrent states to the repetition of time steps, inhibiting long-term dependence storage. That is where GAM comes into play, providing a dedicated and persisting memory structure. At every time point, the network will be able to read and write to the auxiliary memory such that the necessary information is kept, while the entire process is carried out without modification over the network. Such separation of memory functions allows RHNs to attain information on crucial long-range dependencies whilst affording the main recurrent states the luxury to learn short-range dependencies. An auxiliary memory structure arranged into groups, each designed to hold a certain piece of information in the context of the input sequence. This allocation of the input ensures that long-term dependencies are retrieved and updated expeditiously, thus preserving any crucial contextual information throughout the process. The grouped structure of the data allows a network to access only the specific information needed for effective prediction in a completely non-redundant and unencumbered way. This is how RHNs improve the accuracy of prediction, even with complex extended sequences of data.

Our proposed methodology does work within the GAM-RHN framework by introducing attention mechanisms. Attention mechanisms work by helping the network to focus on important aspects of the input sequence, therefore helping to increase the performance. Besides that, it also allows the model to focus more on some parts of the sequence that are closely related to the goal task, letting the model direct its resources toward more informative segments. This increased focus enables the RHN to extract detailed relationships amongst the data, dynamically focusing on parts of the data relevant to the task. The attention mechanism built into the architecture works by computing an attention score that indicates how much correlation exists between any input element and the target output. This score then guides the network by directing its focus toward those inputs that have more inflating influence. Thus, narrowing the focus of the network per se helps the model in knocking off the noise caused by less relevant sequences so that a more accurate and efficient outcome can be obtained. Built on this, one sees particular applicability in application domain/generative tasks where only some sections of long sequence inputs dominate and affect the final output while the rest introduces further meaningless noise.

In conclusion, the proposed methodology, which integrates the strengths of GAM-RHNs and attention mechanisms, attempts to capture long-term dependencies and improve the ability of the model to pay attention to and focus on relevant segments of the data. This combination ultimately gives rise to a better gateway and responsive network, capable of more complex sequence tasks with better accuracy. The complete evaluation of the model will give insight into its ability to predict and the potential of the attention mechanism in facilitating long-term memory retention and focus.

2. Literature Survey

Recurrent neural networks are neural networks that leverage memory to capture information across time steps to process sequential data. The hidden states of the RNN are used as dynamic memory and are overwritten at each time step to hold information related to the current input sequence and make predictions as demonstrated by Graves et al [1]. While the recurrent units generate sequences that match the style of the input sequences, they are not immune to the “vanishing gradient problem”. While the LSTM units alleviate the issue to a certain extent, the ability of the model to capture long term dependencies over very long sequences still presents a challenge for the network. A similar observation was made by De Mulder et al [2] in their paper, stating that RNNs can only theoretically capture long term dependencies in sequences. RNNs still perform better than non-sequential models in capturing dependencies but are restrained in their ability due to the vanishing gradient problem.

As introduced by Zilly et al [4], Recurrent Highway Networks are primarily built to address the vanishing gradient problem in RNNs. This variant of Recurrent Neural Networks uses a gating mechanism to allow certain information to bypass the non linear transformations ensuring that the gradients do not diminish exponentially. This ensures a longer term memory retention by ensuring that there is no gradient decay. This form of neural network however, can be complex and computationally intensive and thereby are prone to instability. One avenue of stabilising the model could be to introduce batch normalisation into the model, as proposed by Chi Zhang et al [5]. This is achieved by applying batch normalisation after each non linear activation to both the transform and carry gates. Thus, using normalisation enables the operation of normalised output which leads to a more consistent gradient flow thereby stabilising the model.

RHN models can be modified to improve the performance on tasks like speech recognition, where retaining long term dependencies proves crucial. As proposed by Pundak et al [6], Highway LSTM and RHN architectures are combined to facilitate better information flow across the layers. It is done by adding the carry gate and transform gate to the LSTM units and the highway layers control the flow of information across timesteps enabling the model to capture complex temporal dependencies. Another form of controlling the flow of information through the recurrent states is by using Fast-Slow Recurrent Neural Networks [11]. Choi et al [12] demonstrates a similar mechanism wherein the model is used for character level language modelling and thereby captures long term dependencies on a smaller scale making it more accurate. It uses gating mechanisms to help retain and focus on relevant information. The model as proposed by Mujika et al introduces two recurrent pathways that operate at varying speeds to capture both local and global dependencies. The performance of the model is enhanced by separately capturing different temporal scales which makes it more efficient. Both the LSTM and Fast-Slow RNN model tackle the vanishing gradient problem while also making significant strides towards long-term dependencies in terms of learning contextual information.

This would involve the introduction of an auxiliary memory module into the RNN model so that temporal information at the previous timesteps can be stored within the neural network. An important example is given wherein Luo et al [7] use a grouped distributor unit to capture long term dependencies. Grouped memory units break the memory into different groups, each unit of which retains Segment One of the input sequence. Zang et al[8] uses this similar kind of auxiliary memory augmentation in the model to provide easy data access while retaining long term dependencies and previous contextual information of the input sequences.

