Machine Translation using RNN

Study Project 2

Viraj Savaliya
ASU ID: 1217678787
Email: Viraj.Savaliya@asu.edu
Arizona State University, Tempe, AZ

Raunak
ASU ID: 1217240245
Email: rraunak@asu.edu
Arizona State University, Tempe, AZ

Nayankumar Patel
ASU ID: 1217420191
Email: npatel48@asu.edu
Arizona State University, Tempe, AZ

***Abstract*—Recurrent neural networks has proved useful in implementing language translation models due to it's ability to map sequences to sequences. In this project, we are looking into traditional machine translation methods like Recurrent Neural Network and their performance improvised later on by introducing LSTM or GRU. we have described a gated hidden unit also known as Gated Recurrent Unit(GRU) architecture to encode an input sequence of words and, an another multilayered GRU with an attention layer to decode the encoded sequence of words, which is essentially an encoder-decoder model with GRU attention embedding units. The GRU learns sensible phrase and sentence representations that are crucial to the word order and are unaffected to the change in active and passive voices. Qualitatively, we depict the model learns a syntactically and semantically meaningful representation of linguistic phrases.**

***Index Terms*—RNN, GRU, Encoder-Decoder, Neural Network**

I. Introduction

In Natural Language Processing(NLP) language translation is one of the most prominent features and is widely used in various applications in our day to day lives. Languages are composed of sequences of words, which poses a challenge to learn or unlearn something depending on the previous input so that a sequence of words can be identified to be translated to a different language. In this project, we are going to study an application of Gated Recurrent Unit(GRU) architecture with attention to resolve the sequence to sequence problem. The described recurrent neural network(RNN), encoder-decoder has two RNNs, one of them acts as an encoder and the other as a decoder [1]. The encoder maps an input sequence to a vector of a fixed dimension and the decoder maps the target sequence from the fixed dimension vector. Both the networks are trained simultaneously with attention to obtain the best results. The trained RNN encoder-decoder network is evaluated on the task of German to English translation. An encoder-decoder model which takes an input sequence as "ABC" and produces "WXYZ" as the output sequence is shown in Fig. 1. The model stops the prediction on finding the End of Sentence(EOS) token [2]. It is clear from the Fig. 1, that GRU reads the input in reverse, which enables any short term dependencies to be discovered hence, resulting in higher efficiency.

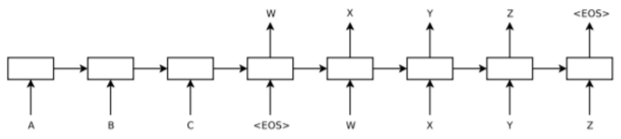


Fig. 1. **Encoder-Decoder**

II. Application

The communication ability of the humans with one another is the fundamental part of the existence of being human. In the world, there are 7000 different languages in existence. Since, day by day our world is becoming more and more connected, machine translation provides a cultural and economic connection between people of different ethnic groups and regions. some of the examples are,

1. Commerce: Foreign good purchase, travel
2. Education: Collaboration in research, idea sharing
3. Media: Information access via search, social networking

To meet these demands, technology companies are heavily investing in research. Recent advancements in convolutional neural networks(CNN) and residual neural networks(RNN) has yielded major improvements in quality of the translation. Today, hundreds of different languages can be translated. However, while machine translation has made lot of progress, it is still not perfect,

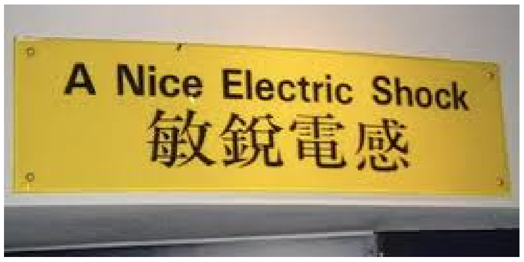


Fig. 2. **An example of a bad translation**

The use of software to translate text or speech from one language to another is known as machine translation. Basically, machine translation substitutes word of one language to another but, that doesn't mean that translation would be performed successfully because there are many intricate details that must be taken care of such as, one language doesn't has similar words in another language and one word could have more than one meaning.

