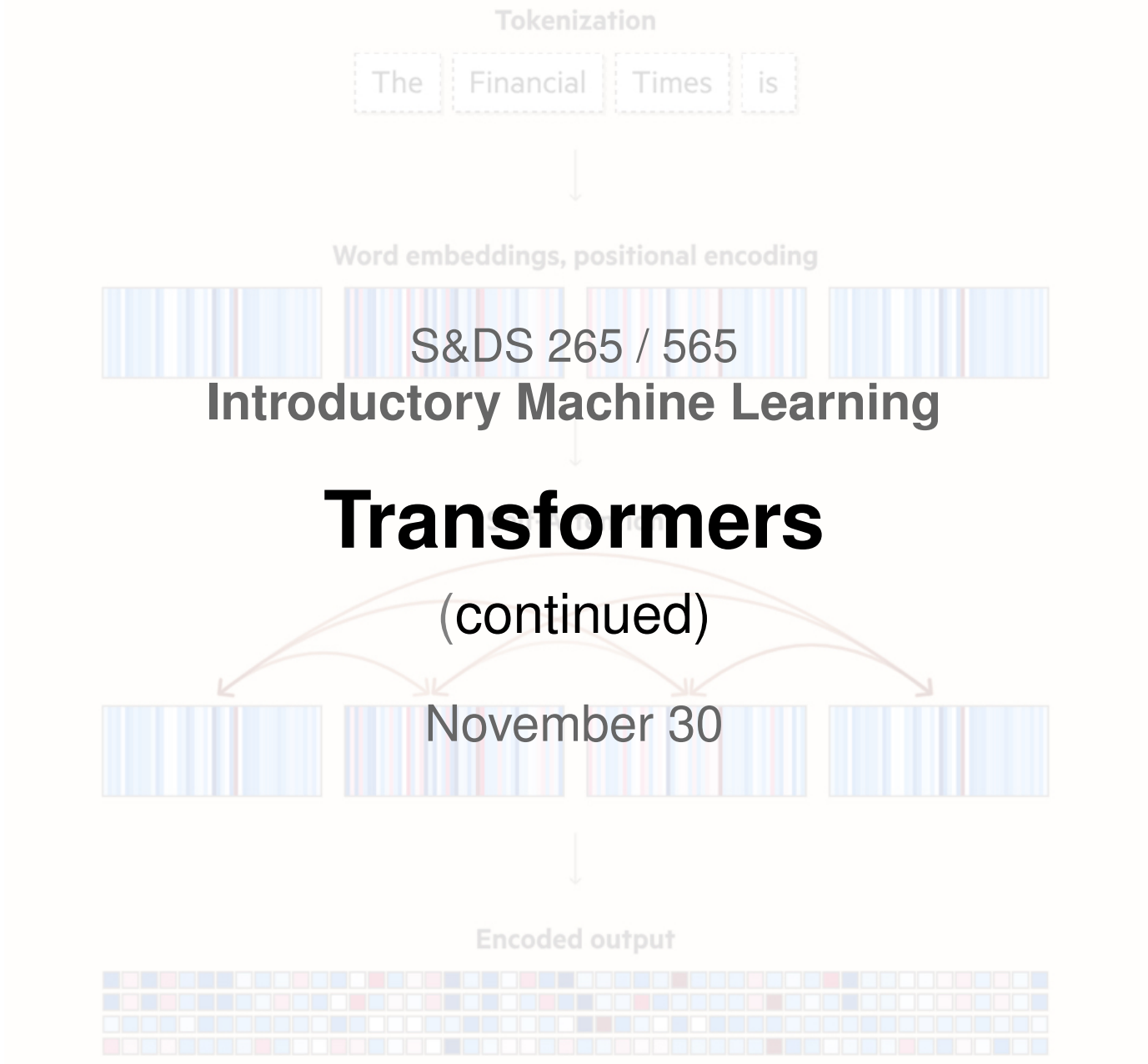


S&DS 265 / 565

Introductory Machine Learning

Transformers

(continued)

November 30

Yale

Some reminders

- Quiz 6 on December 7, next week
- Panel discussion! Tuesday, December 5
- Assn 5 out; due Tuesday, December 5 (extension)
- Final exam: Friday Dec 15, 2pm in SSS 114
- Practice exams are posted
- Review sessions TBA

---

https://registrar.yale.edu/general-information/final-exams

Reminder

- To make future iML classes run more smoothly, we are experimenting with a “ticketing” system
- Request: *For remainder of the semester, please make any requests (late assignments, regrades, etc.) by emailing* `sds265@yale.edu`
- This will allow us to test the system and process your requests more efficiently—thanks!

For today

- Attention
- Transformers—high level

Next: High-level overview of transformers

- We'll treat transformers at a high level
- Key concept: Attention
- Main intuition illustrated with word embeddings

Artificial Intelligence

Generative AI exists because of the transformer

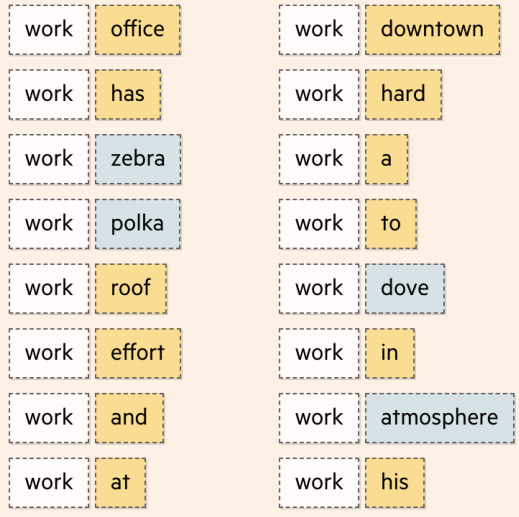
This is how it works

By **Visual Storytelling Team** and **Madhumita Murgia** in London SEPTEMBER 12 2023