Attention mechanisms built into the RNN model to enable it to learn to concentrate more on the informative parts of the input sequence relevant to the output target contribute immensely to the performance improvement of the model. Mikolov et al [3] discuss the use of a context vector containing segments from previous steps to enhance context information for the current input sequence, thereby enabling more accurate prediction. Soydaner et al[10] also provided another implementation of this mechanism. The role and usability of attention mechanisms across different fields are further reviewed by Soydaner et al in order to observe their deployment and effects on the performance of RNNs. Chowanda et al [14] also use attention mechanisms to augment the recurrent neural network in the context of conversational modelling. It has been observed that the attention mechanisms contribute significantly to improving the focus of the model and coherence in the output. Attention mechanisms are also essential to generate coherent dialogues and help focus on contextual information as observed by Mei et al [15].

An innovative stride in the field of Recurrent Neural Network is the combination of the strengths of RHNs and the memory segmentation as implemented by Luo et al [13]. The model overcomes the vanishing gradient problem with the implementation of highway layers to control the flow of information through the network. It also shows improved focus since it separates the memory function from the recurrent unit by using a Grouped Auxiliary memory which stores temporal information regarding the various segments of the input sequence and information of previous timesteps. This ensures that the memory is not overwritten with each timestep and that the long term dependencies are preserved.

The purpose of this paper is to combine the strengths of the models previously explored to build a model that can generate coherent sentences in the field of language modelling. This paper incorporates the improved focus of the attention mechanism with the GAM-RHN model as seen previously.

3. Proposed Work

3.1. Overview

The proposed work seeks to augment the performance of RHNs by integrating an attention mechanism and in doing so address the limitations in capturing the long term dependencies in the input data. RHNs coupled with a GAM, are effective in retaining long term features, but lack dynamic focus on the features within the input that display most significance to the task at hand. Attention mechanisms allow the model to make decisions relevant to the input sequence and minimise noise while boosting accuracy. Attention scores provide a way of directing the focus of the network, allowing selective readings of the data and storage of critical information. This method solves the problem on the previous models that had static attention spans and short-term memory. This combination strengthens the model’s ability to handle complex, extended sequences efficiently.

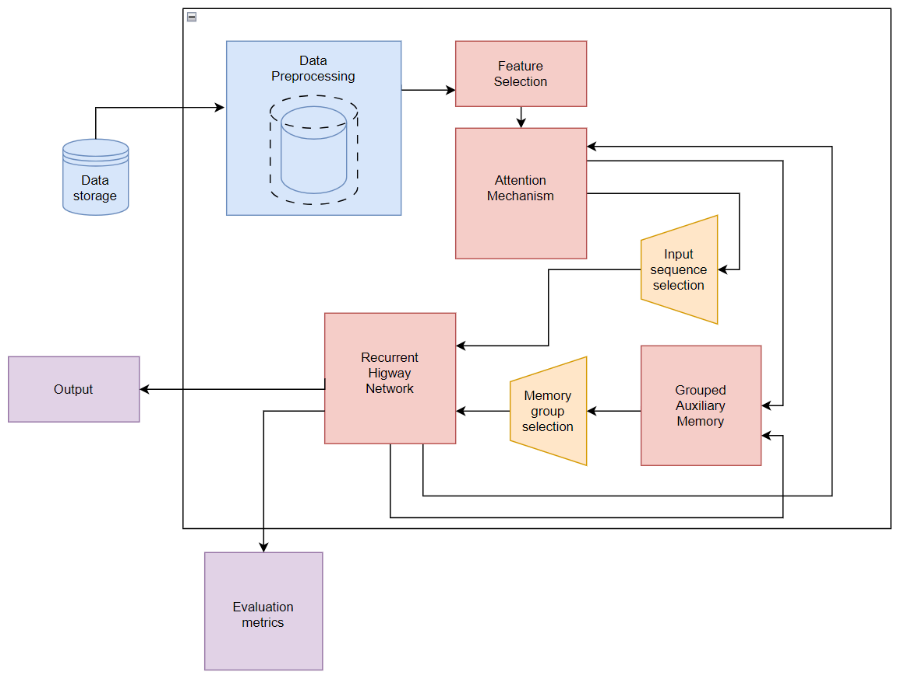

Fig 1. Proposed Work Architecture

The Data Storage module acts as a repository for the input data that is fed into the network pipeline. The data included structured and unstructured data that is relevant to the network used and the application domain. This enhances the quality of the data for preprocessing and further pipeline stages and factors in activeness for the recommendation algorithms.

The recommended preprocessing operates to cleanse data to create a training-compatible state. This involves normalisation, deconstruction, augmentation, etc. This should also take care of any missing or null data components and make sure the data is scaled universally so that the loss is minimal during the gradient train. This becomes helpful to reduce the noise that can help to do proper data quality improvement, which in turn plays a key role in the model accuracy. Feature selection allows minimisation of the computational complexity which reduces the dimensionality of the data and ensures the model focuses on impactful features to avoid overfitting the data.