The human translation process consists of two parts,

1. Firstly, we decode the meaning of the language to be translated.

2. Secondly, we re-encode this meaning in the translated language.

Behind this inherently simple procedure, a complex cognitive operation is hidden. All the features of the text must be interpreted and analyzed to decode the meaning of the text in the language to be translated. This process consists of deep knowledge of the grammar, idioms, semantics, syntax and culture of its speakers of the language to be translated. The same level of details are required to re-encode the meaning to another language. Hence, therein lies the challenge in machine translation.

III. Architecture

When Machine Translation comes into the picture - Recurrent Neural Networks(RNN) is default choice. In Machine Translation applications standard RNN has vanishing gradient problem and network is not able to memorise words from deep layers which results in to poor translation. This problem can be overcome by using Long Short Term Memory Unit(LSTM) or Gated Recurrent Unit(GRU). Base architecture of all the discussed networks are same with addition to some control layers or blocks to resolve the issues faced in other architecture. In a machine translation problem a sequence of words in one language is given as input and in output those sequence of words being translated into different language. Similar way Recurrent Neural Network are designed to take sequence in one language and generate output in other language. Let's look into the human analogy when an individual reads or write some thing in a specific language.

When we start reading a book, from first line we starts remembering pieces of information from all previous content what we read from the book. For this we need to store some important information and use those information to create context in future when we advance our reading to understand future sentences and their context. This kind of problem need an architecture which remembers information from previous layer and helps to create context for consecutive layers. In Machine learning application Convolutional Neural Networks are simple feed forward neural network where there is time series context. On the other hand RNN will provide such time series context and provide a memory component with in the architecture to resolve the problems. For a RNN networks there are multiple model topology possible like one to many, many to one and many to many. If we look into the Language Translation problem which takes multiple inputs from a specific language and generate translated outputs in to different language which seems a many to many architecture of RNN.

This reference model represents a machine learning application model where a sequences of words in English is converted into French using Recurrent Neural Network. Here, in the model it is visible that it has sequence to sequence network architecture which process data at consecutive time stamps. In Machine translation problem encoder and decoder network are linked by sequence to sequence model. As we have seen earlier encoder helps to create contexts/state. Generated context is used by decoder RNN and generate the output sequence.

If we observe the encoder and decoder sequences, they contains loops which process input word sequence and generate the outputs. In the image at different time steps, it gives an inputs and generate outputs. Encoder reads input words

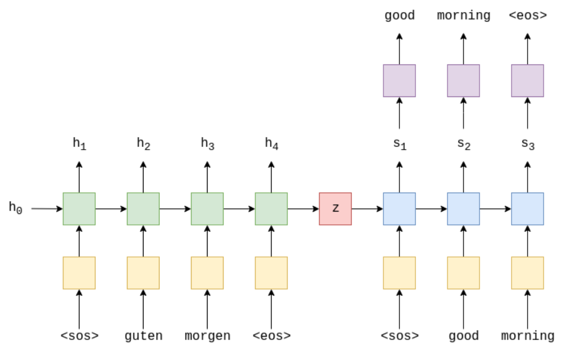


Fig. 3. **Machine Translation Model with RNN Cells**

at different time step and generate a single context vector. At every time step it takes current input words and also takes inputs from the earlier time steps which is similar to human way of understanding and remembering contexts while reading or listening. At encoder level entire input sequence is represented by a context vector. Hidden states between layer plays vital role in the learning capacity of the model with greater computational unit. Later on at decoder level, generated context vector is used to translate the input and generate the output. the When human starts processing language, some times it needs future contexts apart from historical contexts to understand and process the sentence more accurately. This also can be achieved with the Recurrent Neural Network using future context with better model accuracy. In Standard RNN architecture, by adding reverse layer which process reversed input word sequence and use future contexts as well to train the weights.

Recurrent Neural Network is a standard feed forward network in addition to considering inputs from earlier time step. For given inputs $(x_1, x_2, x_3.....x_T)$ for total T time steps - RNN find the output sequence $(y_1, y_2, y_3.....y_T)$ by doing following calculation.

$$h_t = sigm(W^{hx}x_t + W^{hh}h_(t-1))$$

$$y_t = W^{yh}h_t$$

h_t is the current time step transformation which takes current inputs and previous non linear time step inputs as well. At specific time step output is calculated. At the end of the final hidden state at encoder - output h_T is considered as context vector Z. This vector represents the input source sequence. Decoder architecture works similar way as we discussed earlier which takes output from previous layer which is the first translated word and use it for historical context.

