[**https://openreview.net/pdf?id=B14TlG-RW**](https://openreview.net/pdf?id=B14TlG-RW)

**Notes from the above technical paper- Question Answering Network using CNNs and self-attention**

Current QA models use RNN with attention. They are good enough but slow during training and inference. This is because of the sequential nature of RNNs (Recurrent Neural Networks).

What is better? QANet- No need to use RNNs. The encoder consists of convolution and self-attention. Convolution models local interactions and self-attention models global interactions.

**Previously…**

Machine reading comprehension and automated question answering was done using:

1. A recurrent model that processes sequential inputs,
2. Attention component that deals with long-term interactions.

The combination of 1 and 2 results in BiDAF (Bidirectional attention flow).

**The disadvantages:**

1. Slow training- hence can’t be used with large datasets.
2. Rapid iteration is not possible due to slow nature.

**How to improve the speed of machine learning comprehension?**

The recurrent nature of the model must be removed.

The building block of encoders would be convolutions and self-attentions. This will separately encode the query and the context. Hence, the interaction between the context and question will be understood with the help of standard attentions. The resultant representation is encoded with the encoder. This is finally decoded to give the most probable position of the start and end index of the answer.

**Why convolution and self-attention?**

Convolution captures local structure of the text and self-attention learns global interaction between the pair of words.

3x to 13x faster while training the data.

4x to 9x faster in inference.

F1 score is the metric used to see how the model performs.

**F1 score = 2 \* (Precision \* Recall)/(Precision + Recall)**

Tells about the balance between precision and recall.

**Precision**

* Also known as Positive Predictive Value (PPV).
* It is the number of positive predictions divided by the total number of positive class values predicted.
* **Precision = True positive/ (True Positive + False Positive)**

**Recall**

* Also known as sensitivity or True Positive Rate.
* It is the number of positive predictions divided by number of positive class values in the test data.
* **Recall = True Positive/ (True Positive + False Negative)**

**Model**

QANet is a feedforward model that consists of convolutions and self-attentions.

**Model overview**

5 major layers-

* An embedding layer
* An embedding encoder layer
* A context-query attention layer
* A model encoder layer
* An output layer

Note: For the embedding and modelling encoders, convolutional and self-attention mechanism is used. RNNs are not used at all.

**Why use convolutions?**

Common regularization methods such as stochastic depth.

**Note:** Enough gains can be achieved if data augmentation is done on the dataset to enhance it.

**Data augmentation to enhance training data**

In this process, context and passage pairs are translated to another language and then translated back into English. This way, examples are paraphrased thereby increasing the number of training instances and diversifying the training.

---------------------------------------

**My notes:**

**Deep learning**

It takes huge volumes of structured/unstructured data as text. It applies complex algorithms to train the neural networks. Complex operations are performed to extract hidden patterns and features from the data.

**What is an encoder?**

Encoder is a network (CNN, RNN) that takes input and generates an output.

The output can be a feature map, a vector or a tensor. This output holds information (features) that represents the input.

The decoder is a network that has the same network structure as that of the encoder but in the opposite orientation.

It takes the output (the feature vector) from the encoder and gives the closest match with respect to the actual input or the intended output.

Encoder is trained with the help of the decoder. No labels are present; hence it is unsupervised learning.

Loss function is based on computing the difference between the actual input and the reconstructed input.

The optimizer tries to train both encoder and decoder to reduce the reconstruction loss.

After the encoder has been trained, the encoder gives feature vector as output. This is used by decoder to construct input with relevant and important features.

This reconstructed input would be closer to or recognizable as the actual input.

**Self-attention**

BERT is a transformer-based architecture.

Transformer-based architectures are used to model language-understanding tasks, avoiding the use of recurrence in neural networks, and depending on self-attention mechanisms to draw global dependence on input and output.

**Attention**

Seq2Seq (sequence to sequence) models are deep learning models that have achieved success in tasks like machine translation, text summarization, and image captioning.