Attention mechanisms enhance the model's ability to focus on specific parts of the input sequence. In this context it may be used to give higher importance to more impactful segments of the data, allowing the model to capture the intricate details in long sequences. The input sequence is selected based on the output of the attention mechanism indicating that the sequence with highest relevance to the desired output is selected for the input data. This manages the amount of data flowing into the network thereby optimising resource utilisation.

The Grouped Auxiliary memory module serves as an additional memory structure to augment the model's ability to focus on relevant sequences of data. These units are tasked with capturing different aspects of the input sequence, providing more context for each recurrent step in the network. Memory groups are selected based on the contextual aspect of the input sequence to the current recurrent stage. This determines which group of memory of the input sequence is more significant regarding the current time step. It provides information about previous time steps regarding the current input sequence.

The Recurrent Highway Network is the primary model component responsible for processing sequential data. It takes the input sequence dictated by the attention mechanism and the knowledge at previous time steps that happens through the grouped auxiliary memory to learn the long-term dependencies in the data. They use highway connections to facilitate gradient flow for the hard errors.

3.2. Dataset Exploration:

The Penn Treebank data set has become a standard within the field of NLP and is used for the construction and evaluation of language modelling, part-of-speech tagging, parsing and syntax analysis. It has a journalistic source in the set of texts that are used by the PTB, mainly on issues in sequence modelling tasks, such as on language modelling and part-of-speech tagging. With the availability of annotated texts exceeding 4.5 million words, it lends itself most to those tasks that demand the capturing of long-term dependencies, found in the core of the GAM-RHN model.

The PTB text is tokenized into sequences of tokens for the task of language modelling, where each token generally represents a word or punctuation. Tokenization is essential in that it allows models to learn dependencies between words and represent syntactic relations among elements within sentences. The structure of the PTB allows these sequence modelling tasks to learn the long-term dependencies within the input data, thus enabling models to predict sequences of words based on context provided by multiple preceding words, thereby making the dataset suitable for the evaluation of models such as GAM-RHN.

Typical splits of the PTB data set are represented by training, validation, and test sets that allow the model enough exposure to train adequately without overtly overfitting. Annotated POS tags and syntactic trees make this corpus all the more valuable for linguistic analyses and syntactic parsing beyond language modelling. These long-term dependencies allow the comparison of models like GAM-RHN, particularly when FOE techniques are incorporated to boost the model's affinity for input-order elements.

Moreover, the PTB corpus has undergone meticulous and consistent courses of annotation and tag. This capacitor resource is thought to represent a high-quality venue for assessing algorithms across different domains within NLP. Such consistency, together with the depth of linguistic detail within the PTB, would allow it to serve as the standard in benchmarking how advanced neural networks could approach the problem of language understanding and generation. Often, it is coupled with focusing techniques and auxiliary components such as memory mechanisms to probe models to see if performance regarding capturing and maintaining critical aspects of long-term contextual dependencies is improved.

This complexity and scope hence make PTB dataset an efficient medium to put the focus of GAM-RHNs up to test and its ability to extract intricate details of the input sequences, challenging to cope with real-world language patterns and structures.