If we observe the historical context pattern while reading or listening - all the historical context data is not useful. There should be some control mechanism with in the network to control the flow of information. These control mechanism allows relevant information only and discard the other. In some application there is possibility of longer historical contexts where current cell output not only depends on previous one but also 30 cells earlier. Longer historical context and vanishing gradient problems scatter the RNN performance which can be resolved by Long Short Term Memory Unit(LSTM) or Gated Recurrent Unit(GRU). In case of GRU - control mechanism is implemented with the help of an update gate and reset gate. How much information passed from previous state to the next state is controlled by update gate on other hand reset gate decides how much of the past information to clear. In case of LSTM or GRU - generated vector context contains relevant information only which needs to be transferred to decoder sequence.

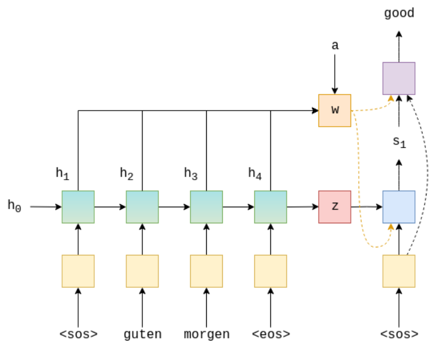


Fig. 4. **GRU with Attention in the Encoder-Decoder**

In GRU, a signle context vector is used to map the historical context with the future context. With these fix length vector context - it is difficult to remember long sentences. This issue can be solved by adding more context vectors to the network. Here, we are giving attention to different generated contexts. GRU with Attention, calculates the attention vector similar to input source length where each element is has value between 0 and 1. With the attention vector with original context vector weighted source vector is calculated.

$$w = \sum_i a_i h_i$$

All new weights are updated at every time step of the decoding sequence to predict the outputs.

For Implementing and training Recurrent Neural Network for machine learning applications, resources like Keras and TensorFlow are used as Front end and back end computing libraries on Python environment. All these application can be run on general computing architecture with higher resources based on the architectural complexity. Raw input data needs different tools to process it in the model understandable vector input formats. The experiment performed by the authors for training machine learning problem with enough video memory of GPU where a mid range GPU like Nvidia TITAN RTX – 24GB Memory. With lesser memory - training of the model can be achieved with higher training time. Different hyper parameters like initial learning rate, size of hidden layers, number of layers, batch size, training samples, training epochs, decay rate along with input data processing decides the overall training time and model training and testing accuracy. Selecting specific architecture also decides the output accuracy based on the application use case and RNN Model discussed earlier like RNN, LSTM, GRU, LSTM+Attention, GRU+Attention.

IV. Comparison

Sequence to sequence models using RNN are used to solve supervised learning problems which are based on time series in input and output layers. Language translation is a time series model as the input will be a sequence of words in one language and the output will be a sequence generated in the desired language. These are many-to-many type of RNN model and there are various architectures which can be used to solve the problem [3]. The selection of architecture will depend upon the training time of the model, complexity, scalability, accuracy and other performance metrics. There isn't any clear or proven evidence yet for an architecture which outperforms the rest for all cases. Architecture is selected depending on the application and resources available.

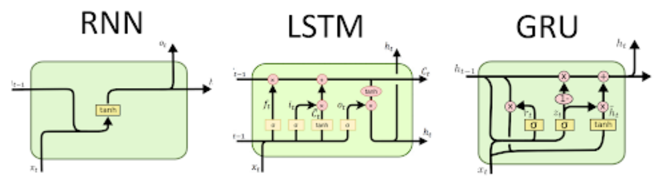


Fig. 5. **Basic Architectures of RNN Units [4]**

A. Architecture

The architecture for encoder-decoder model consists of an input sequence (sentence) and then converted into a context vector by the encoder and later converted into target sequence (sentence) by the decoder. The RNN architecture consists of small units embedded which are either simple RNN units, GRU(Gated Recurrent Unit) units or LSTM(Long Short-Term Memory) units shown in Fig. 5. A simple unit is just a multiplication of input (x_t) and previous output which then passed through $Tanh$ activation function. There are no gates present in this unit. GRU has an update gate which decides if the previous output should be passed on to the next cell or not. The reset gate in GRU is used to decide how much of the past information can be deleted. The third type LSTM consists of 3 gates, forget, input and output gates. Forget gate is similar to update gate and decides which information to throw away by using a sigmoid function. The input gate selects relevant information to add to current step and the output gate determines the next hidden state. The architecture in this report used to solve the language translation model consists of GRU as it does not require a memory unit, is computationally more efficient and less complex, provides performance on par with LSTM and also because the application does not require to remember very long sequences of data which is the main advantage of LSTM.