Financial Times graphical intro: https://ig.ft.com/generative-ai/

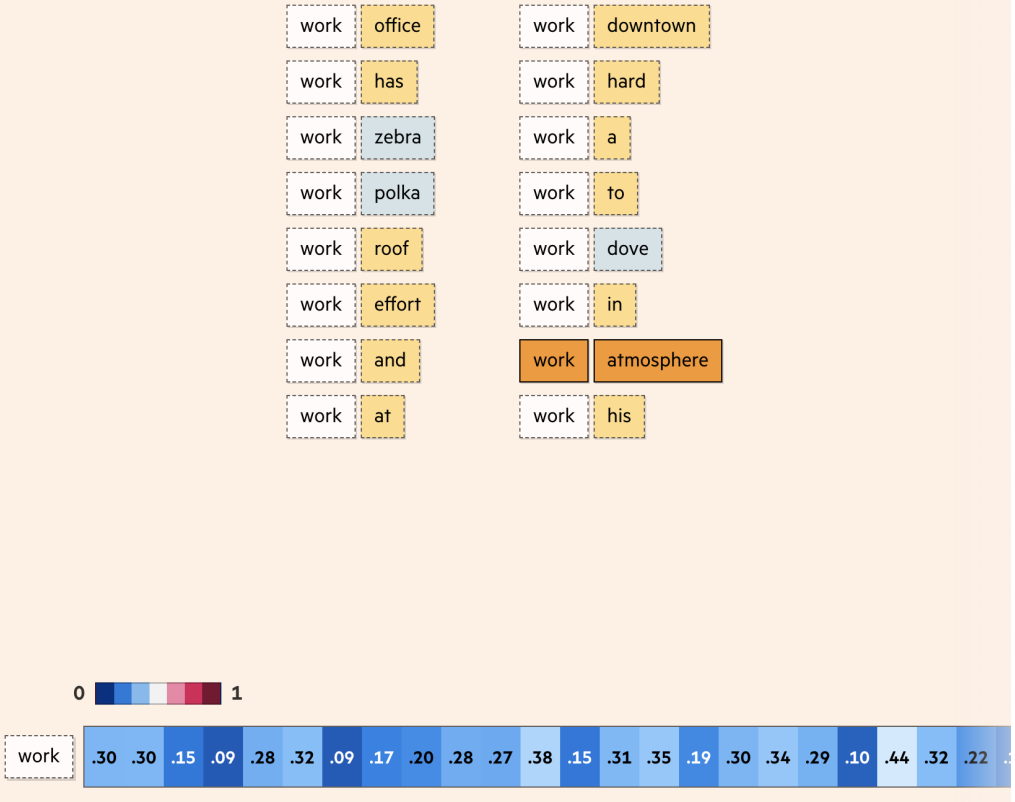
Starting point: Word embeddings

Eventually, we end up with a huge set of the words found **alongside** **work** in the training data, as well as those that **weren't** found near it.



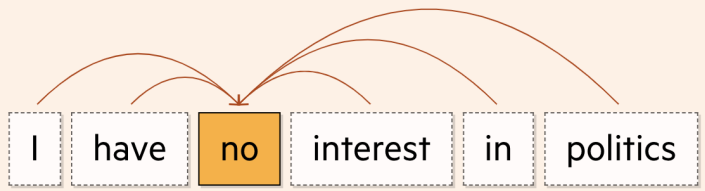
Starting point: Word embeddings

As the model **processes** this set of words, it produces a vector — or list of values — and adjusts it based on each word's proximity to **work** in the training data. This vector is known as a word embedding.



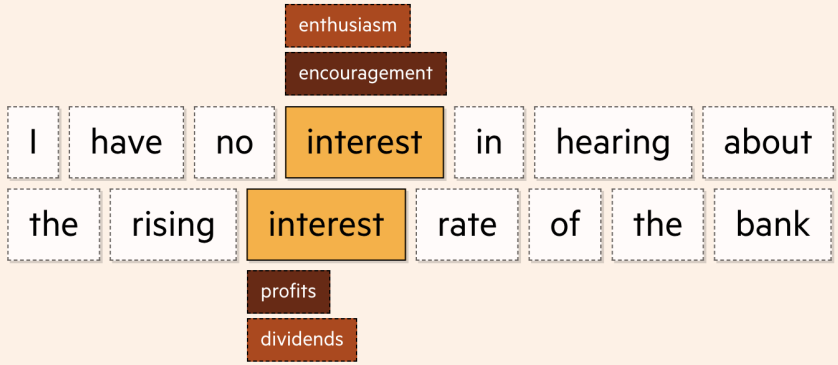
Introducing context: Attention

Self-attention looks at each **token** in a body of text and decides which others are most important to understanding its meaning.



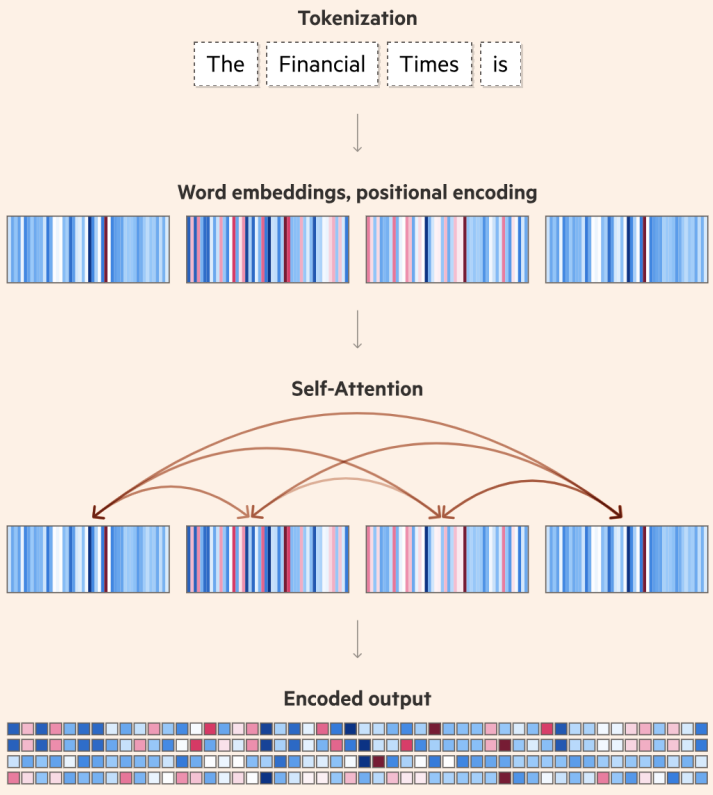
Introducing context: Attention

This functionality is crucial for advanced text generation. Without it, words that can be interchangeable in some contexts but not others can be used incorrectly.



