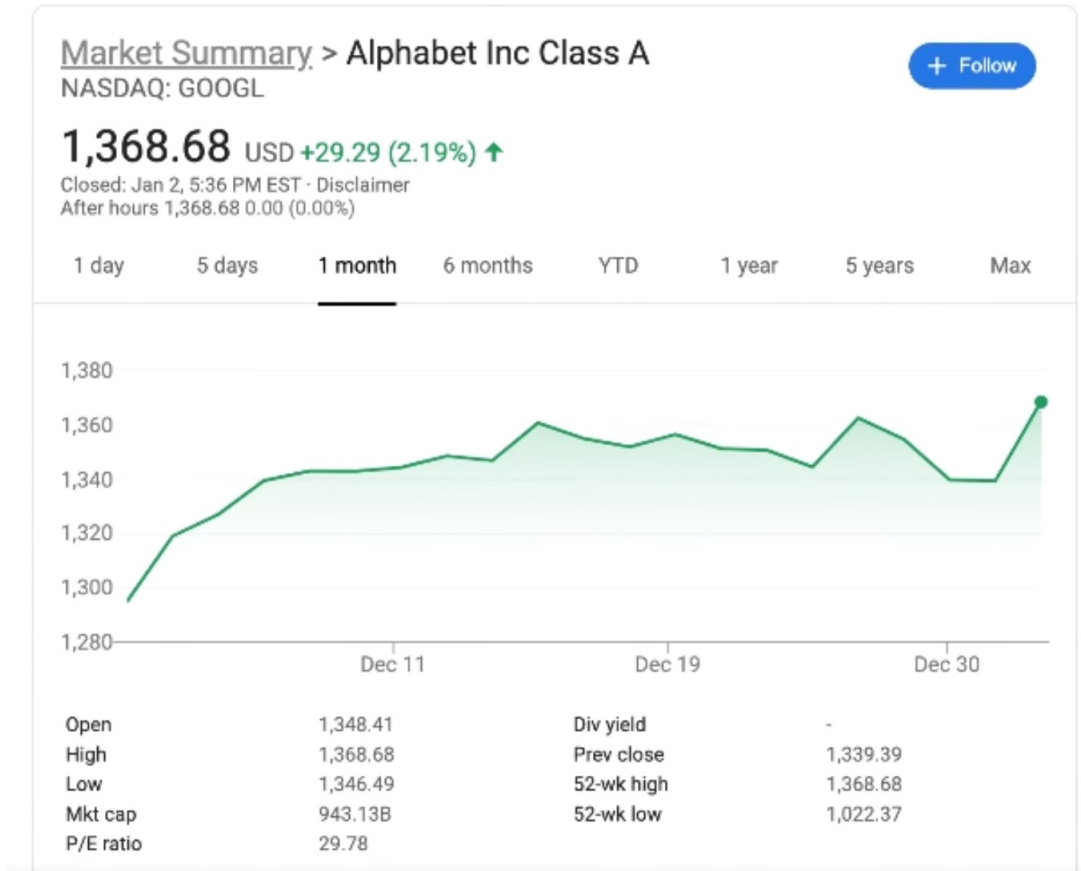
Recurrent Neural Networks

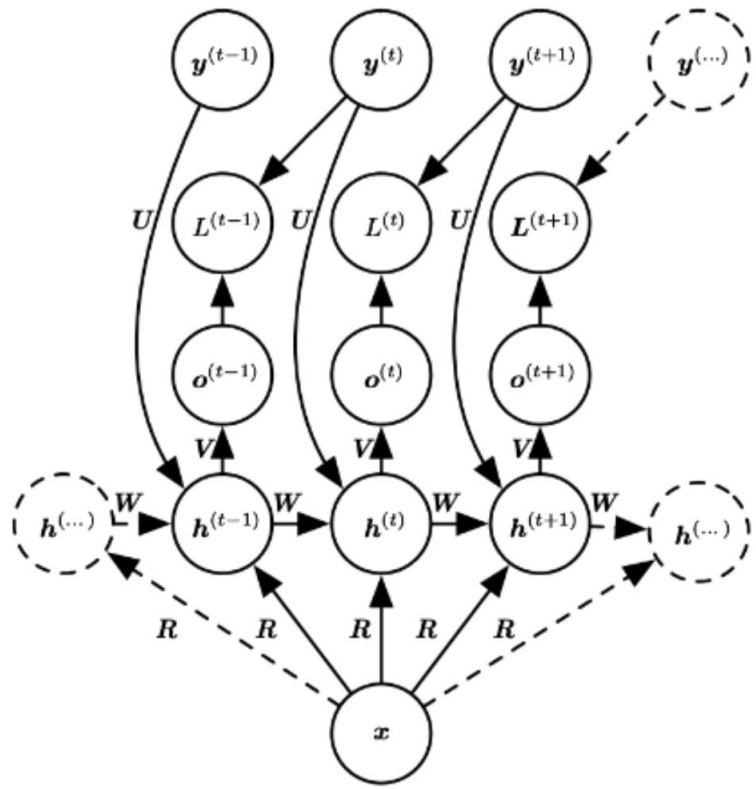
The unicorn is scotland's national animal