Another important modification in architecture is usage of Attention layer [5]. The attention layer is an extension in the original encoder-decoder model and is used to align and then translate the data jointly. This layer aligns the input sequence in the order of its relevance to the output in each time-step [6]. It makes the model focus more on the relevant information and then accordingly generate output. A score is generated for each encoded input as per the output generated by the decoder in each step. The score is calculated based on the output of the decoder from the previous step. Attention layer searches for source positions in the sentence where most relevant information is generated from and based on the context vectors associated with those source positions and all the previous generated target words it predicts the next target word. The architectures can also contain teacher forcing which makes the ground truth token as input to next step instead of highest predicted token.

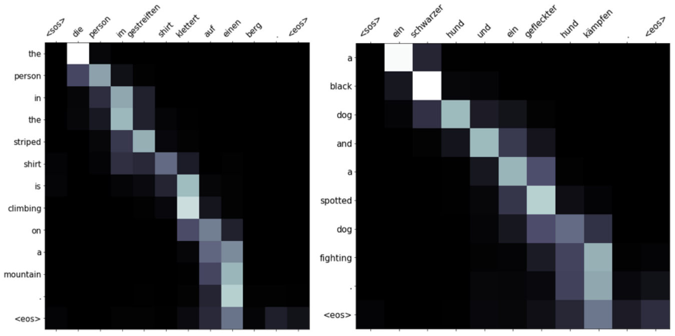
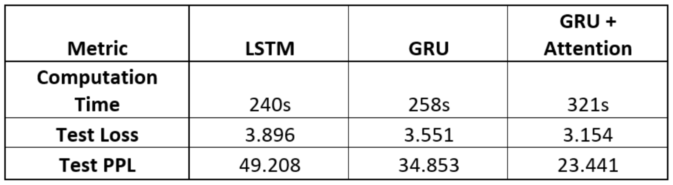


Fig. 6. **Training and Testing Translation Inference examples**

B. Implementation

In the architecture using GRU without attention [7] consists of a single layer of GRU in encoder as well as decoder instead of multiple layers of LSTM in order to reduce the training time and complexity of the model. GRU unit does not take dropout as an argument as there is only one layer and it only generates the hidden state and there is no cell state generated like in LSTM. The encoder takes in the input sequence passed on from the embedding layer and recurrently calculates the hidden states and generates the context vector. In the decoder information compression is alleviated. The GRU unit in the decoder takes the embedded target token, previous hidden state and the entire context vector to generate new hidden state. Here information compression is achieved as there is no need for information about the source sequence as it is always available and also as the layer can directly see what tokens are generated.

The architecture which we are using for the application is similar to the GRU architecture and has an additional attention layer [8], bidirectional GRU network and uses packed padded sequences and masking. The underlying methodology remains the same here as well of what the encoder and decoder implements. The encoder consists of bidirectional GRU network i.e. forward and backward RNN which passes the sentence from either direction respectively. Two context vectors are generated from the encoder which are concatenated and passed over the tanh activation function and then sent to the decoder network. The attention layer takes what all is decoded so far and all of what is encoded and produces a vector which tells us about where to pay most attention to predict the next work correctly. In the decoder the attention vector is used to create a weighted source vector and along with the previous hidden state and input word sequence it then predicts the next target word. Packed padding sequence is used to create uniformity and avoid unnecessary computations and Masking in the network is also implemented to force the model the network to ignore certain values. The inference done to see where the model pays most attention can be seen in Fig. 6 where image on the left is a training example and the one on the right is a testing example. The model is implemented on pytorch 1.4 and torchtext 0.5 using python 3.6 [9].