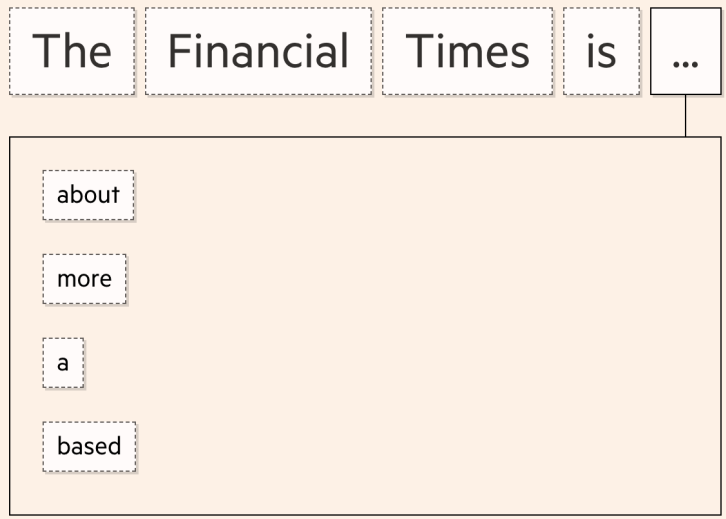
Building up: Encoding Layers

After tokenising and encoding a prompt, we're left with a block of data representing our input as the machine understands it, including meanings, positions and relationships between words.



Building up: Encoding Layers

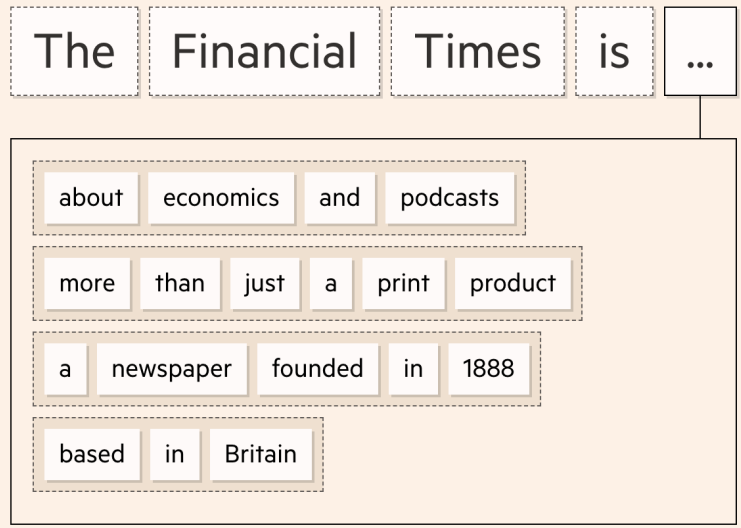
At its simplest, the model's aim is now to predict the next word in a sequence and do this repeatedly until the output is complete.