**What is a seq2seq model?**

**Reference-** <https://towardsdatascience.com/day-1-2-attention-seq2seq-models-65df3f49e263>

<https://medium.com/@bgg/seq2seq-pay-attention-to-self-attention-part-1-d332e85e9aad>

A model that takes a sequence of words (words, letter, time series) and outputs another sequence of items. In the task of machine translation, the input is in language A and the output is in the desired language (translated from language A).

Used for tasks like machine translation, chatbot question answering, caption generation, and in tasks that require producing a sequence from another sequence.

The seq2seq model consists of an **encoder** and a **decoder**.

**What does the encoder do?**

It captures the context of the input sequence in the form of a hidden state vector. The size of this hidden state vector is a power of 2 (256, 512, 1024). This is sent to the decoder. The decoder produces the output sequence.

Encoder processes the input sequence and compresses the information into a ‘**context vector**’ that has a fixed length.

**Disadvantage of this fixed length context vector**

* Long sentences can’t be placed in memory. This would lead to unsatisfactory results.

This representation (context vector of fixed length) will contain a relevant summary of the meaning of the entire source sentence. The decoder is given the ‘**context** **vector**’ as input that generates the output.

**Why is size of hidden state vector in powers of 2?**

DNN are trained on GPUs to improve the speed of training time. GPU can take advantage of optimizations associated with efficiencies in working with powers of 2.

Since the task is sequence based, both the encoder and the decoder use some form of RNN, or LSTM or GRU.

**RNN (Recurrent Neural Network)**

**A close up of text on a black background

Description automatically generated**

RNN takes 2 inputs- **the current example** that they see and a **representation of the previous input**.

Hence the output at time ‘t’ depends on the current as well as previous input at time ‘t-1’. The sequential information is preserved in the hidden state of the network that is used with the next instance. Hence, they perform better when they work with sequence related tasks.

The encoder consists of RNNs that take a sequence as input and generate a final embedding at the end of the sequence.

This final embedding is sent to the decoder which uses this embedding to predict a sequence. After every successful prediction, the previous hidden state is used to predict the next instance of the sequence.

**HS- Hidden State**

A close up of a sign

Description automatically generated

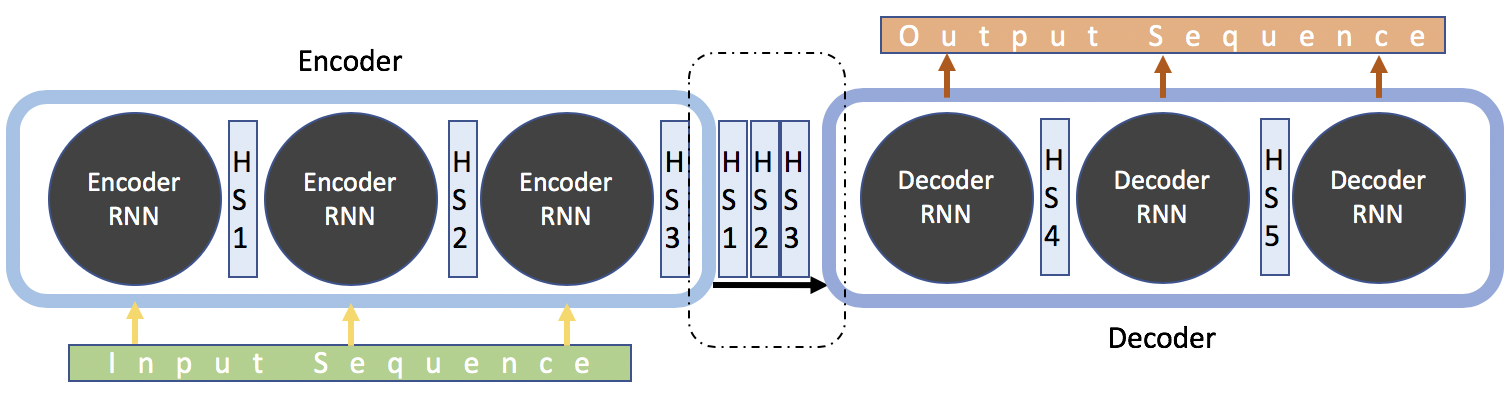
**Drawback**

Difficult for the model to deal with long sentences. When there are long sentences, the initial context would mostly get lost by the end of the sequence.