Metric	LSTM	GRU	GRU + Attention
Computation Time	240s	258s	321s
Test Loss	3.896	3.551	3.154
Test PPL	49.208	34.853	23.441

Fig. 7. **RNN performance evaluation**

C. Evaluation

The two models are evaluated using the Multi30k dataset which consists of around 30k parallel English, German and French sentences with approx 12 words per sentence [10]. The dataset is divided into training, testing and validation sets. The different performance metrics for LSTM model, GRU model and GRU model with attention can be seen in the Fig. 7. The model with Attention layer achieves a BLEU score of 29.04 which is comparable to the original model in the paper and better than the first two models but is still low as the size of the model and dataset for implementation is small. Also, It can be clearly seen the model architecture with GRU having attention performs the best and is most accurate being quite robust at the same time. In addition, inferencing can also be done on the model to observe how attention affects the translation of sequences.

V. Conclusion

In this study project, we solve Natural Language processing (NLP) problem of language translation in text. This is an extremely important problem as there are around 7000 languages around the world and with increasing cultural and economic connections between people from different regions, it cannot be ensured always that there exists a common language which everyone can understand. Therefore, language translators prove to be of immense help here making the connections possible even without having common language between the people.

Language translations is done using a sequence to sequence RNN model which is basically an encoder-decoder model. We are using GRU based single layer RNN network for encoder as well as decoder. Where, the encoder is bidirectional and is followed by an activation layer and then decoder network. The model also uses packed padded sequences and masking to further improvise it's performance. The selection of the model is then justified by comparing its performance with multi-layered LSTM model and single layer GRU model without attention.

RNN model for solving any problem needs a selection of base units and justified performance enhancement techniques embedded into it for the application. The evaluations provided in Fig. 7 clearly indicate that for language translations between German, English and French based on the dataset consisting of commonly spoken sentences, the GRU model with attention outperforms the other models and takes proper advantage of the benefits of GRU and enhancement techniques added.

Contribution

Name	Sections
Raunak	Abstract, Introduction, Application
Nayankumar Patel	Architecture
Viraj Savaliya	Comparison, Conclusion

References

[1] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. [Online]. Available: https://www.aclweb.org/anthology/D14-1179

[2] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'14. Cambridge, MA, USA: MIT Press, 2014, p. 3104–3112.

[3] R. Shrott, "Sequence models by andrew ng-11 lessons learned," Feb 2019. [Online]. Available: https://towardsdatascience.com/sequence-models-by-andrew-ng-11-lessons-learned-c62fb1d3485b

[4] S. Rathor, "Simple rnn vs gru vs lstm :- difference lies in more flexible control," Jun 2018. [Online]. Available: https://medium.com/@saurabh.rathor092/simple-rnn-vs-gru-vs-lstm-difference-lies-in-more-flexible-control

[5] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, cite arxiv:1409.0473Comment: Accepted at ICLR 2015 as oral presentation. [Online]. Available: http://arxiv.org/abs/1409.0473

[6] G. Loye, "Attention mechanism," Jan 2020. [Online]. Available: https://blog.floydhub.com/attention-mechanism/

[7] Z. Chen, "Install cuda 9.0 and cudnn 7.0 for tensorflow/pytorch (gpu) on ubuntu 16.04," Aug 2019. [Online]. Available: https://medium.com/repro-repo/install-cuda-and-cudnn-for-tensorflow-gpu-on-ubuntu-79306e4ac04e

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017. [Online]. Available: https://arxiv.org/pdf/1706.03762.pdf

[9] Spro, "spro/practical-pytorch." [Online]. Available: https://github.com/spro/practical-pytorch

[10] Bentrevett, "bentrevett/pytorch-seq2seq." [Online]. Available: https://github.com/bentrevett/pytorch-seq2seq

By completing this form, all team members agree to be responsible for the tasks divided below towards completing the assignment.