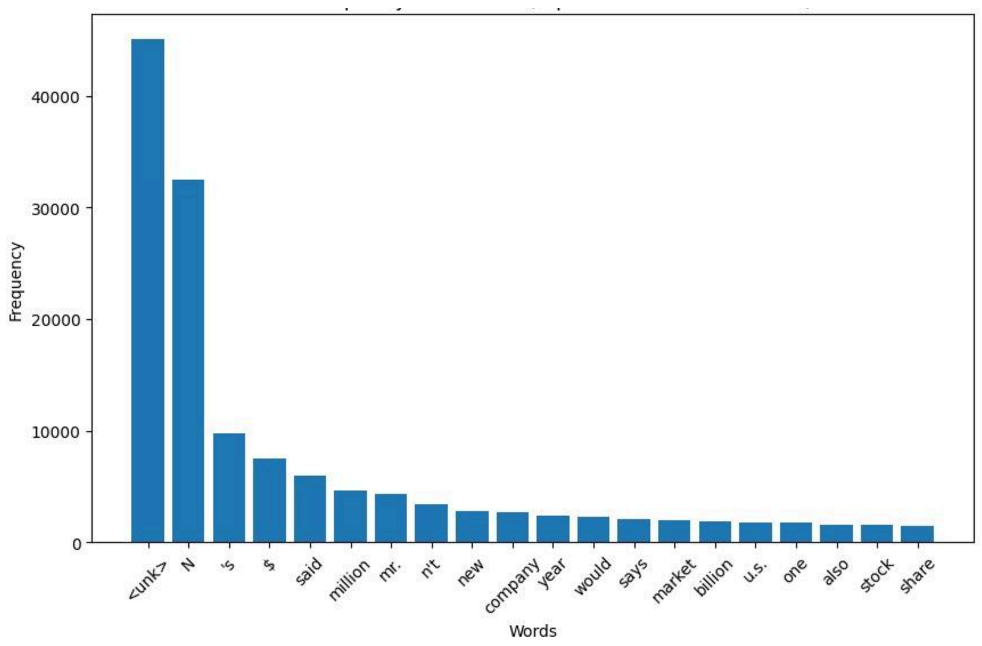

Fig 2. Word Frequency Distribution for Penn Tree Bank Dataset

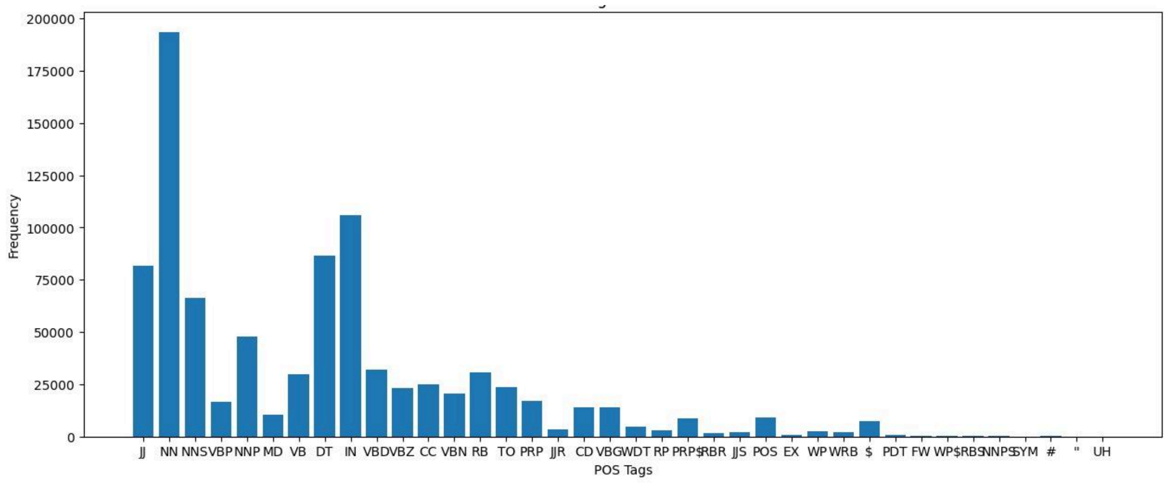

Fig 3. Part-of-Speech Tag Distribution for Penn Tree Bank Dataset

Table 1. Token Length Distribution by Dataset Split

split	average	min	max
train	21.097295	1.0	82.0
validation	20.887240	1.0	74.0
test	20.917043	1.0	77.0

3.3. Data preprocessing

Data preprocessing is one step taken by researchers prior to delegating their finalized Penn Treebank dataset to the GAM-RHN model for training. It first tokenizes the raw text into the predefined numerical codes upon the vocabulary set by the researcher. This allows for the model to work on text input in its most processable format. Each one of these varies in length and will require the same introductions in the batch through padding and truncation so that all sequences in a batch will be required for training purposes.

Normalisation creates a fairness distribution of tokens by avoiding any excess bias in any alphabet. This speeds up convergence and generalization. Then these tokens are embedded into dense vectors using the now-in-fashion word embeddings, which, like most fancy things in the world, come with magic explanations and conjunctions. The word embeddings can be initialised using pre-trained models to increment their understanding of the language.

Words not present in the training vocabulary are replaced with a special unknown token to maintain the model’s robustness when encountering unfamiliar terms. Batches of data are then created to optimise the training process, ensuring efficient use of computational resources and reducing training time.

This preprocessing pipeline ensures that the input data is in an optimal state, enabling the GAM-RHN model to effectively capture long-term dependencies and perform efficiently during training and evaluation.

3.4. Recurrent Highway Network

Recurrent neural networks have considerable problems in tracing long-term dependencies owing to vanishing gradient problems. This means that as the signal propagates through the deep network, it diminishes to zero and thus information is said to be lost over deep neural networks.

To mitigate this problem, RHNs introduce a highway layer that includes transitions along with deep state transitions and highway layers to accomplish a gating mechanism that regulates the flow of information through the neural network. The highway layer, through a gating mechanism, controls each portion of the data that undergoes nonlinear transformation while the rest is passed through unchanged. Thus, the highway would solve the vanishing gradient problem because a portion of the input data is very rarely modified, and hence the long-term dependencies can be captured, which would have otherwise been lost into the network due to the nonlinear transformations.

The highway layer contains 2 control gates, the transform gate and the carry gate. The transform gate determines what portion of the input undergoes nonlinear transformation, while the carry gate determines what portion of the input goes unmodified through the highway.

$$H(\mathbf{x}) = H(\mathbf{x}, \mathbf{0}) = G_\beta(\mathbf{x}) \odot \mathbf{x} + G_\alpha(\mathbf{x}) \odot F(\mathbf{x}).$$

where H(x) is the highway operator, G_β is the carry gate and G_α is the transform gate and F(x) is the non linear transformation.