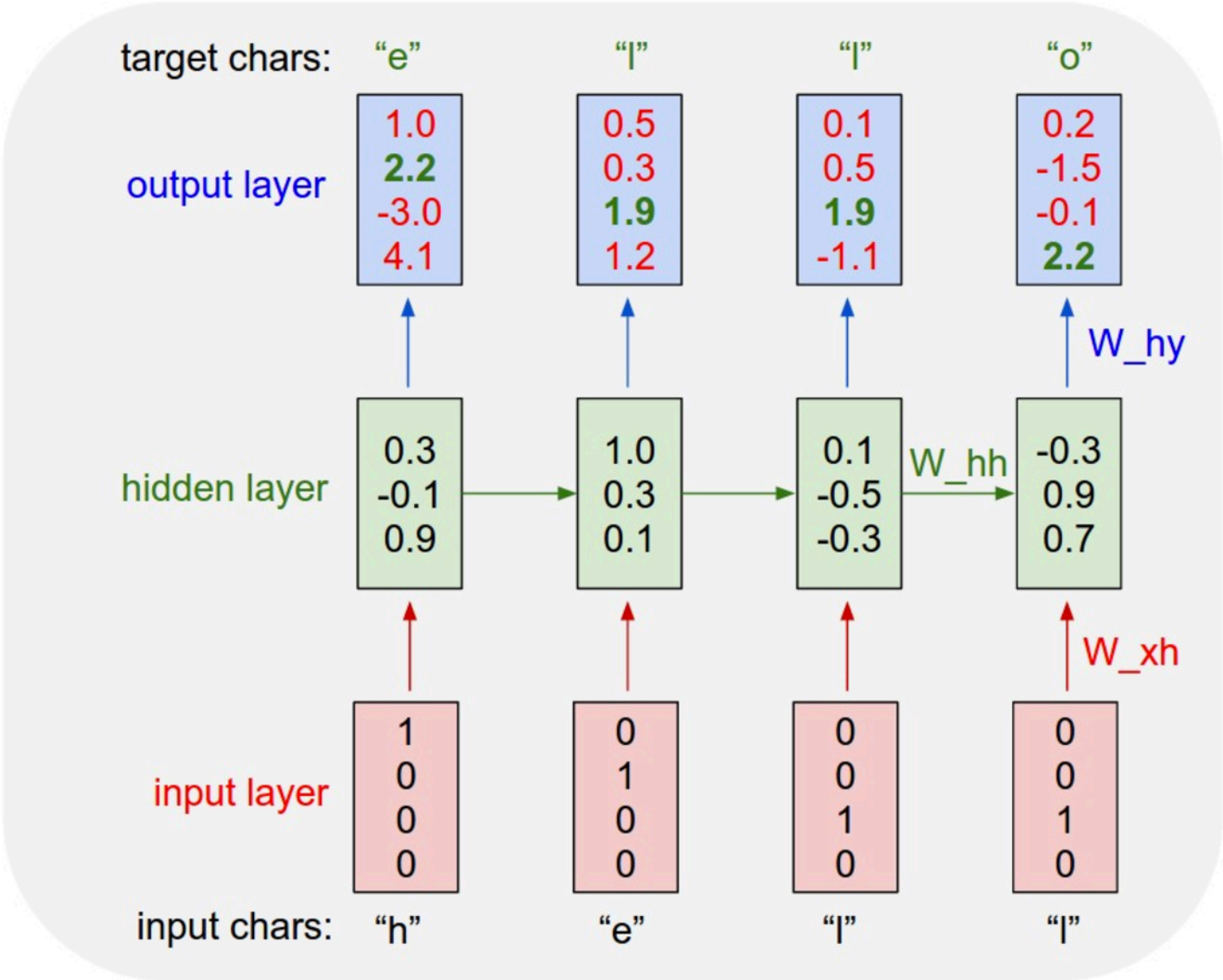
Making predictions: Beam search

Transformers use a number of approaches to address this problem and enhance the quality of their output. One example is called **beam search**.

Rather than focusing only on the next word in a sequence, it looks at the probability of a larger set of tokens as a whole.



RNNs http://karpathy.github.io/2015/05/21/rnn-effectiveness/



Attention/Alignment

We'll next illustrate attention in a "sequence-to-sequence" task—translating a sentence from French to English.

- Each source/targe word is associated with an encoder/decoder state—a vector of neurons.
- On each step of decoding, directly connect to the encoder, and focus on a particular part of the source sequence

Attention/Alignment

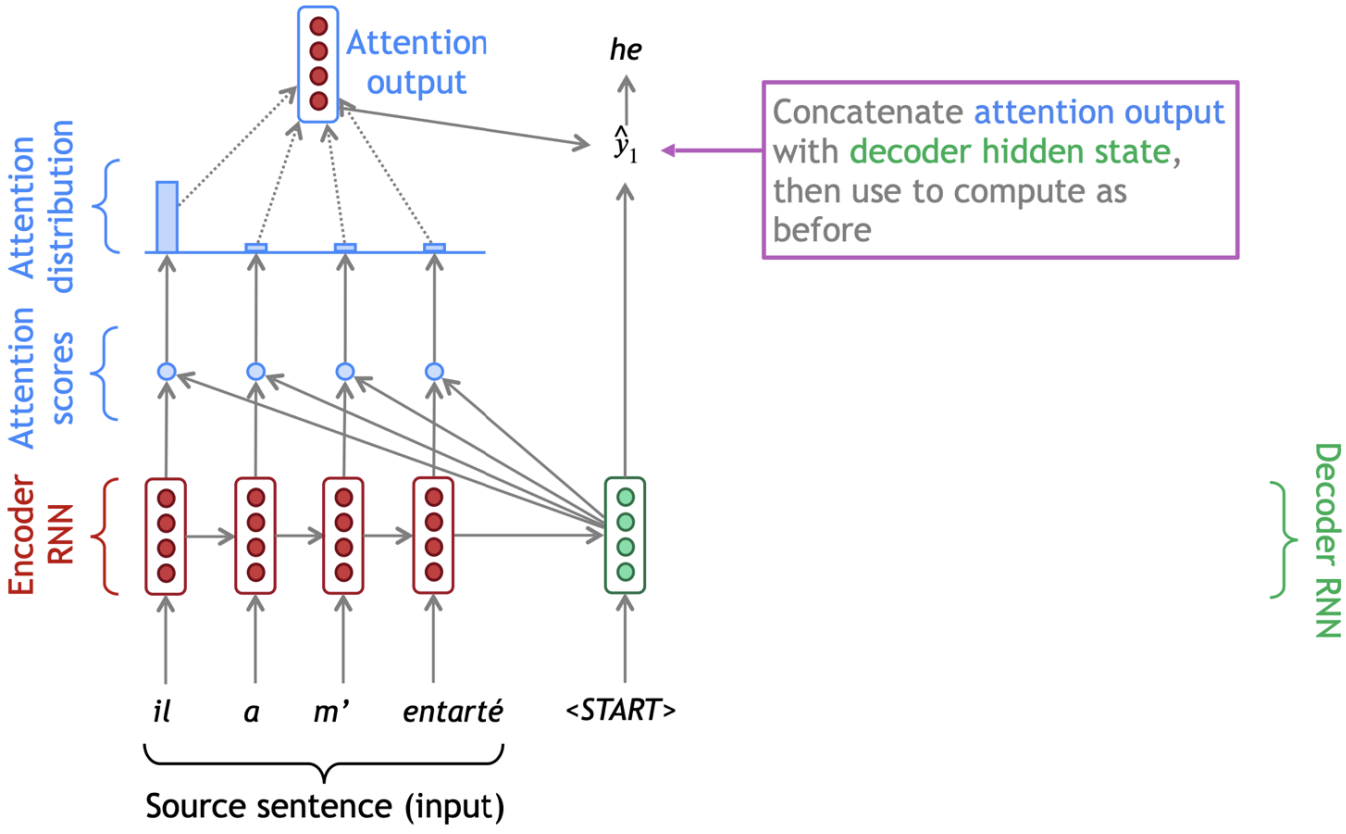


figure source: http://web.stanford.edu/class/cs224n/slides/cs224n-2020-lecture08-nmt.pdf