Sequences



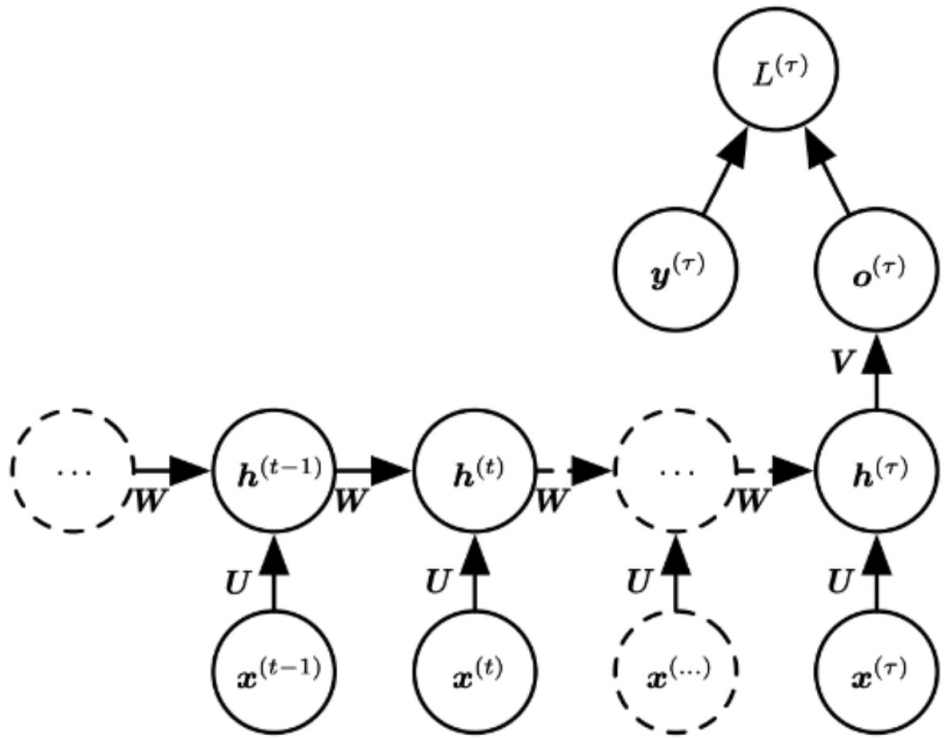
Recurrent Neural Networks

1. Vector-Sequence Models



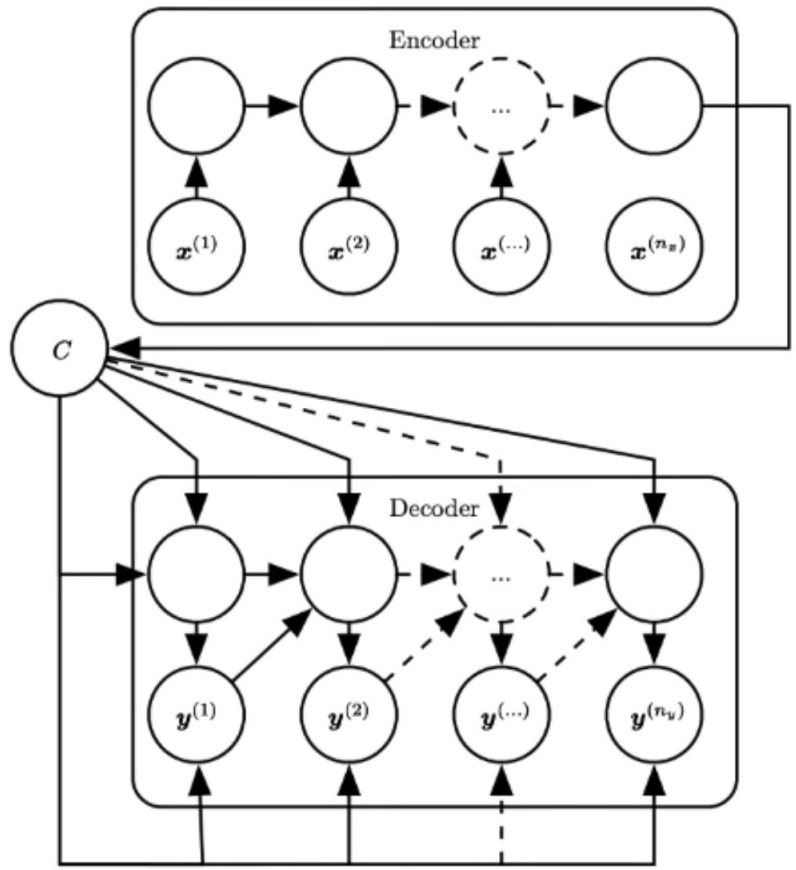
Recurrent Neural Networks

2. Sequence-Vector Models



Recurrent Neural Networks

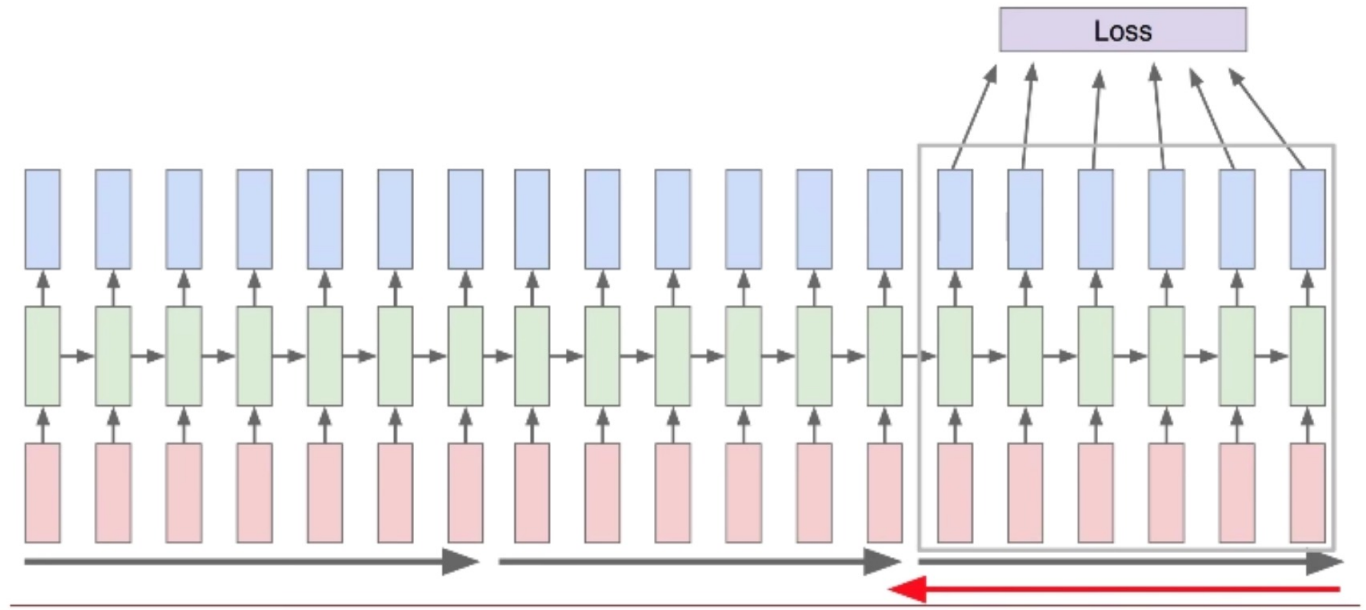