This is how recurrent highway networks are able to capture more complex features over the long term and reduce the risk of vanishing gradients.

3.5. Grouped auxiliary memory

GAM is an extension of the RHN and it is designed to improve the model's ability to capture long term dependencies by augmenting the neural network with an external memory that can be read from and written to at any time. The main problem with the RHNs is that the data can be overwritten or attenuated at any time during recurrent state updates. GAM minimises the number of times the data is overwritten by the recurrent state updates and thus improves the model's performance to capture more complex features over the deep neural network.

The GAM is composed of memory units that are partitioned into groups. This grouping allows for selective updates and improved access to bits of memory. Each time step has a read from and write to memory-and on this basis the model decides which memory bundles to update through its learned addressing mechanism.

When writing to the GAM, an address vector is calculated and based on the address vector the specific groups are updated. The network generates an address vector using a softmax over groups to determine how much each memory group should be updated.

$$\mathbf{a}_t^i = \mathrm{softmax}(\mathbf{q}_t^i) \in \mathbb{R}^S, \quad i = 1, 2, \cdots, N.$$

a_t^i is the address vector and q_t^i is the weighted sum of the inputs of each group

$$\begin{aligned}\mathbf{m}_t &= H_{\mathrm{GAM}}^{S\times N}(\mathbf{m}_{t-1}, \mathbf{x}_t, \mathbf{s}_{t-1}^L) \\ &= (\mathbf{1} - \mathbf{a}_t^w) \odot \mathbf{m}_{t-1} + \mathbf{a}_t^w \odot (\mathbf{U}_{S\times N}\overline{\mathbf{m}}_t)\end{aligned}$$

m_t is the current memory state, m_{t-1} is the previous memory state and U_{SxN} is the duplicating matrix that updates the state of the memory

When reading from the GAM, the groups are read based on the read vector. The network generates a read vector by performing the weighted sum of the memory groups.

$$x_t^i = (\mathbf{a}_t^i)^\mathsf{T}\mathbf{m}_t^i \in \mathbb{R}, \quad i = 1, \cdots, N,$$

where x_t^i is the read vector a_t^i is the address vector and m_t^i is the state of the memory.

In this way the Grouped Auxiliary Memory enables the neural network to capture long term dependencies by providing an external memory that is selectively updated to avoid overwriting of the data during recurrent state updates.

3.6. Attention mechanism

Due to this, attention actually grants GAM-RHN the ability to give more importance to several parts of the input sequence, thus facilitating the comprehension of the relevant information and enhancing the performance of the model with greater success. Specifically, it assigns weights now to the different inputs, allowing the model to allocate more priority toward significant parts of the input rather than others. Furthermore, the introduction of attention mechanisms allows one of the typical limitations of RNNs, namely the long-range dependency problem, to be solved.

The attention mechanisms compute a score for inputs in the sequence. Different attention mechanisms compute the attention score using various strategies such as the dot product, additive functions, or multiple attention heads.

For Dot-Product Attention, the attention score is computed as the similarity between query and key vectors. The query is usually the concealed state of the decoder or the present state that requires attention. The crucial element is the concealed states of the input sequence, which help determine the significance of each segment of the sequence in relation to the current time step.

The attention score at time *t,i* is computed as:

$$\text{attention\_score}_{t,i} = q_t \cdot k_i$$

where q_t is the query vector at time *t*, and k_i is the key vector at time *i*. The attention weight α_i is calculated by performing the softmax function on the value of attention score at *t,i*:

$$\alpha_i = \text{softmax}(\text{attention\_score}_{t,i})$$

In Additive Attention, the attention score is obtained by a compatibility function between the query and the key. It combines the query and key through a neural network layer, with a learned weight.
The attention score $e_{t,i}$ is thus calculated as:

$$e_{t,i} = \tanh(W[q_t; k_i])$$

where W is the learned weight matrix, q_t is the query at time *t*, and k_i is the key at time *i*.
The attention weight is calculated by performing the softmax function on the score.

$$\alpha_i = \text{softmax}(e_{t,i})$$

Scaled Dot-Product Attention is similar to dot-product attention. However it includes a scaling factor, preventing large attention scores when the dimensionality of the vectors increase. Hereby, the value for attention score at *t,i*:

$$\text{attention\_score}_{t,i} = \frac{q_t \cdot k_i}{\sqrt{d_k}}$$

where q_t is the query vector at time *t*, k_i is the key vector at time *i*, and d_k is the dimensionality of the key vector. Here the attention weight is calculated by performing the softmax function.

$$\alpha_i = \text{softmax}(\text{attention\_score}_{t,i})$$

Once the attention weights α_i are calculated, a context vector c_t is computed as the weighted sum of all the hidden states in the input sequence:

$$c_t = \sum_{i=1}^{T} \alpha_i h_i$$

where h_i represents the hidden state at time *i*.

