As opposed to RNN models that take the input sequence token-by-token, the transformer takes the input sequence as a whole (parallel), then generates the output one-by-one as before.

Another difference between transformer seq2seq models and RNN seq2seq is that they use feed-forward neural networks instead of RNNs.

The third major novelty in transformers is a concept called self-attention.

It has a stack of encoders and decoders.
Encoder

- feed-forward
- self-attention

Decoder

- feed-forward
- self-attention
- enc-dec attn

focus on the relevant parts of the inputs

focus on the previous decoder outputs

this attn is in the encoder now!
Self-Attention

- Embed
- Score
  - Scale by $\sqrt{d_k}$
  - Softmax
  - Multiply & Sum
  - $\sim 0.73$
  - $14$
  - $28$
  - $\uparrow$
  - Word 1
- $\sim 0.27$
  - $13$
  - $26$
  - $\uparrow$
  - Word 2
  - Word 3
- Assume we are working on Word 2
- Context vector
Decrypt the transformer paper

- Problem in this figure: model is mainly focusing on other similar words. 
  Let's modify it!
Query Layer

Query 1
Query 2
Query 3

learned

embed

word1

word2

word3

mult by a Query matrix
Using the keys and queries, we calculate like this:

<table>
<thead>
<tr>
<th>Q</th>
<th>K</th>
<th>score(Q, K)</th>
<th>$\sqrt{dk}$</th>
<th>softmax</th>
<th>softmax * K</th>
</tr>
</thead>
<tbody>
<tr>
<td>word1</td>
<td>embed1</td>
<td>key1</td>
<td>28</td>
<td>14</td>
<td>0.2</td>
</tr>
<tr>
<td>word2</td>
<td>embed2</td>
<td>key2</td>
<td>key2</td>
<td>key2</td>
<td>key2</td>
</tr>
<tr>
<td>word3</td>
<td>embed3</td>
<td>key3</td>
<td>26</td>
<td>13</td>
<td>0.9</td>
</tr>
</tbody>
</table>

$vec1 + vec2 + vec3$  

self-attend context vector → passed to FFNN
- The authors also add a third thing: values. So:

<table>
<thead>
<tr>
<th>word1</th>
<th>embed1</th>
<th>Q</th>
<th>K</th>
<th>V</th>
<th>score(Q, K)</th>
<th>$\sqrt{d_k}$</th>
<th>softmax</th>
<th>softmax* V</th>
</tr>
</thead>
<tbody>
<tr>
<td>word2</td>
<td>embed2</td>
<td>Query</td>
<td>key2</td>
<td>Val7</td>
<td>28</td>
<td>14</td>
<td>0.2</td>
<td>vec1</td>
</tr>
<tr>
<td>word3</td>
<td>embed3</td>
<td>key3</td>
<td>Val3</td>
<td>26</td>
<td>13</td>
<td>0.9</td>
<td>vec2</td>
<td>vec3</td>
</tr>
</tbody>
</table>

- **Q** matrix: learned during training process
- **K**: by multiplying embeddings by $K$ matrix
- **V**: by multiplying embeddings by $V$ matrix