3. Sequence-Sequence Models



Recurrent Neural Networks

Disadvantages

1. Slow to train.



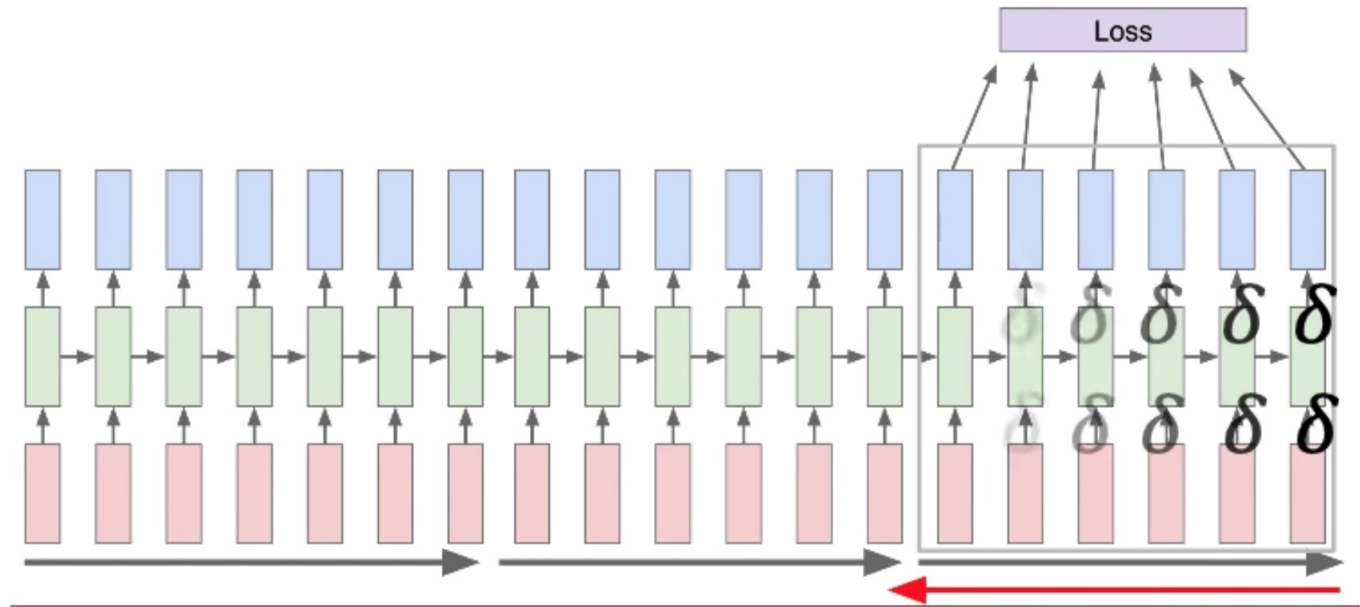
Truncated Backprop Through Time

Recurrent Neural Networks

Disadvantages

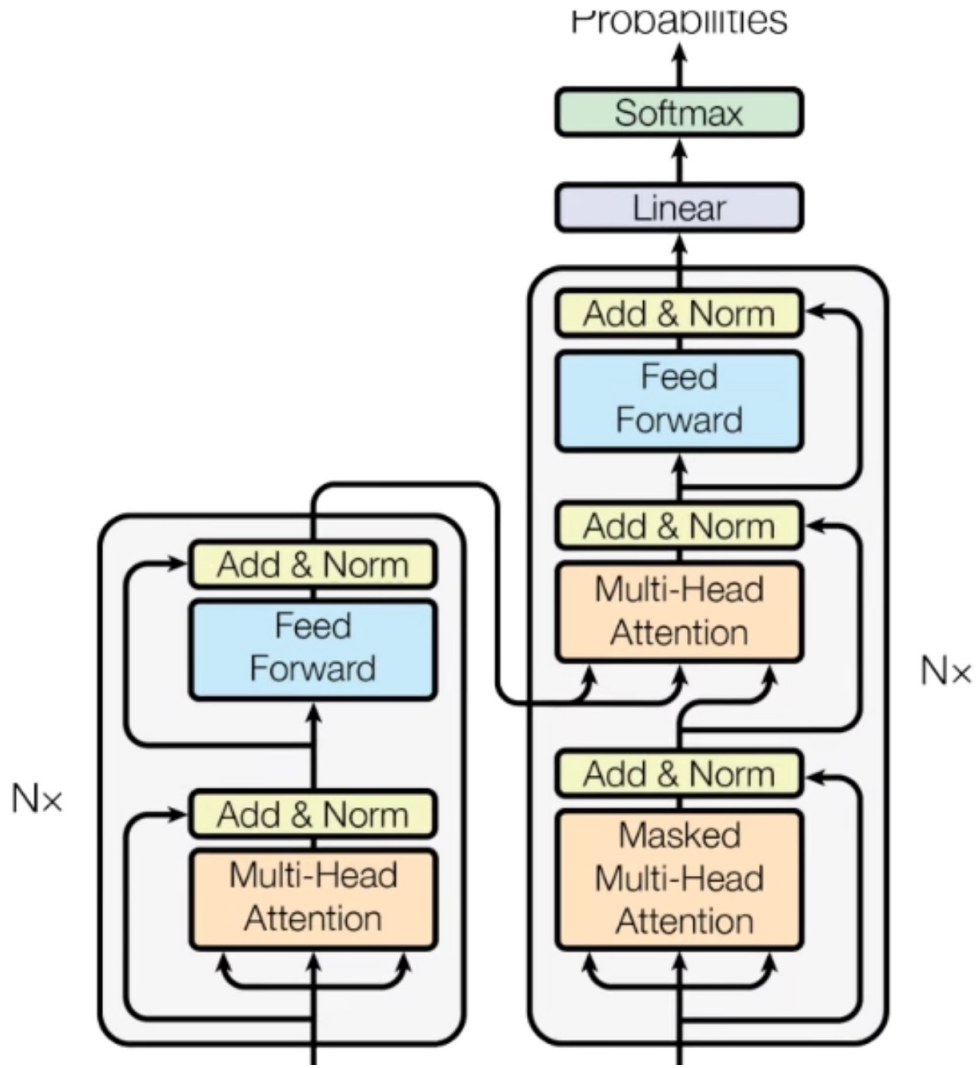
1. Slow to train.

