So far we have considered cases where:

- $y \in \mathbb{R}$ classification
- $|X| = |Y|$ Sequence Tagging
  Here we 'tag' $X_j$ with $\hat{Y}_j$, assuming these are aligned.

But many tasks do not conform to either of these assumptions.

- Image captioning
  \begin{center}
  \includegraphics[height=0.5in]{image.png}
  \end{center}
  a cat

- Translation
  \begin{center}
  C'est la vie \hfill That's life
  \end{center}

In general: Any time we have sequences as both input and output, and they may differ in shape.

$$X = [x_1, \ldots, x_n] \quad n \neq m \text{ in general}$$

$$y = [y_1, \ldots, y_m]$$
**Seq2Seq** models provide an abstraction for such cases.

Two components

- An **encoder** \( \text{enc} \)
- A **decoder** \( \text{dec} \)

Then:

\[ Z_i = \text{enc}(x, \theta_{\text{enc}}) \]

\[ \hat{y} = \text{dec}(z, \theta_{\text{dec}}) \]

This is pretty abstract. Consider a concrete example: Email response generation.

At **train time**, we have a choice: Condition the decoder at time \( t \) on (1) prediction or (2) truth.

We call (2) **teacher forcing**.
Let's consider a simple (toy) example: Learning to add in a strange way.

"56 + 8" → "64"

A string! Also a string!

Exercise Let's design this together!

What modules do we need?

- Encoder
- Decoder
- Seq2Seq to package together

Let's flesh these out.

Encoder

Input: chars
Embed
RNN

Shapes

In
Out

< b x max_len x num_chars>
< b x max_len x num_chars> < b x max_len x o1>
< b x max_len x o1>
< b x h >

Assuming we take just last hidden state
Decoder

Note: We do one step at a time here.

Input: \( Z \) and \( \hat{y} \) (char from last step)

Embed \( <b \times h>, <b \times \text{num\_chars}> \) \( <b \times d> \)

RNN \( <b \times h>, <b \times d> \) \( <b \times h> \)

Prior hidden states \( \hat{y} \) embeddings

Out \( <b \times h> \) \( <b \times \text{num\_chars}> \)

Now let's write pseudo code for \textit{Seq2Seq} (forward)

\textbf{Seq2Seq}

\[ Z \leftarrow \text{enc}(x) \]
\[ \hat{y}_t \leftarrow \mathbf{0} \]

decoded \( \leftarrow [\] \)

for \( t \) in \( \text{MAX\_OUT\_LEN} \)

\[ \hat{y}_t, Z \leftarrow \text{dec}(Z, \hat{y}_t) \]

decoded. append (\( \hat{y}_t \))

return decoded

To Colab!