Project Final Report

Topic: Parallel Attention Mechanisms in Neural Machine Translation

Team Members:

Ravi Teja Kolla

Introduction:

1. This topic is important in terms of training the data in less time. In the current world we have data in terms of millions. To train this amounts of data we need a way to reduce training time compared to generic models that were built and being used. So Making parallel attention mechanisms is the way to go.
2. Main applications of this will come in the case of google translator when they need to add new language to the search. This would be a best way to go
3. The paper mainly stresses on how to make the training distributed and how can we achieve that and results from the training

Background:

This is the paper:

<https://ieeexplore.ieee.org/document/8614113>

1. Main challenge : the main challenge of the project is to show that performance in training the data in Parallel Attention Mechanisms compared to traditional MLT where they use sequential stacked encoder and decoder models to translate the language.
2. As for as the implentation goes, We used two or more different parallel models to train the data then use an additive encoder to combine the learning data then use decoding layers and translate the data.
3. In the paper they have shown the different types of the architectures used and explained to show the performance in the training time.

A screenshot of a cell phone

Description automatically generated

1. This was a generic paper on parallel attention mechanisms. There wasn’t much discussion online. But according the paper I have read : they discussed mainly on comparison between sequential model and parallel attention model
2. Even in the citation papers I have read I didn’t see any particular interest or deep discussion on parallel attention mechanisms. In this particular paper they used different models to implement the idea like transformer model and cho’s model etc.
3. Well I don’t say my work is different from them. The only thing is tried to implement on gpu’s rather on multiple workers or instances in the cluster

Project:

1. My approach to problem was simple. First I learned how Encoder and Decoder model works . Then I learned to implement attention in the model. This was my 50% of the project. Then I started to learn how to do distributed training and how to implement distributed training on this problem which is where I had most of issues trying to implement
2. As I am single member of this project. Everything is done by me.
3. Well as far as the project goes the sequential model is implemented perfectly. The part that didn’t work for me is the parallel attention mechanism. I tried to implement this using Tensorflow distributed strategy api. But at first I had gpu crash issues or I had issues with parallel optimization. I tried to implement using threads. But this is a lot of parallel data synchronization across threads. This takes a lot of time to implement. I looked at the transformer model to see if I could get any idea. I Tried as much as I can considering my experience in this field

Results:

This is the example and plot of the seq-seq model:

translate(u'hace mucho frio aqui.')

reference = ['it','is','too','fast']

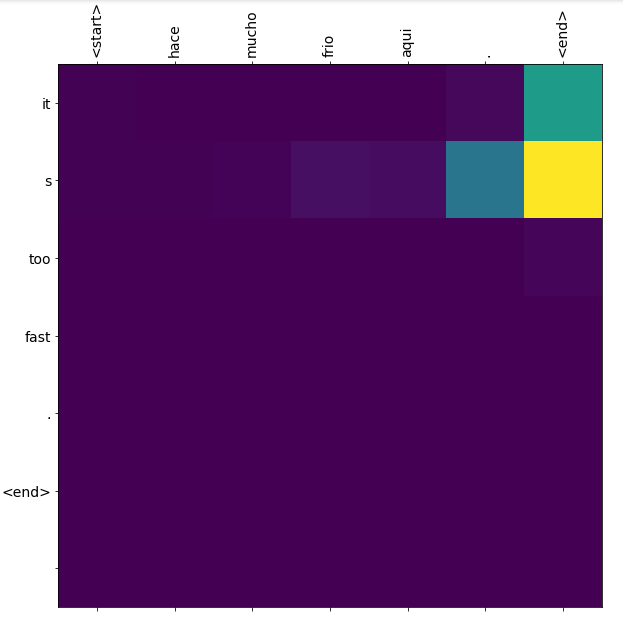
hypo = ["it","s",'too','fast']

bleu = nltk.translate.bleu\_score.sentence\_bleu([reference], hypo)

print (bleu)

Input: <start> hace mucho frio aqui . <end>

Predicted translation: it s too fast . <end>

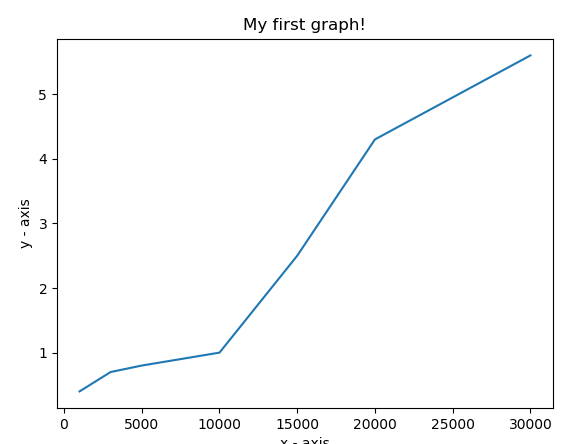


The bleu score for the seq-seq to model came around :

0.7 plus or minus 0.1

This is a decent bleu score.

So now moving onto the training time:



X-axis represents : num of sentences trained on the sequential model

Y-axis represents: num of minutes it took to train the model

As far as the Parallel training goes I couldn’t implement the entire model in parallel mode. So I don’t have any results to show for this part. All the results I have is above.

Summary:

Well as far as the problem goes . Everything makes sense on why choosing parallel training is worth it compared to sequential model. But I think even then it takes lot of time to develop a model and assign resources ,manage resources etc. The main point of this project is to keep the performance of the model same while reducing the run time of the training.

As you can see from the graph it takes almost 5-6 mintutes to run just 30000 sentences. But with the data we have which is around millions . It takes a lot of time to just run the model. So even if we reduce the training time by 20%. It means from hours – days in real world time which is pretty good.

Conclusion:

The only thing I accomplish by doing this project was I am confident with Encoder-Decoder model along with the basics of LTSM and GRU and why are they used in real world scenario.

As far as the work that ought to be complete :

I would have taken more assistance from the seniors or professors to accomplish this task. I would have kept little more time digging into tensor-flow repos for more info on distributed training.

Future work:

Future work to this would be to complete the parallel model and make it work and do any modifications on any sequential steps involved in the model.

References:

<https://ieeexplore.ieee.org/document/8614113>

Other references:

<https://www.tensorflow.org/guide/distributed_training>

<https://www.tensorflow.org/guide/gpu>