Attention/Alignment

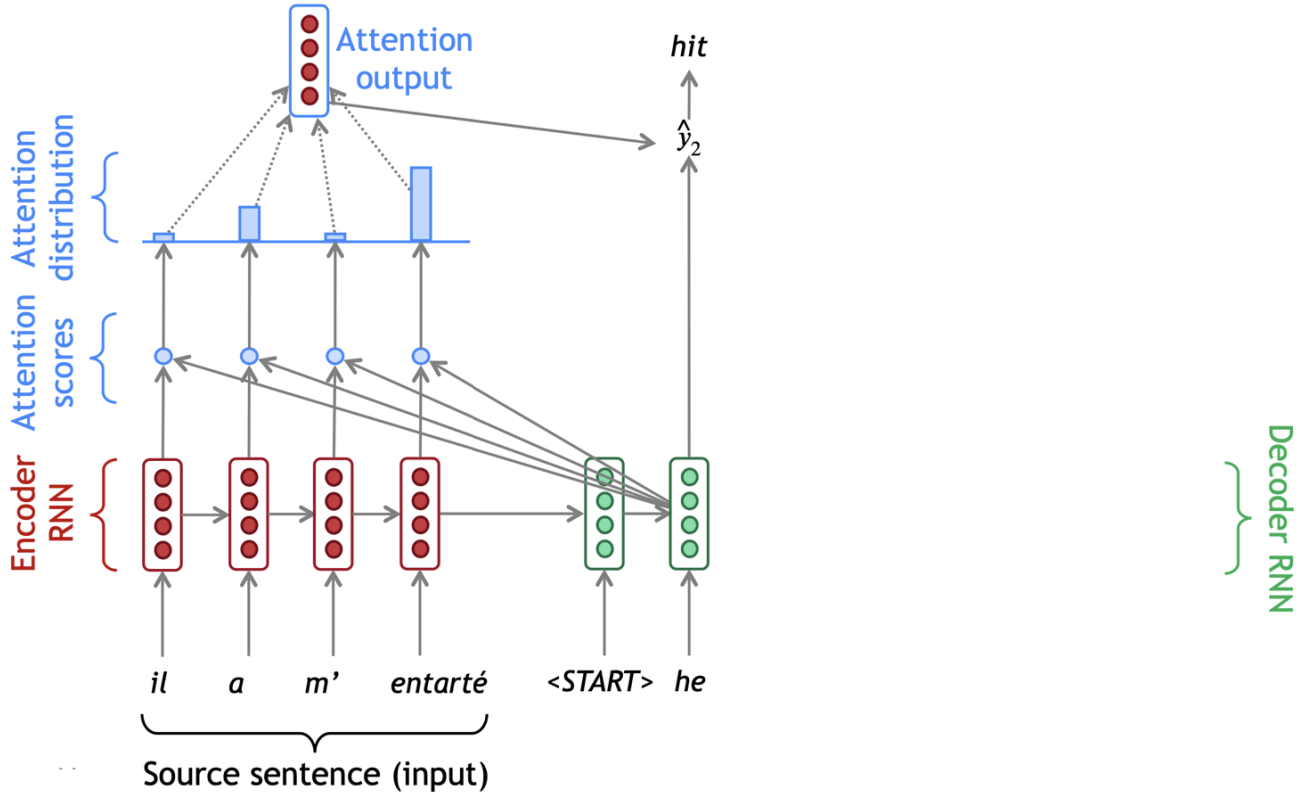


figure source: http://web.stanford.edu/class/cs224n/slides/cs224n-2020-lecture08-nmt.pdf