	Percentage	Identify research topic/reference(s) (team to provide further detailed division)	study deep architecture & design (R1 & R2) (further detailed division)	compare with another deep architecture (further detailed division)	compare dfferent development tools (further detailed division)
Viraj Savaliya	33.33	Understanding the issues in Machine translations, The Key challenges faced for translation of languages, Finidng different architectures used to solve the problem, Understanding the Base architectures of RNN, Understanding various Optimizations techniques, Comparing various models built and figuring the best model for daily spoken language phrases	Figuring the main challneges in language translation, Understanding the key elements of the architecture to solve the problem, Figuring the major implementation and evaluation steps for the model	Comparing the base architectures of any RNN model, Understanding the architectural and implementation differences involved in various model, Summarizing the various advantages and effects of different optimization techniques, Comparing the evaulated resulgts for architectures and providing evidence for the best performance of architecture used to solve the probelm	NA
Raunak	33.33	Finding real life problems and their possible solutions using RNN, Understanding machine translation, it's importance and applications in the real world. Identification of different architectures to solve the problem, Finding out the ineffeciencies in the current machine translation architecture, model selection to describe and compare based on advantages and disadvantages of different approaches	Understanding the application-machine translation, learning methods other than RNN to solve the machine translation problem. Learning the evolution of different architecture models in residual neural network and internal architecture.Finidng out the issues in existing models and the solution achieved in subsequent models.	Evaluating the alternate RNN architectures for the initial comparison and understanding the attention model for higher accuracy	NA
Nayankumar Patel	33.33	Finding the problem and existing application of machine learing in real world. Figuring out how effective and efficient the problem solution should be?	Undertanding the basic RNN architecture in the generic application context. Relating it with the specific application - Machine Trnslation and how it can be mapped to Recurrent Neural Network. Reading papers for LSTM and GRU and it's application context.	Observing the reports and real time application where different RNN model used with reference to Machine Trnaslation problem. How and Why a perticular model architecture performs better or worst with specific applications. Comparing RNN architecture flavours and additional complexities to add feature to base architecture.	NA

MACHINE TRANSLATION USING RNN

- Viraj Savaliya

- **APPLICATION -** Translation of text from one language to another

  Brings the world closer and connects people of various cultures

- **PROBLEMS –** A. Necessary Information Compression

  B. Phrase Representations and Alignment

  C. Sentence Length

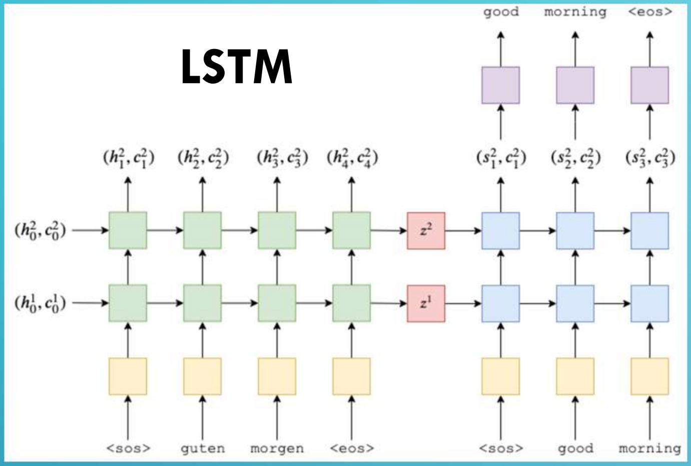
  D. Context Translation

  E. Computational Complexity, Speed, Resource requirements

- **ARCHITECTURES –** Basic units of LSTM, GRU or Standard RNN cell

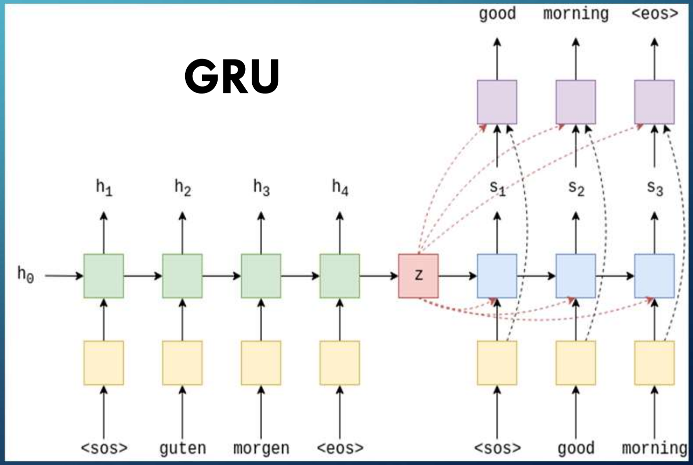
  Evolved with more layers added, connections of cells, optimization techniques, etc

