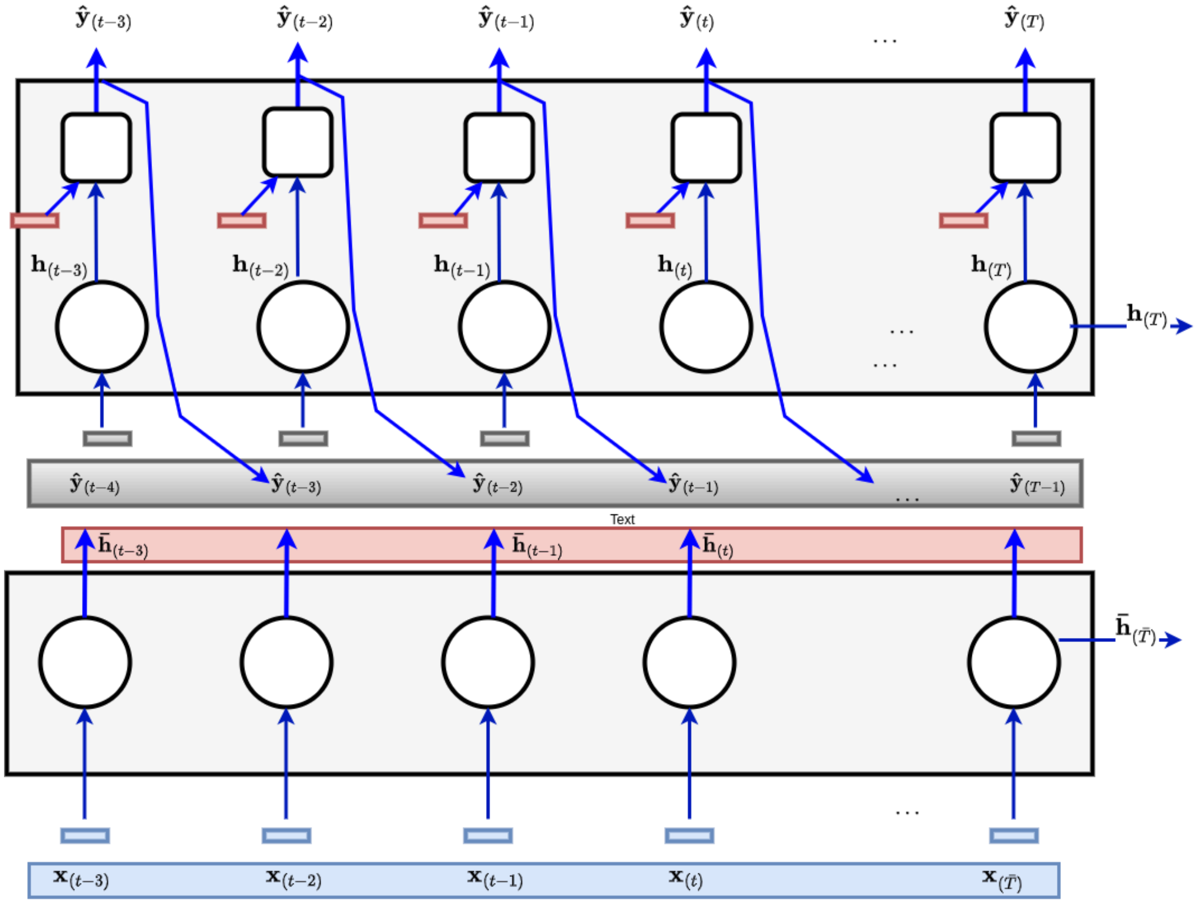
Teacher Forcing

Here is a diagram of the Encoder/Decoder architecture.

RNN Encoder/Decoder with Cross Attention and Self Attention (Encoder/Decoder)



The grey box represents the *entire* output sequence

$$\backslash\hat{\mathbf{y}}_{(1:T)}$$

From this diagram: it appears that

- the Encoder/Decoder can produce output $\backslash\hat{\mathbf{y}}_{\backslash\mathrm{tp}}$
- while attending to outputs *that have not yet been generated* at the start of step
$$\backslash\hat{\mathbf{y}}_{(\texttt{:T})}$$
- "looking into the future"

That is, it is computing

$$\backslash\mathrm{prc}\backslash\hat{\mathbf{y}}_{\backslash\mathrm{tp}}\backslash\hat{\mathbf{y}}_{(1:T)}$$

What is going on ?