Attention/Alignment

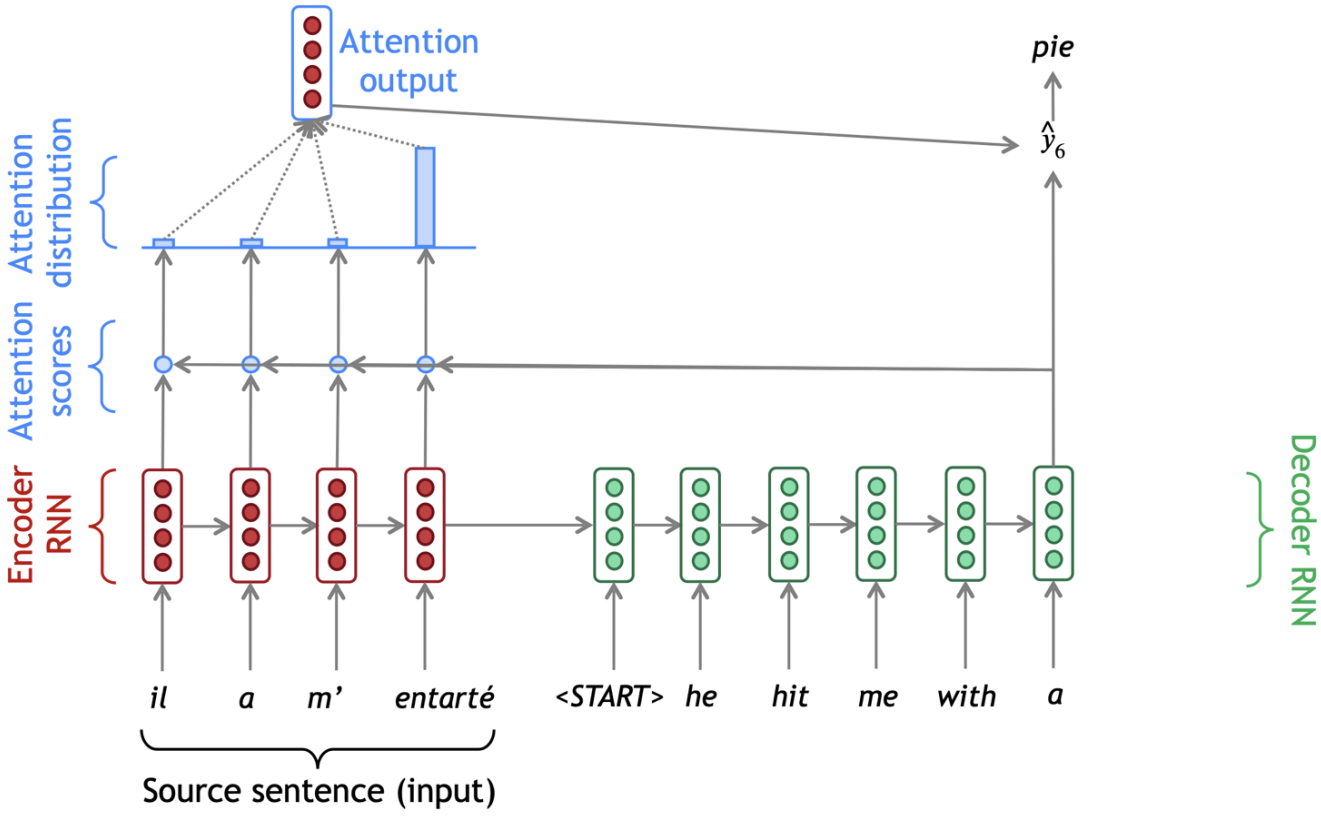
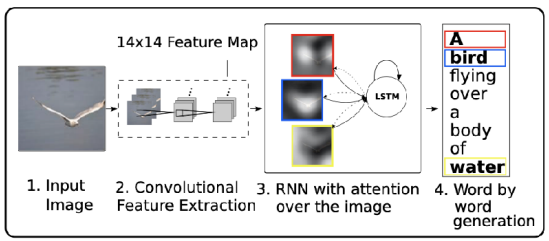
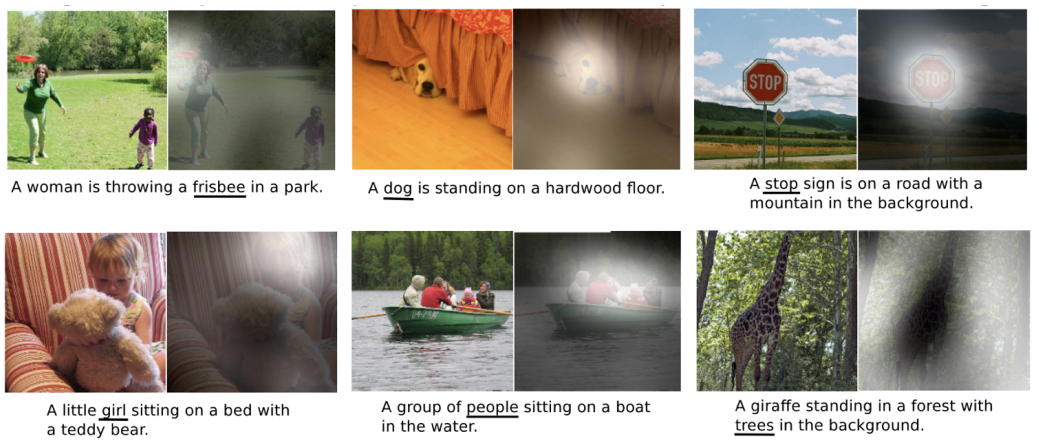


figure source: http://web.stanford.edu/class/cs224n/slides/cs224n-2020-lecture08-nmt.pdf