For Multi-Head Attention, the attention mechanism is applied to multiple projections of the query and key values. Each head uses the dot-product attention formula, with different projections for each head. For each head *h*, the attention score at time *t,i* is:

$$\text{attention\_score}^{h}_{t,i} = q^{h}_{t} \cdot k^{h}_{i}$$

where q_t^h is the query vector at time t for head h, and k_i^h is the key vector at time i. The attention weight is calculated using the softmax function for each head h.

$$\alpha_i^h = \text{softmax}(\text{attention\_score}_{t,i}^h)$$

The context vector c_t is generated for each head h as:

$$c_t^h = \sum_{i=1}^{T} \alpha_i^h v_i^h$$

where v_i^{h} represents the hidden state at time i, and α_i^{h} is the attention weight for the head h.
When the context vector for all heads are calculated, they are concatenated and projected back to the original hidden state:

$$c_t = \text{concat}(c_t^1, c_t^2, \ldots, c_t^H)$$

$$c_t = W_{\text{out}} c_t$$

where, c_t is the concatenated context vector, and W_{out} is a learnable weight matrix.

3.7. Evaluation Metrics

For language modelling tasks, we value metrics that would determine the capability of the model to predict the next word or token in a sequence. Perplexity is the main metric to measure the model's likelihood of predicting a sample. Perplexity is the exponential of the cross-entropy loss, with lower perplexity indicating better performance.
Cross-entropy loss expresses how confidently the model predicts the right class. The cross-entropy loss becomes large when model confidence is high about the wrong class or weak model confidence is offered for the correct class. Thus, it is vital that in such cases, a model predicts the correct class very confidently or is assured of its answer so that the value of the cross-entropy loss can be decreased.

$$\text{PPL}(X) = \exp\left\{ -\frac{1}{t} \sum_{i}^{t} \log p_\theta(x_i | x_{<i}) \right\}$$

The goal of the language-modelling approach is to assign high probabilities to the more likely sequences. The model with lower perplexity would be better able to describe the structure of the language and make superior predictions for the subsequent word in a sequence that is indicative of a low cross-entropy loss.
Accuracy can also be used to measure the percentage of correct predictions for the next token, but it is often secondary to perplexity in language modelling.

4. Experimental Results and Discussion:

4.1. Experimental Setup and Configuration:

The set environment is made with a Python version 3.10 that renders support to modern programming and machine-learning frameworks. PyTorch version 2.5.0+cu121 is installed, allowing for handy GPU acceleration and support for the deep-learning models. A Matplotlib version 3.8.0 is provided to make visualizations of high quality and detail out data analysis and representation of results. Furthermore, it offers the Datasets library in version 3.1.0 in order to make it easy to access and preprocess various datasets for machine-learning tasks. In addition, the Transformers library in version 4.44.2 supports the state-of-the-art NLP tasks.

4.2. Loss:

Intended as an evaluation on how accurately the model predicts the words, cross-entropy loss measures not simply how akin the predicted word is to its target, but also the confidence with which that word is predicted. Regardless of the correct prediction, a word correctly predicted but with lesser confidence still incurs a penalty.

$$L = -\frac{1}{N}\left[\sum_{j=1}^{N}\left[t_j \log(p_j) + (1 - t_j)\log(1 - p_j)\right]\right]$$

for N data points where t_i is the truth value taking a value 0 or 1 and p_i is the Softmax probability for the i^{th} data point.

Cross entropy loss is an effective loss measure to gauge the effectiveness of the model in its prediction since it accounts for the linguistically equivalent words as predicted by the model. The loss is also a measure of the confidence of the model in its prediction and thus forces the model to improve the softmax probability in predicting the correct word.

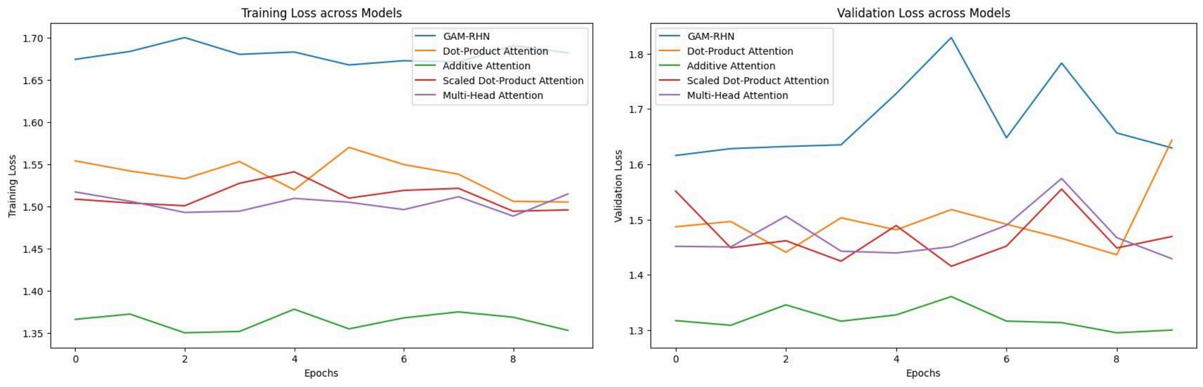

Fig 4(a). Training loss across models (left) 4(b). Validation Loss across Models (right)