2. Long sequences lead to vanishing/ exploding gradients

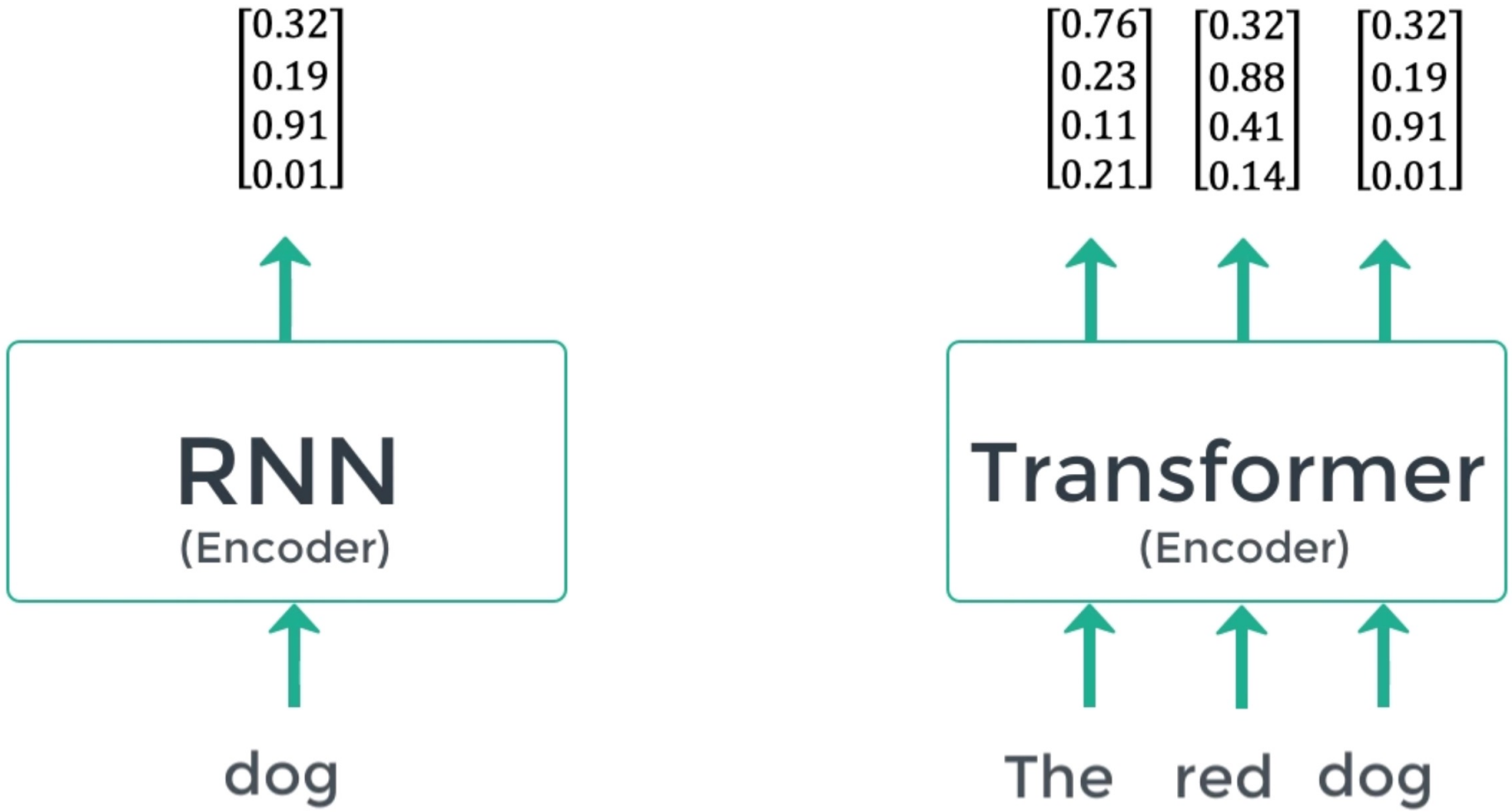


Truncated Backprop Through Time

Transformers



English-French Translation



Transformer Components

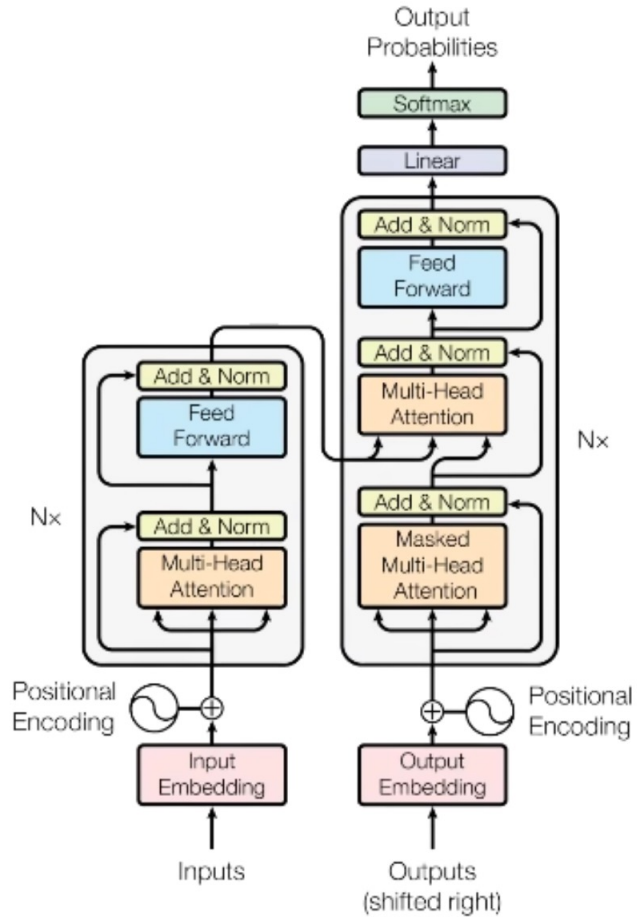
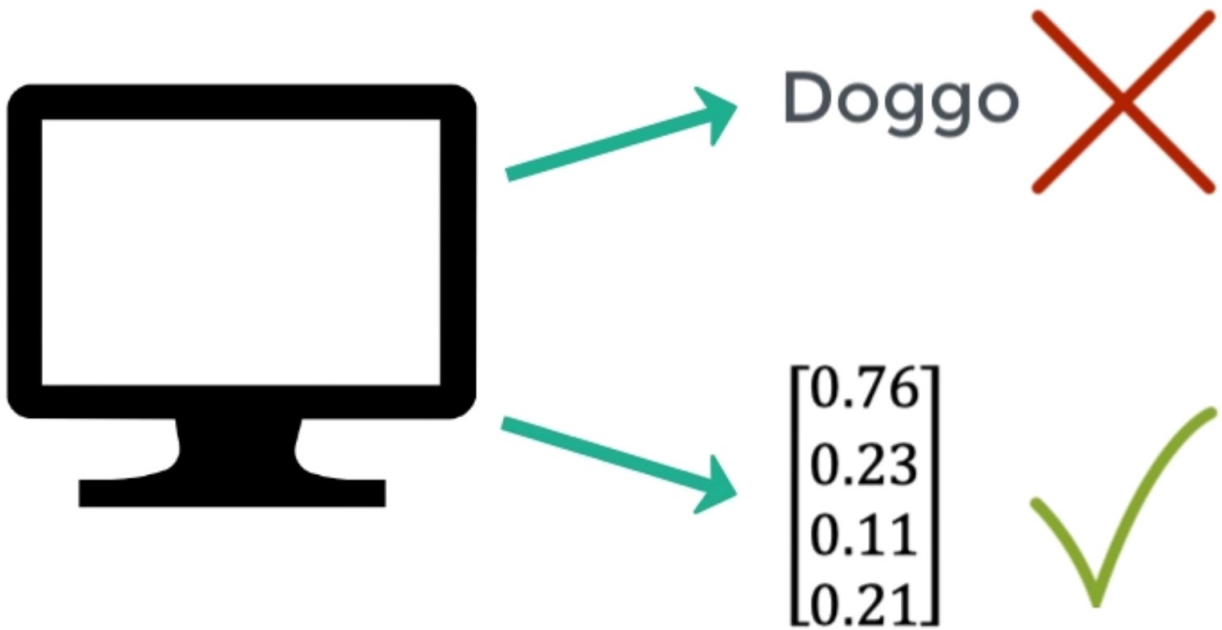


Figure 1: The Transformer - model architecture.

Transformer Components

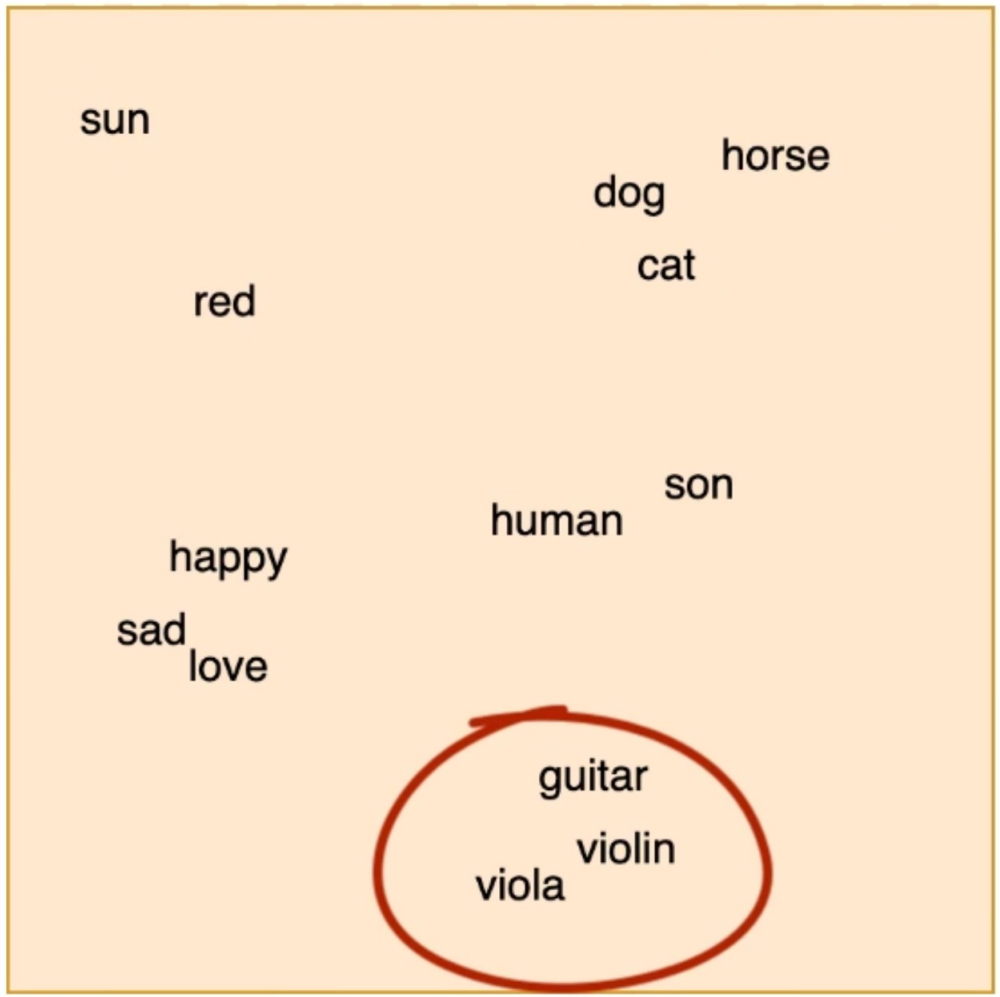
Input Embedding



Transformer Components

Input Embedding

learns



nlp.stanford.edu/projects/glove/

GloVe: Global Vectors for Word Representation

Jeffrey Pennington, Richard Socher, Christopher D. Manning

Introduction

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

Getting started (Code download)

- Download the code (licensed under the Apache License, Version 2.0)
- Unpack the files: unzip GloVe-1.2.zip
- Compile the source: cd GloVe-1.2 && make
- Run the demo script: ./demo.sh
- Consult the included README for further usage details, or ask a question
- The code is also available on GitHub

Download pre-trained word vectors

- Pre-trained word vectors. This data is made available under the Public Domain Dedication and License v1.0 whose full text can be found at: http://www.opendatacommons.org/licenses/pddl/1.0/.
  - Wikipedia 2014 + Gigaword 5 (6B tokens, 400K vocab, uncased, 50d, 100d, 200d, & 300d vectors, 822 MB download): glove.6B.zip
  - Common Crawl (42B tokens, 1.9M vocab, uncased, 300d vectors, 1.75 GB download): glove.42B.300d.zip
  - Common Crawl (840B tokens, 2.2M vocab, cased, 300d vectors, 2.03 GB download): glove.840B.300d.zip
  - Twitter (2B tweets, 27B tokens, 1.2M vocab, uncased, 25d, 50d, 100d, & 200d vectors, 1.42 GB download): glove.twitter.27B.zip
- Ruby script for preprocessing Twitter data

Citing GloVe

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. [pdf] [bib]