**Reason?**

The final output of the decoder depends on context that is defined by the hidden state.

**What is the solution to overcome this? ATTENTION**

****

It was developed on 2015 to overcome the issue with context vector of fixed length. It helps memorize long source sentences in neural machine translation.

**IDEA**

Instead of building a single context vector from the encoder’s last hidden state, attention creates context vectors for evert input word. This context vector is a sum of the hidden states in the input sequence, that are weighted by an attention score. Attention score (also known as alignment score) is the most important idea in attention model, that helps in interpreting importance of a word.

A screenshot of a cell phone

Description automatically generated

**\*\*- READ AGAIN ABOVE- didn’t understand**

**Global attention**

Global attention model considers all hidden states of the encoder when deriving the context vector. In such a model, a variable-length alignment vector ‘alpha’ (whose size is the same as number of time steps on the source side) is derived by comparing current target hidden state with every source hidden state.

Once the e\_{ij} value is obtained, the attention score ‘alpha’ is parametrized by a feed-forward network. The alignment model ‘a’ is calculated using ‘softmax’ equation and context vector is obtained. This is trained with other parts of the model. The matrix of attention scores depicts the correlation between the source and target words.

**Disadvantage of using Attention**

* Context vector is calculated using the hidden state between the source and target sequence. Due to this, the attention information inside the source sentence and target sentence is ignored.
* RNN is difficult to parallelize, hence calculations consume time.

This mechanism allows the model to concentrate on different parts of the input sequence at every stage of the output sequence. This way, the context is preserved from the beginning to the end.

HS1, HS2, and HS3 have been sent as the input to the decoder. These are the input instances of the encoder. These are sent to the decoder as input.

This attention-based model also comprises of a **context vector**. This context vector is generated for every ‘time instance’ in the output sequence. At every step, the context vector is a weighted sum of the input hidden states.

A picture containing clock

Description automatically generated

**How can context vector be used in the process of prediction?**

The context vector (that is generated) is concatenated with the hidden state vector. This is together known as ‘**attention hidden vector**’. This resultant can be used to predict the output at a specific ‘time instance’. This attention hidden vector is generated for every ‘time instance’ in the output sequence. It replaced the hidden state vector.