4.3. Accuracy:

For a sequence-to-sequence task, the goal is to map an input to an output sequence. In the given task, accuracy is defined as the measure of correctly predicted words. The model generates a sequence of predicted words that corresponds to the target sequence. Each predicted word is matched against the target word.

$$\text{Accuracy} = \frac{\text{Number of Correctly Predicted Words}}{\text{Total Number of Words in the Target Sequence}}$$

A strong divergence has been shown by the model in making predictions of words that are linguistically equivalent words to the target word. Because the predicted word is established as not exactly the same, it therefore is classified as an incorrect prediction. This leads to the conclusion that the performance of the model cannot satisfactorily be expressed in terms of accuracy alone.

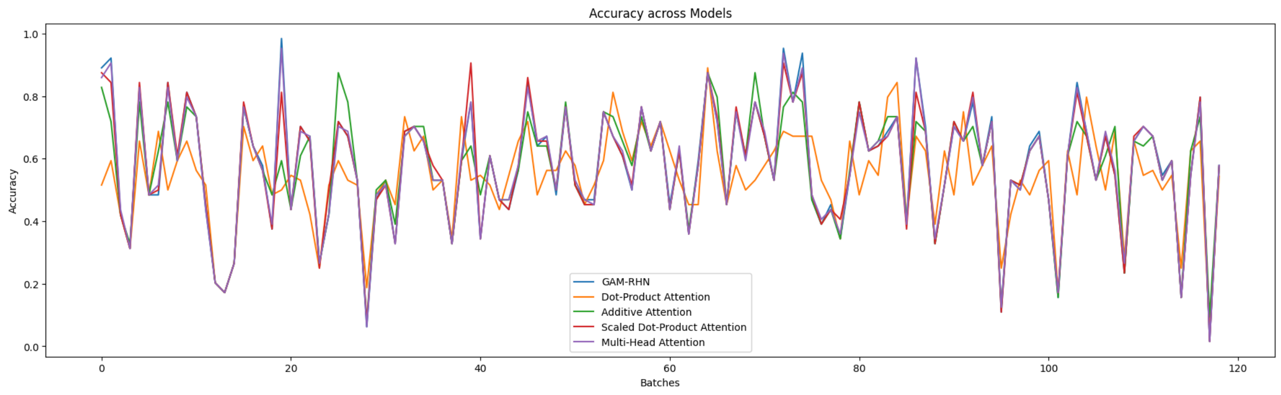

Fig 5. Accuracy across Models

4.4. Perplexity:

Perplexity is defined as an exponential of cross-entropy loss as a measure of language models when it comes to text generation and machine translation tasks. The exponential plays a crucial role in transforming the cross entropy loss into a more interpretable and intuitive metric. It provides an indication of how well a probabilistic model predicts the words.

$$\text{PPL}(X) = \exp\left\{-\frac{1}{t}\sum_{i}^{t}\log p_\theta(x_i|x_{<i})\right\}$$

The cross-entropy loss indicates the likelihood that the model has predicted the right word. Perplexity determines the scope of the model to predict the next word. Exponential improbability is relevant since it produces a more interpretable result by giving the probability distribution of the model with respect to its guess of the right word.

If the perplexity is high then it is indicated that the model is uncertain and has a high number of possible choices for each token. If the perplexity is low then the model is more confident and the choices are limited and the probability that the model predicts the correct word is higher. The exponential function amplifies the effect of larger uncertainties or poor predictions.