Highlights

1. **Nearest neighbors**

Transformer Components

Input Embedding

dog →

→ $\begin{bmatrix} 0.37 \\ 0.99 \\ 0.01 \\ 0.08 \end{bmatrix}$

AJ's dog is a cutie

AJ looks like a dog

Transformer Components

Positional Encoder :vector that gives context based on position of word in sentence

AJ's dog is a cutie → Position 2

AJ looks like a dog → Position 5

Transformer Components

Positional Encoder :vector that gives context based on position of word in sentence

AJ's dog is a cutie → Position 2

AJ looks like a dog → Position 5

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Transformer Components

Positional Encoder :vector that gives context based on position of word in sentence

Embedding of "Dog"

Vector Encoding of position in sentence

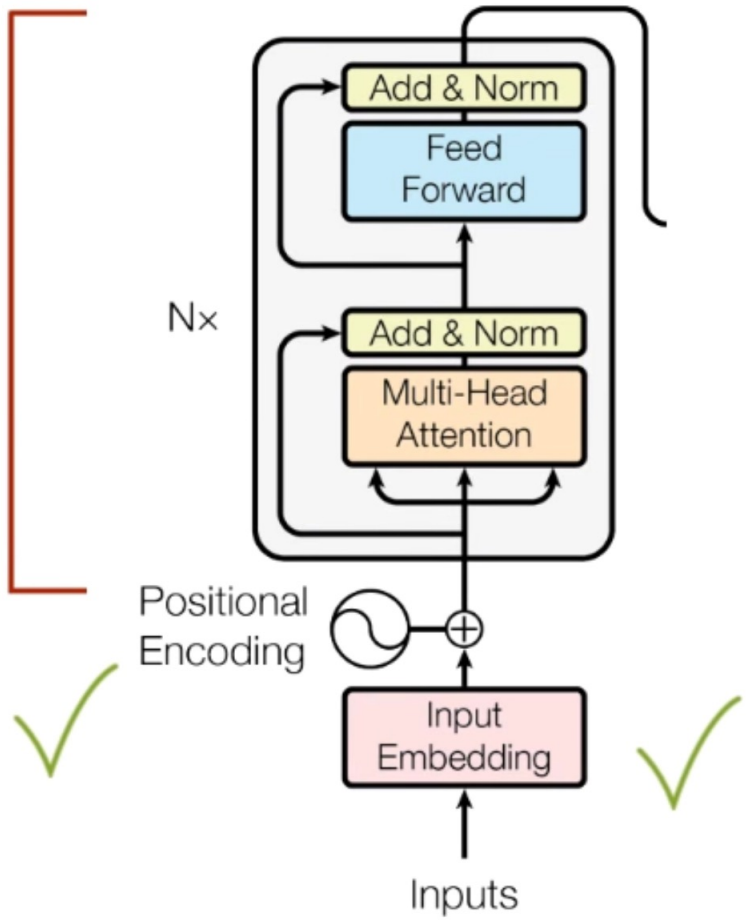
Embedding of Dog (with context info)

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Transformer Components

Encoder Block

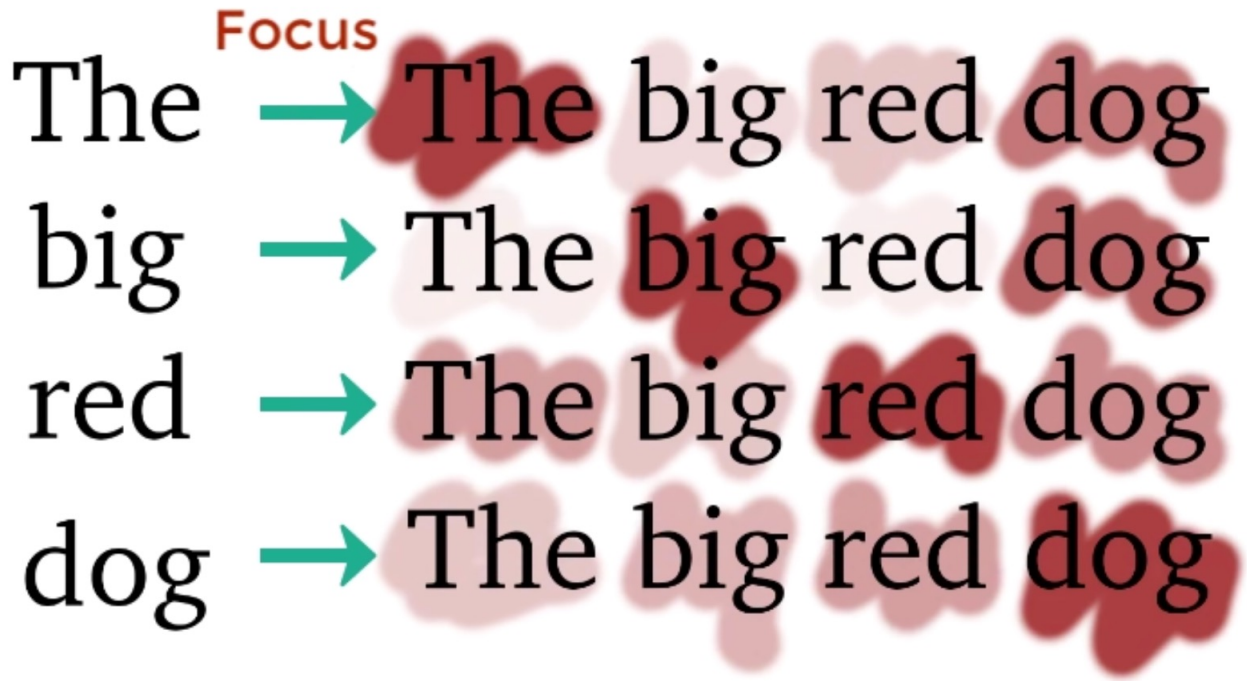


Transformer Components

Attention : What part of the input should we focus?

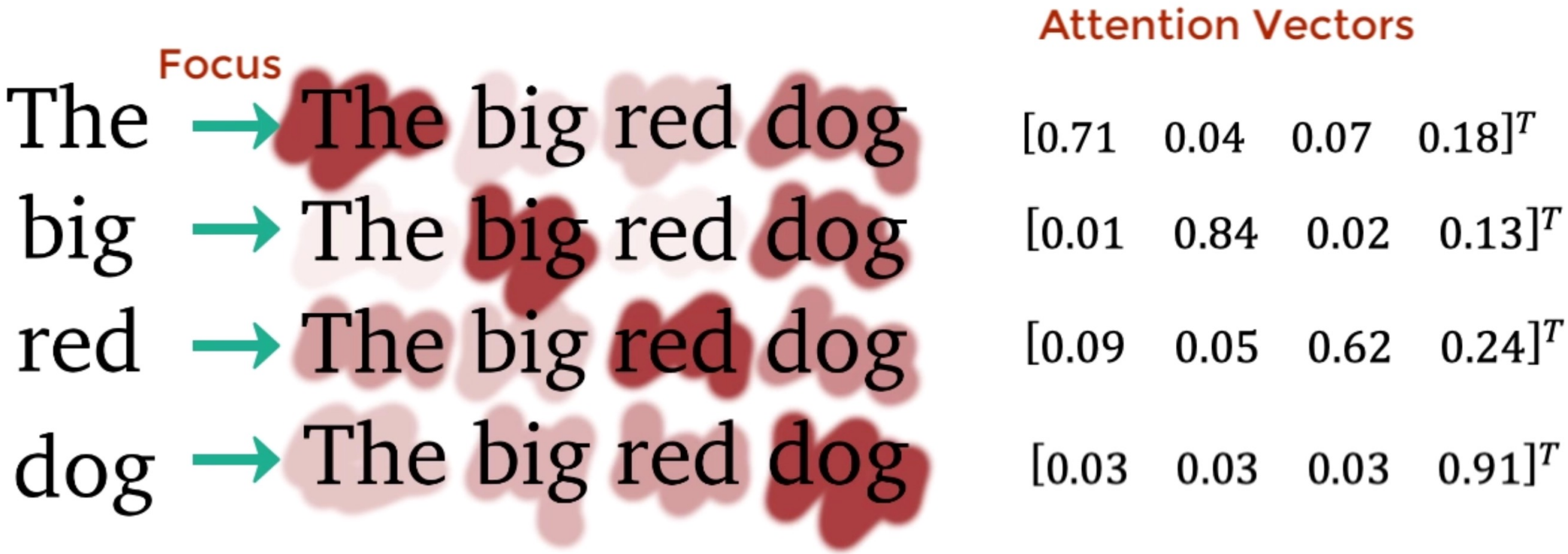
Transformer Components

Attention : What part of the input should we focus?