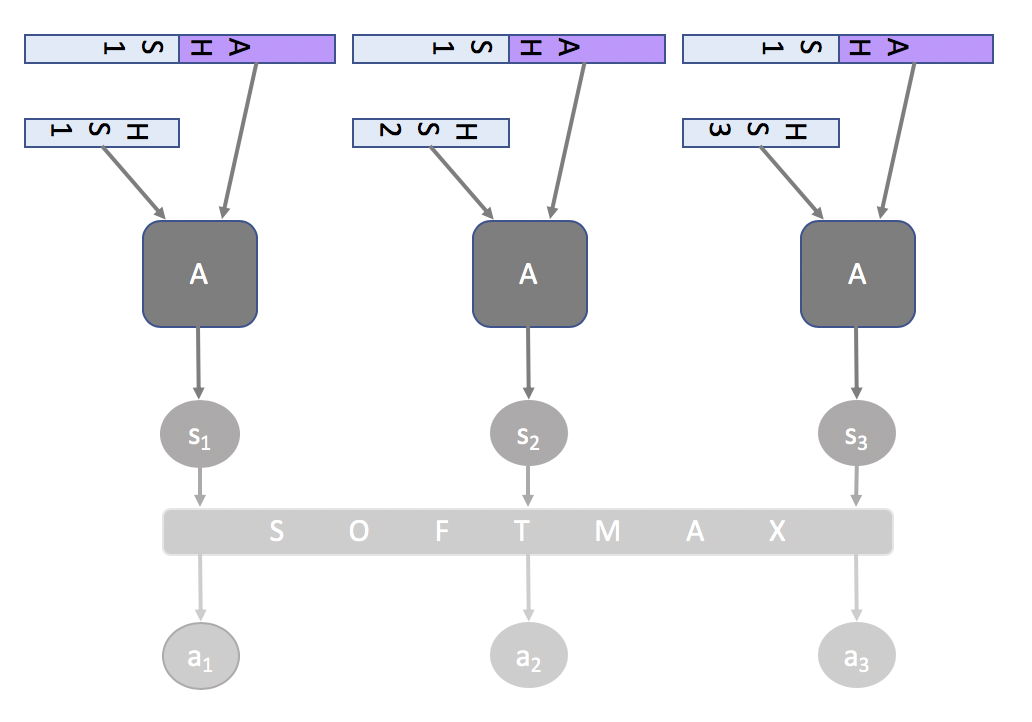
A close up of a sign

Description automatically generated

**How are the weights a1, a2, and a3 decided?**

This is the output from a different neural network, known as the ‘**alignment model**’. Initially, it is trained along with seq2seq model.

The **alignment model** is used to understand and give a score on how well an input (represented using its hidden state) matches with its previous output (represented by attention hidden state). This matching is done for every input with its previous output. The ‘softmax’ is applied over these scores which results in a number that is the attention score for the respective input.



In the training phase, the model learns how to align various instances from output sequence to input sequence.

A screenshot of a computer

Description automatically generated

This is a seq2seq attention-based model.

------------------------------------------------------

**RNN/LSTM**

Feedforward neural networks don’t use memory to remember/process data.

On the other hand, RNN uses memory to remember past data while processing sequential data. Having too much memory harms the model during training.

In order to overcome this drawback of RNN, LSTM was introduced.

LSTM has methods like forget/input/output gate, cell memory, hidden state, etc,. LSTM and GRU are two specific recurrent neural networks that are used frequently.

-----------------------------------------

**What is the solution to drawbacks raised by Attention model?**

**Self-attention**

Reference- <https://medium.com/@bgg/seq2seq-pay-attention-to-self-attention-part-2-cf81bf32c73d>

It is the concept of transformer model, introduced by Google. It outperforms attention model in many tasks.

**Concepts in transformer:**

* Self-attention
* Multi-head

The Transformer is used for **self-attention** and **parallelization**. Transformers don’t use the architecture of RNN or CNN.

**What is a transformer?**

It consists of two parts- encoder and a decoder.

The encoder is not a single network, but a stack of encoders placed one above the other.

The decoder is like the encoder, that will have same number of decoders as that of encoders.

**Recalling attention model concepts:**

From the input sequence x1, x2, … xm, the encoder generates hidden input states h1, h2, … hm. The context vector is a sum of the hidden states (of the input sequence), weighted with attention score ‘alpha’. With the help of context vector and hidden state, the output sequence can be calculated as y1,y2,..yn.

**Translating this to ‘Query, Key, Value’**

Input words in the source sentence is nothing but an array of data. Hence every element in the array has an address (can be translated to key) and element at the address (can be translated to value). The output word in the target sentence can be understood as a query.

**Leetcode STARS**

First 15 minutes was just talking about the projects and resume. I dived into really deep about some projectes that I was working on. (Ex: I had this problem so I used this way to solve, why I used this way bla bla bla...)

Then CoderPad question:  
Create a ImageManager that can be used for downloading an image from an URL. If the image has been downloaded previously, retrieve from cache.

Honestly this was a surprise to me because I thought would be some algorithms and DS questions.

Explained what my implementations would look like, talked about concurrency and start coding. I also explained edge cases, cache memory issues and best practices for caching images. Supposed to be one hour interview but I finished the coding question within 20 minutes and no more after that.

Recruiter sent me a take home assignment the next day.

**Take home assignment**(2 days):  
Second round was just a take home project, retrieve data from some URLs and display on the screen. (Not allowed to use any 3rd party frameworks)

Very classic take home assignment. Just need to make sure your code is clean, good architecture and testable. (Make sure to write tests and use **git**)

**Python stars**

<https://python.quantecon.org/index_toc.html>

utsav Aggarwal

joy during job search linkedin

**// to find the largest item in an iterable**

**max(iterable, \*iterables, key, default)**

**// to find the largest item between two or more objects**

**max(arg1, arg2, \*args, key)**

**w3schools sql**