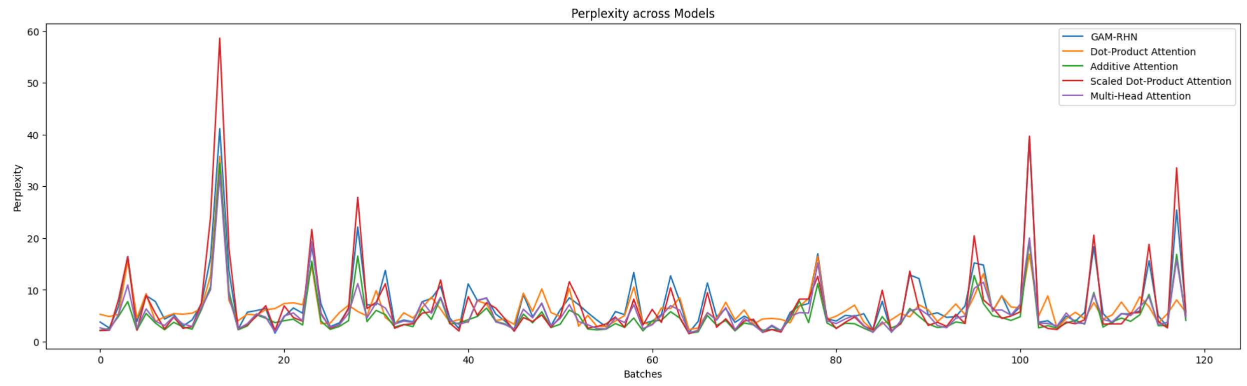

Fig 6. Perplexity across Models

Table 2. Comparative Values of Base Model vs Proposed Models

Model	Train Loss	Val Loss	Accuracy	Perplexity
Base GAM-RHN model	1.6806	1.6786	0.5785	7.3168
Proposed Dot-Product Attention	1.5371	1.4965	0.544	6.3415
Proposed Additive Attention	1.3639	1.3202	0.5738	4.8876
Proposed Scaled Dot-Product Attention	1.5123	1.4717	0.5772	6.9117
Proposed Multi-Head Attention	1.5036	1.4702	0.5781	5.4762

5. Conclusion and Future Work:

In conclusion, upon comparison of GAM-RHN with the various attention mechanisms, it is established that attention-based models show competitive or better results in comparison to the GAM-RHN model. Furthermore, Multi-Head Attention surfaces more frequently into higher peaks of accuracy than the rest of the models throughout the number of batches. While considering the Perplexity across models, it is notable that the Scaled Dot-Product Attention demonstrates higher peaks indicating that the model encounters sequences in the data that are more difficult to predict accurately. On average, the additive and Multi_head attention based models display lower perplexities and are thus able to learn the complex dependencies in the data. While GAM-RHN is effective, it is not shown to outperform the attention based models indicating a clear advantage of the attention based models over GAM-RHN.

These findings pave the way for future research involving integrating the existing RHN model with transformer architectures which are adept at capturing global dependencies through the use of self attention mechanisms. This hybrid approach would provide an excellent opportunity for the combination of RHN strengths in its ability to propagate gradients throughout the network, along with that of the transformer framework for learning on long-term dependencies in a focused manner. Another domain of exploration could involve multi-scale attention mechanisms: capturing accurate local and global dependencies so as to perform better on multifaceted tasks that vary in terms of granularity of sequences

References:

[1] Graves, A. (2013). Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850
[2] De Mulder, W., Bethard, S., & Moens, M. F. (2015). A survey on the application of recurrent neural networks to statistical language modeling. Computer Speech & Language, 30(1), 61-98.
[3] Mikolov, T., & Zweig, G. (2012, December). Context dependent recurrent neural network language model. In 2012 IEEE Spoken Language Technology Workshop (SLT) (pp. 234-239). IEEE.
[4] Zilly, J. G., Srivastava, R. K., Koutnık, J., & Schmidhuber, J. (2017, July). Recurrent highway networks. In International conference on machine learning (pp. 4189-4198). PMLR.
[5] Zhang, C., Nguyen, T., Sah, S., Ptucha, R., Loui, A., & Salvaggio, C. (2017, September). Batch-normalized recurrent highway networks. In 2017 IEEE International Conference on Image Processing (ICIP) (pp. 640-644). IEEE.
[6] Pundak, G., & Sainath, T. N. (2017, August). Highway-LSTM and Recurrent Highway Networks for Speech Recognition. In Interspeech (pp. 1303-1307).
[7] Luo, W., & Yu, F. (2020). Learning longer-term dependencies via grouped distributor unit. Neurocomputing, 412, 406-415.
[8] Zang, K., & Ma, Z. (2020). Automatic modulation classification based on hierarchical recurrent neural networks with grouped auxiliary memory. IEEE Access, 8, 213052-213061.
[9] Wang, F., & Tax, D. M. (2016). Survey on the attention based RNN model and its applications in computer vision. arXiv preprint arXiv:1601.06823.
[10] Soydaner, D. (2022). Attention mechanism in neural networks: where it comes and where it goes. Neural Computing and Applications, 34(16), 13371-13385.
[11] Mujika, A., Meier, F., & Steger, A. (2017). Fast-slow recurrent neural networks. Advances in Neural Information Processing Systems, 30.
[12] Choi, I., Park, J., & Sung, W. (2018, September). Character-level Language Modeling with Gated Hierarchical Recurrent Neural Networks. In INTERSPEECH (pp. 411-415).
[13] Luo, W., & Yu, F. (2019). Recurrent highway networks with grouped auxiliary memory. IEEE Access, 7, 182037-182049.
[14] Chowanda, A., & Chowanda, A. D. (2018). Generative Indonesian conversation model using recurrent neural network with attention mechanism. Procedia Computer Science, 135, 433-440.
[15] Mei, H., Bansal, M., & Walter, M. (2017, February). Coherent dialogue with attention-based language models. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 31, No. 1).