Teacher forcing at training time

The *Auto-Regressive* Decoder behavior

- constructs output sequence $\backslash\hat{\mathbf{y}}$ sequentially
  - output $\backslash\hat{\mathbf{y}}_{\backslash \mathrm{tp}}$ of step
  - becomes part of the input $\backslash\hat{\mathbf{y}}_{(1:)}$ of step +1

So how can the entire output sequence $\backslash\hat{\mathbf{y}}_{(1:T)}$ be available before final step T ?

We need to distinguish between behavior

- during *training*
- versus during *inference*

During inference, clearly we can only compute

$$\backslash\mathbf{prc}\backslash\hat{\mathbf{y}}_{\backslash\mathrm{tp}}\ \backslash\hat{\mathbf{y}}_{(1:-1)}$$

because we haven't generated the future outputs yet.

But at training time, example is

$$\langle \backslash\mathbf{y}_{(1:-1)}, \backslash\mathbf{y}_{\backslash\mathrm{tp}} \rangle$$

We predict *only the immediate next* target $\backslash\mathbf{y}_{\backslash\mathrm{tp}}$

- *not* the full suffix $\backslash\hat{\mathbf{y}}_{(:\mathtt{T})}$

The prediction is conditioned on the *true* target prefix $\backslash\mathbf{y}_{(1:-1)}$

- *not* the prefix generated during training $\backslash\hat{\mathbf{y}}_{(1:-1)}$

The Auto-Regressive behavior is eliminated during training !

Training in this manner has a big advantage.

In a a perfect world, when predicting $\backslash\hat{\mathbf{y}}_{\backslash\mathrm{tp}}$

- the Auto-Regressive behavior during training would result in
- the prefix $\backslash\hat{\mathbf{y}}_{(1:-1)}$ of the generated output
- matching the true target

$$\backslash\hat{\mathbf{y}}_{(1:-1)} = \backslash\mathbf{y}_{(1:-1)}$$

But

- if any element of the *generated* prefix is wrong
$$\backslash\hat{\mathbf{y}}_{(')} \neq \backslash\mathbf{y}_{(')} \text{ for } ' <$$
- it is likely that all *subsequent predicted outputs* $\backslash\hat{\mathbf{y}}_{('+1:T)}$ will be wrong
- because each subsequent output **is conditioned on incorrect** $\backslash\hat{\mathbf{y}}_{(')}$

That is

- a single mis-predicted element of the sequence
- causes a catastrophic chain of errors

Training a model under such conditions would be difficult

- the incorrect sequences don't even come from the true distribution of inputs !
- violating the Fundamental Theorem of Machine Learning

To avoid this, our training examples compute

$$\backslash\mathbf{prc}\backslash\hat{\mathbf{y}}_{\backslash\mathrm{tp}}\,\backslash\mathbf{y}_{(1:-1)}$$

rather than

$$\backslash\mathbf{prc}\backslash\hat{\mathbf{y}}_{\backslash\mathrm{tp}}\,\backslash\hat{\mathbf{y}}_{(1:-1)}$$

That is: we train on *target* prefixes rather than train-time generated prefixes.

This is called *Teacher Forcing*.

Teacher forcing can be implemented

- by making the entire target output sequence
$$\backslash\mathbf{y}_{(1:T)}$$
i.e., the grey box in the diagram

- available at training time via setting example to
$$\langle \backslash\mathbf{y}_{(1:T)}, \backslash\mathbf{y}_{\backslash\mathrm{tp}} \rangle$$

- and using *Causal masking*
- to make only prefix $\backslash\mathbf{y}_{(1:-1)}$ visible during the prediction of $\backslash\mathbf{y}_{\backslash\mathrm{tp}}$

In [2]:

```
print("Done")
```

Done