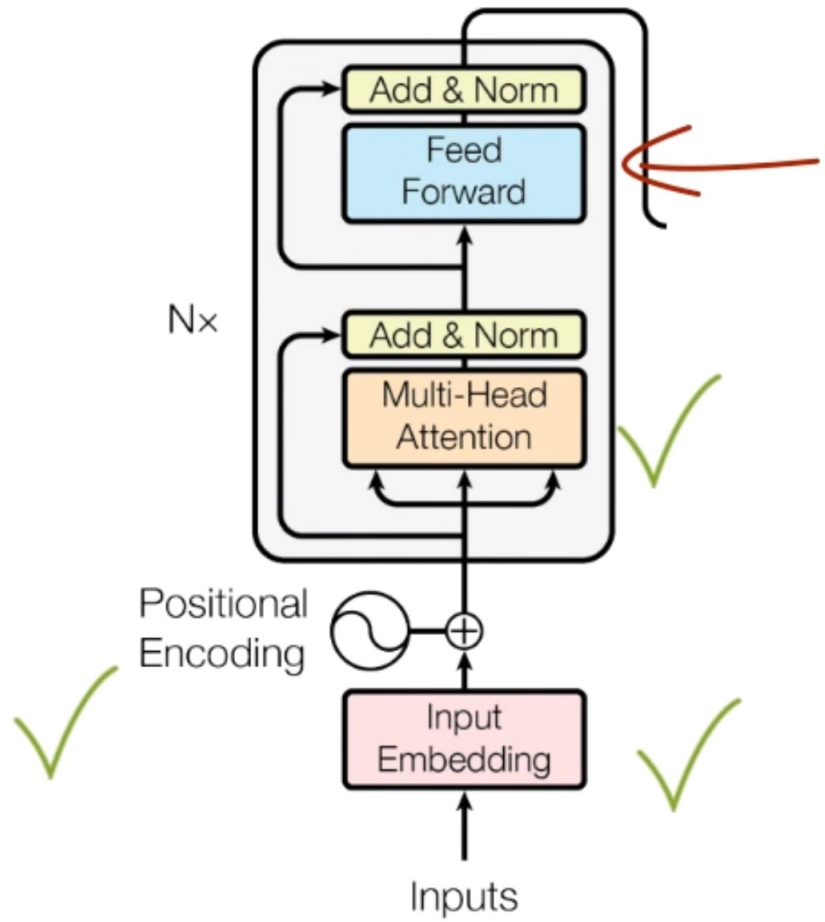


Transformer Components

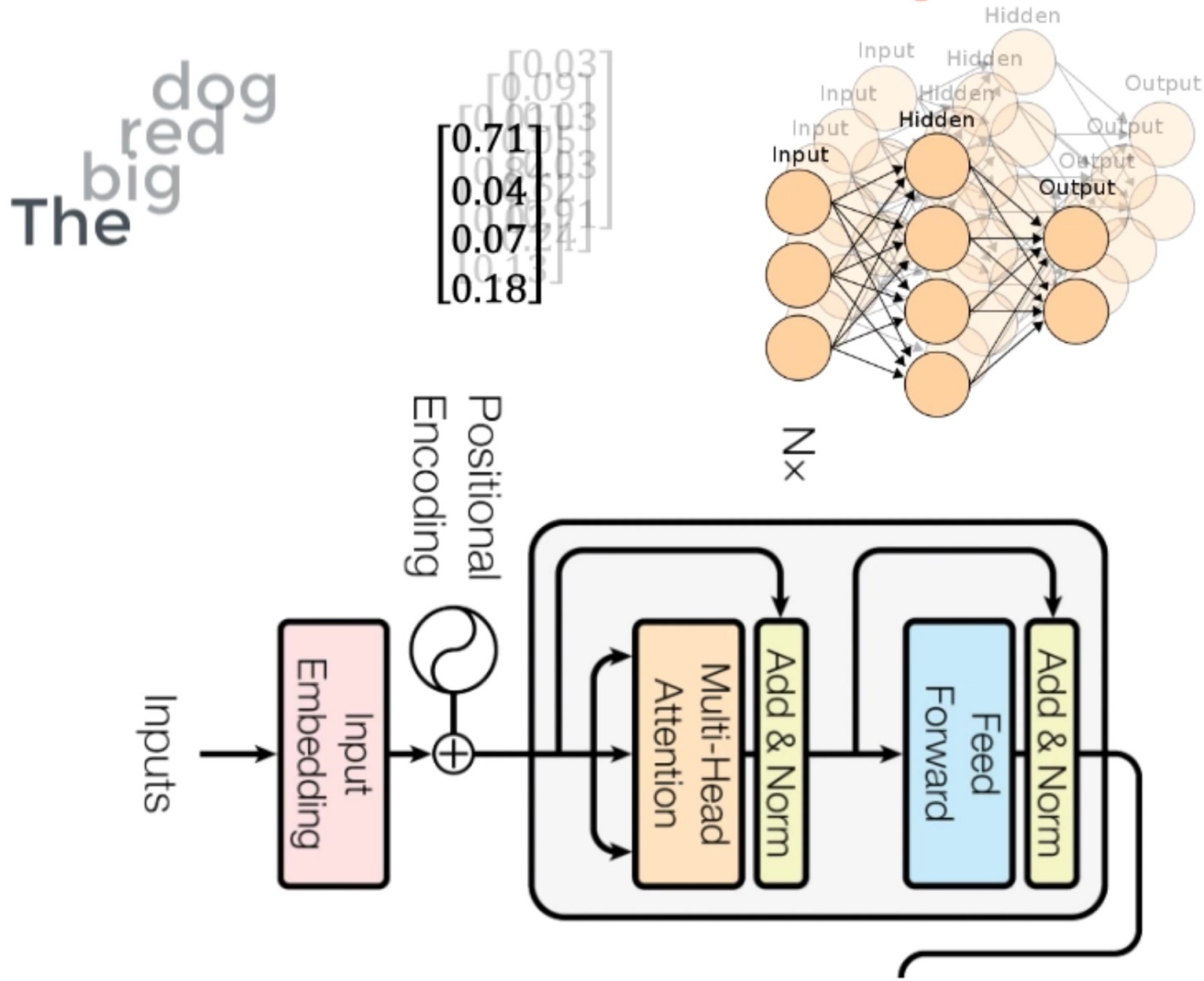
Attention : What part of the input should we focus?