RNN Architectures



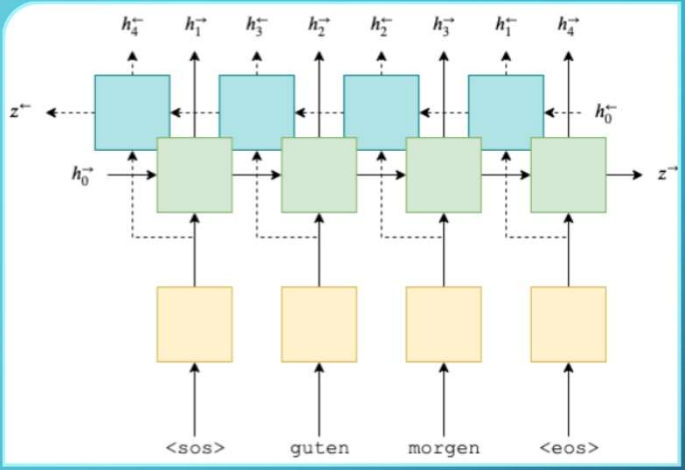
- 2-layer LSTM Encoder & Decoder
- 3-gates – Forget, input and output
- Cell state per time step and Dropout used to prevent overfitting
- Information stored in hidden layers
- Decoder takes target token and previous hidden state

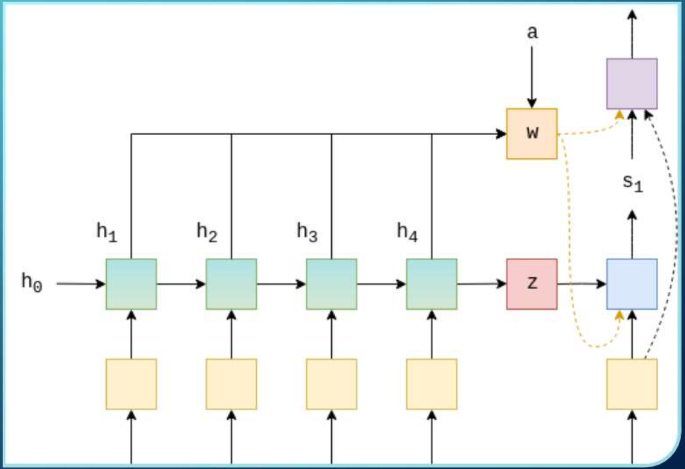
- Single Layer GRU Encoder & Decoder
- 2 gates – Reset and Update
- No Cell State and Dropout
- Information compressed
- Decoder takes target token, previous hidden state and context vector



GRU + Attention Layer RNN Model

- Baseline - GRU Architecture
- Bidirectional Encoder RNN
- Context vectors concatenated
- Attention layer to focus on important segments
- Attention – All encoded and Decoded so far
- Decoder – Weighted source vector, previous hidden state and target token
- Linear layer makes prediction
- Packed padded sequences – skip padding token
- Masking – Ignore certain value





REQUIREMENTS

- Online Source Code [1], [2]
- Datasets – Multi30k (30k English, French, German sentences; 12 words each)
- Environments Used:
  a. Google Colab
  b. TensorFlow
  c. PyTorch
  d. TorchText
  e. Python
- Time required: 321s approx. (10 epochs)

RESULTS

- Multi30k dataset used
- LSTM Model : Time - 240s, Test Loss - 3.896, Test PPL - 49.208
- GRU Model : Time - 258s, Test Loss – 3.551, Test PPL – 34.853
- GRU + Attention Model : Time - 321s, Test Loss – 3.154, Test PPL – 23.441, BLEU – 29.04
- GRU model with Attention layer provides best performance as it uses attention vector to pay more attention to important segments as well as creates uniformity and reduces computation

REFERENCES

- https://github.com/bentrevett/pytorch-seq2seq
- https://github.com/spro/practical-pytorch/blob/master/seq2seq-translation/seq2seq-translation.ipynb
- Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR, abs/1409.0473*.

Presentation Video:
https://drive.google.com/file/d/11fZCBi6gfXMWqTZT-3R0UWnOdmobRjlR/view?usp=sharing