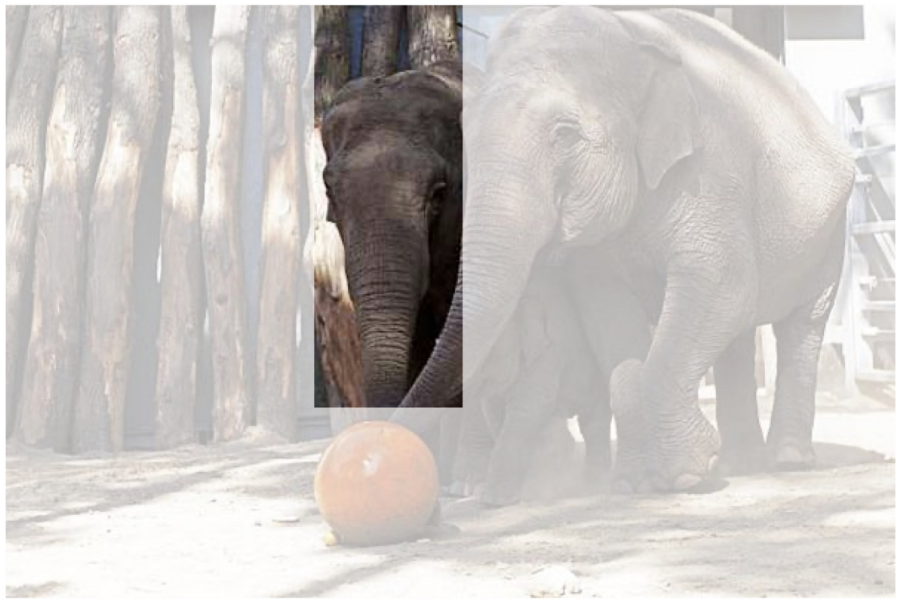
Attention/Alignment

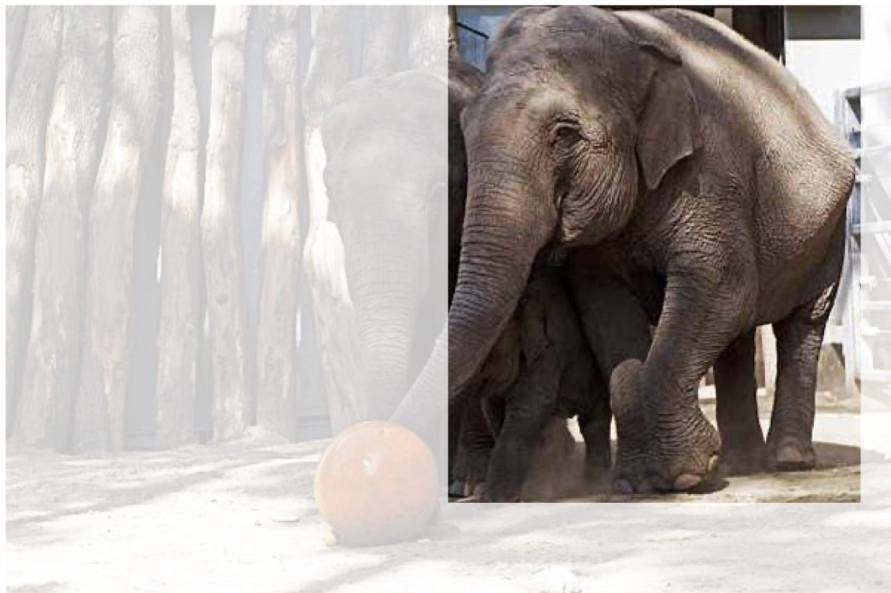
Attention can also be used for other generative models





* "Show, attend and tell: neural image caption generation with visual attention", Xu et al. 2016, https://arxiv.org/pdf/1502.03044.pdf





The two elephants played with an orange ball

The two elephants played with an orange ball

 → orange

 → ball

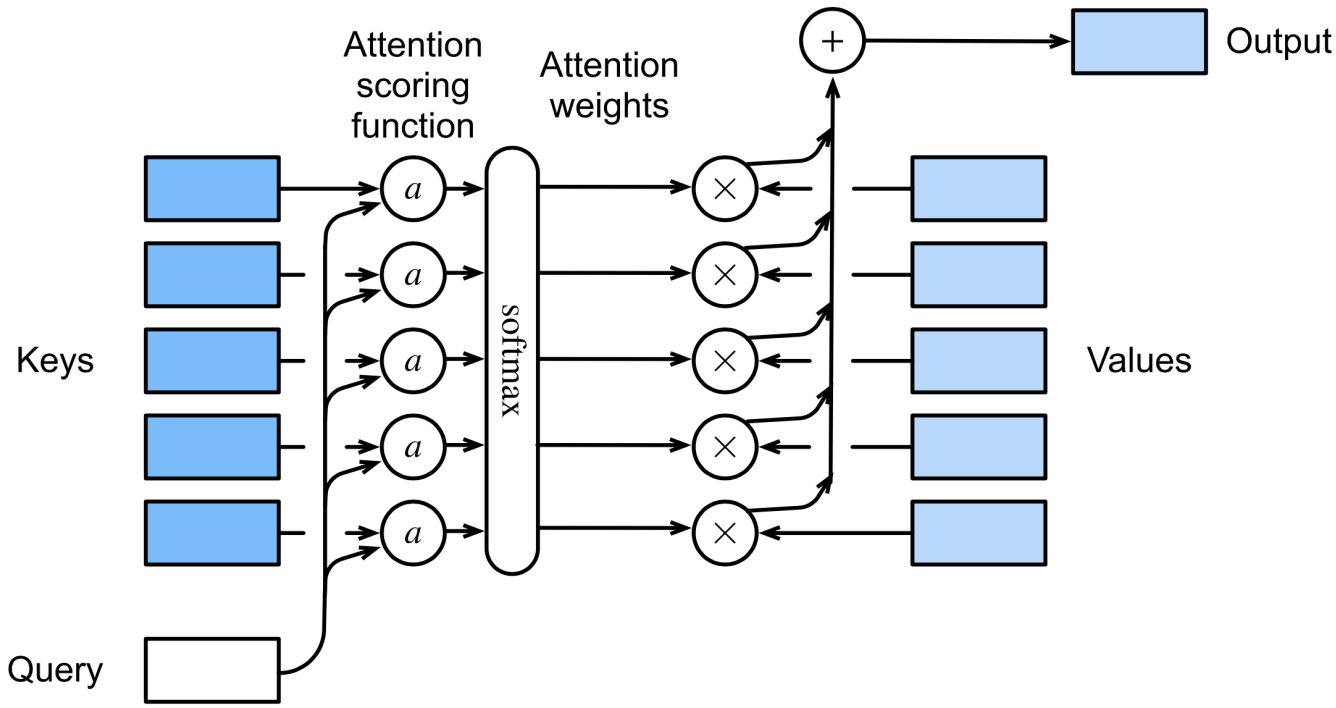
Attention in terms of query/key/values

Basic idea behind attention mechanisms:

- We have a set of m feature vectors or values $V \in \mathbb{R}^{m \times v}$
- The model dynamically chooses which to use
- based on how similar a query vector $q \in \mathbb{R}^{q}$ is to a set of m keys $K \in \mathbb{R}^{m \times k}$.
- If q is most similar to key i, then we use value (feature) v_i.

Attention in terms of query/key/values

Basic idea behind attention mechanisms:



Attention in terms of query/key/values

Basic idea behind attention mechanisms:

$$\text{Attn}(q, \{(k_1, v_1), \ldots (k_m, v_m)\}) = \sum_{i=1}^{m} w_i(q, k_{1:m}) v_i$$

where weights w_i are softmax of attention scores $a(q, k_i)$:

$$w_i(q, k_{1:m}) = \frac{\exp(a(q, k_i))}{\sum_{j=1}^{m} \exp(a(q, k_j))}$$

Dot product attention

Attention scores are computed as

$$a(q, k_i) = (W_Q q)^T (W_K k_i)$$

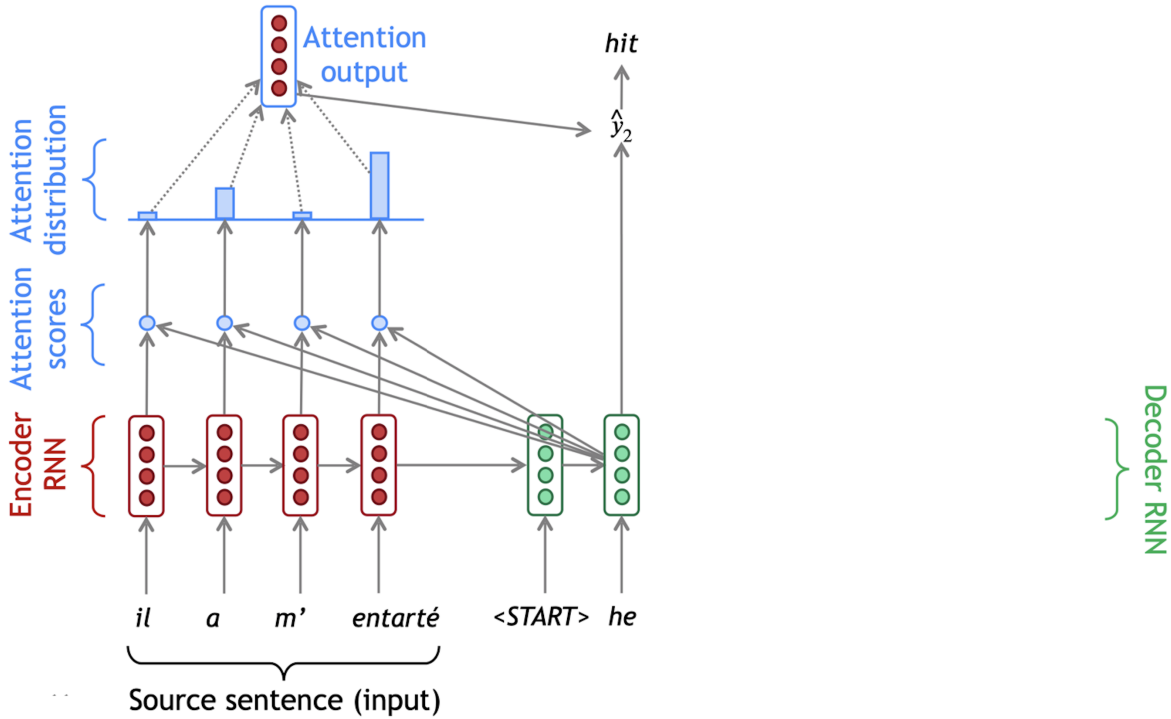
and then the attention weights are the softmax:

$$\text{Attn}(q, k_{1:m}, v_{1:m}) = \text{Softmax}\left(a(q, k)\right) v$$

The matrices W_Q and W_K allow comparison of query and keys from different "languages"—-French and English or images and words

Retour d'entarté

In the *entarté* example, the keys and values are the red state vectors, the queries are the green state vectors



Transformers

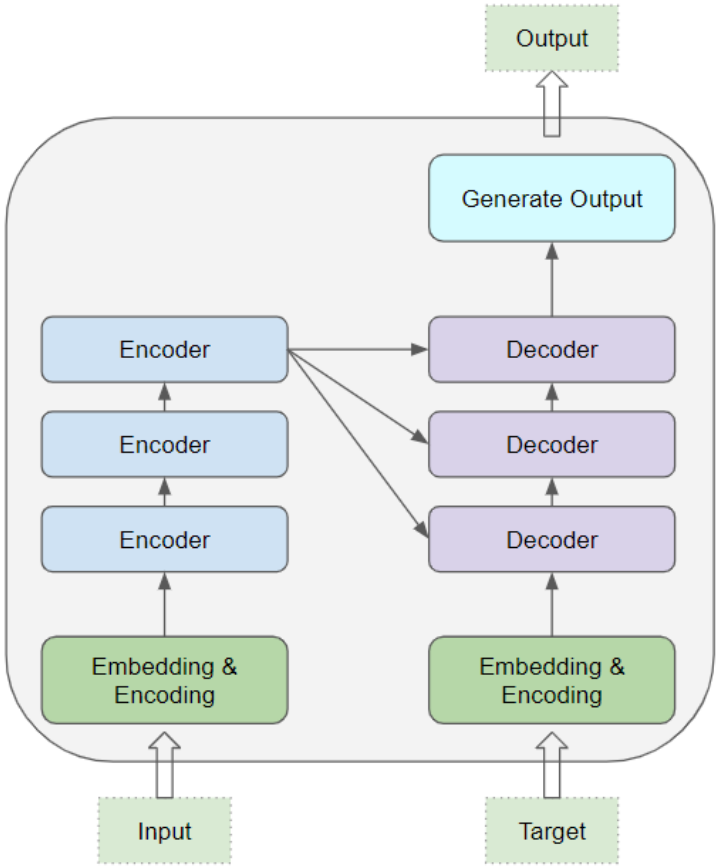
The current state-of-the-art is based on *transfomers*

- Attention is the key ingredient
- Rather than processing sequences word-by-word, transformers handle larger chunks of text at once
- The separation between words matters less

Combining attention vectors

- Different attention vectors are stacked up on top of each other
- Words “hungry” and “sweet” are predicted using all of them

Transformer architecture



Transformer architecture

- Each Encoder transforms its input to generate a vector E for each source word
- Each Decoder transforms its input to generate a vector D for each target word
- The Decoder model uses *cross-attention* to attend to the Encoder vectors

$$\text{Encoder self-attention: } Q \leftarrow E, K \leftarrow E, V \leftarrow E$$

$$\text{Decoder self-attention: } Q \leftarrow D, K \leftarrow D, V \leftarrow D$$

$$\text{Decoder cross-attention: } Q \leftarrow D, K \leftarrow E, V \leftarrow E$$

Demo

Attention

This demo illustrates some of the concepts behind attention in transformers, using word embeddings. The goal is simply to show how the basic calculations underlying attention work, and how weighting embeddings with attention can change the words that are similar. As implemented in Transformers, attention can be notoriously difficult to understand.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import networkx as nx
import gensim
```

Summary

- Attention is a type of alignment between related words
- Attention allows dynamic construction of word contexts, useful for predicting the next word
- Attention is computed using inner products, after mapping query and key to a common space
- Transformers process all words together, using layers of encoders and decoders, each with an attention component