Transformer Components



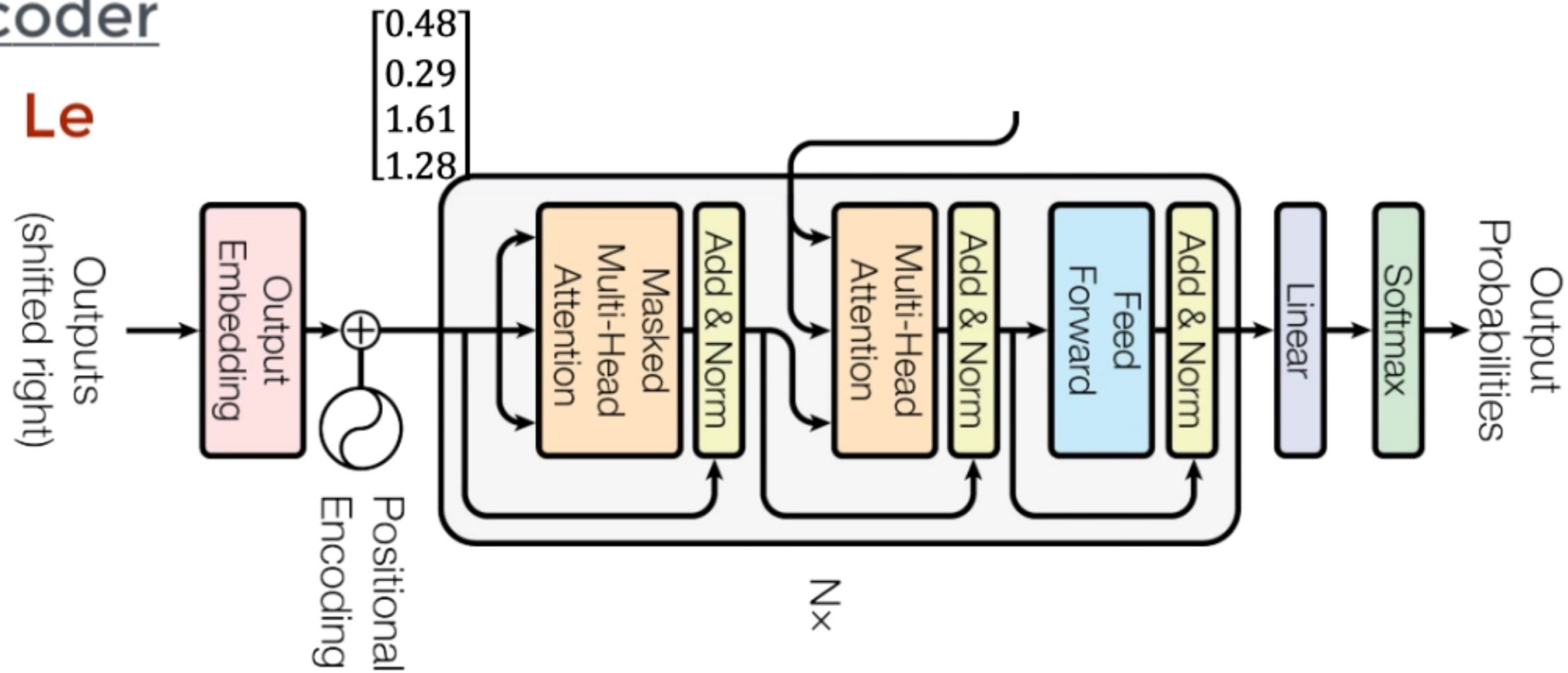
Transformer Components



Transformer Components

Decoder

Le



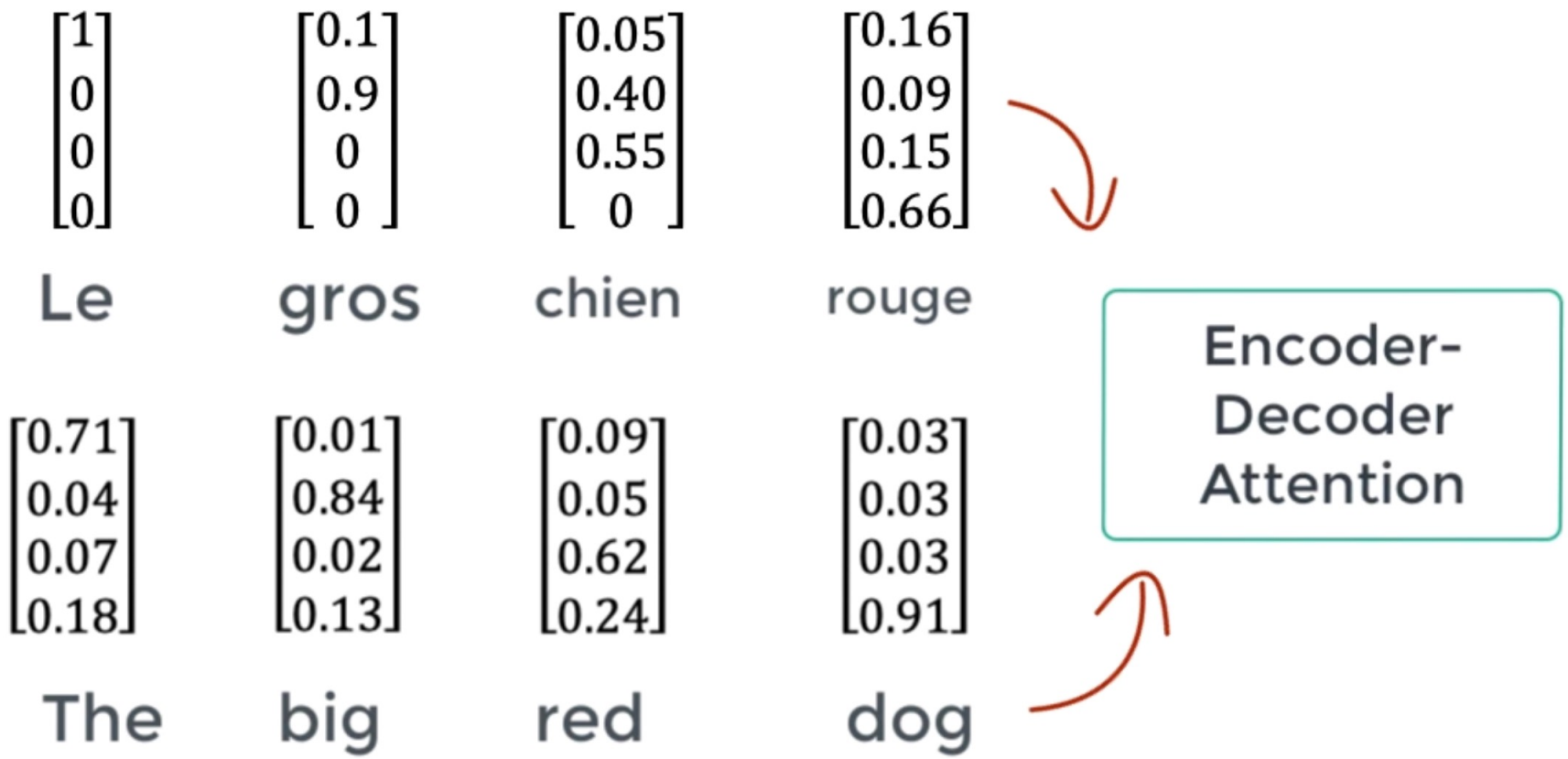
$$\begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Transformer Components

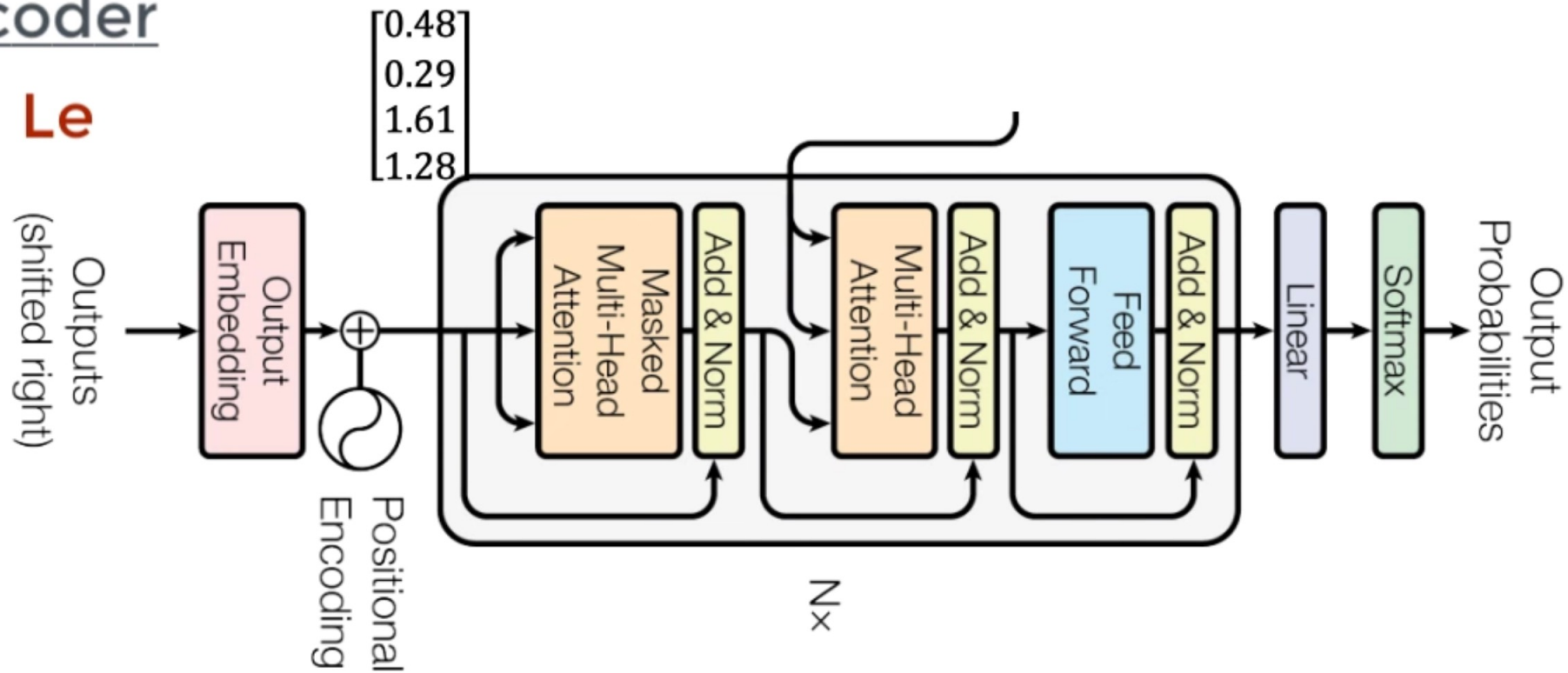
Decoder



Transformer Components

Decoder

Le



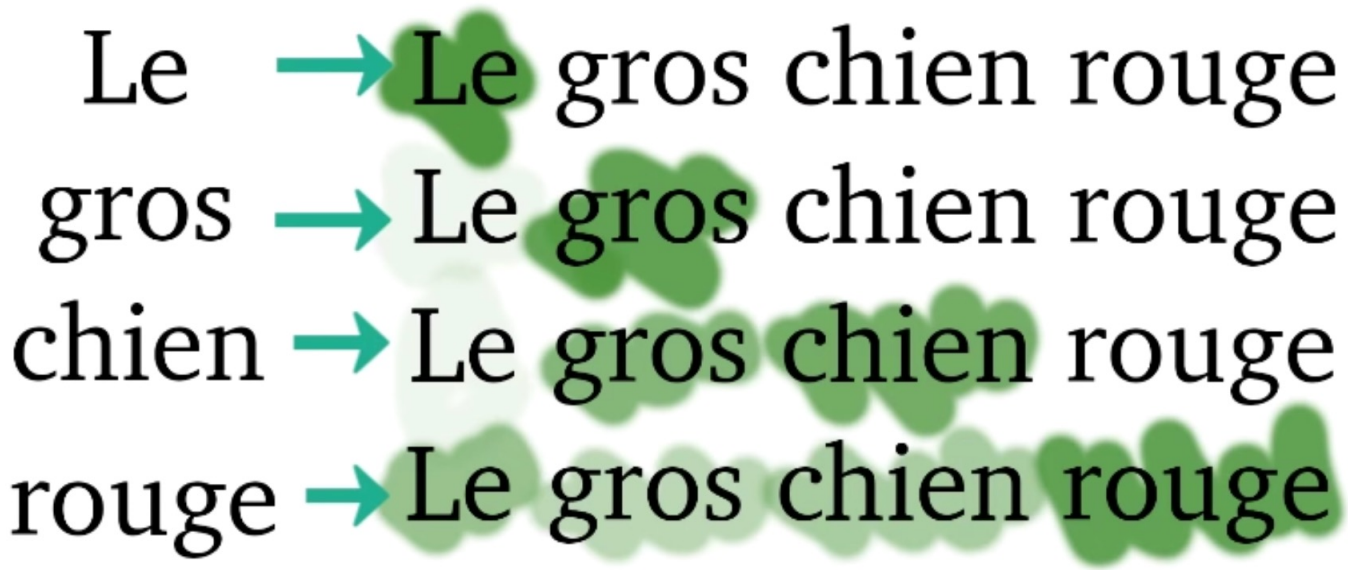
$$\begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

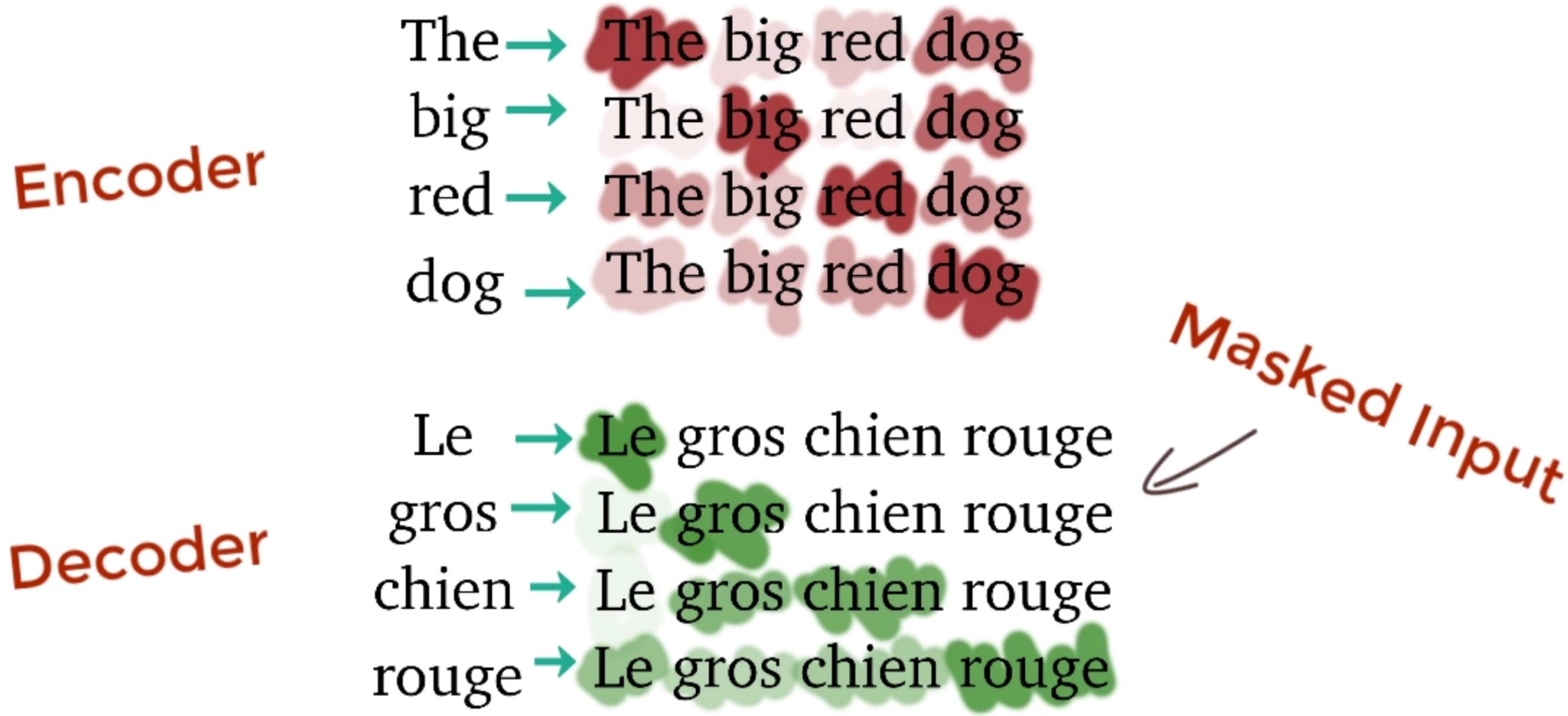
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

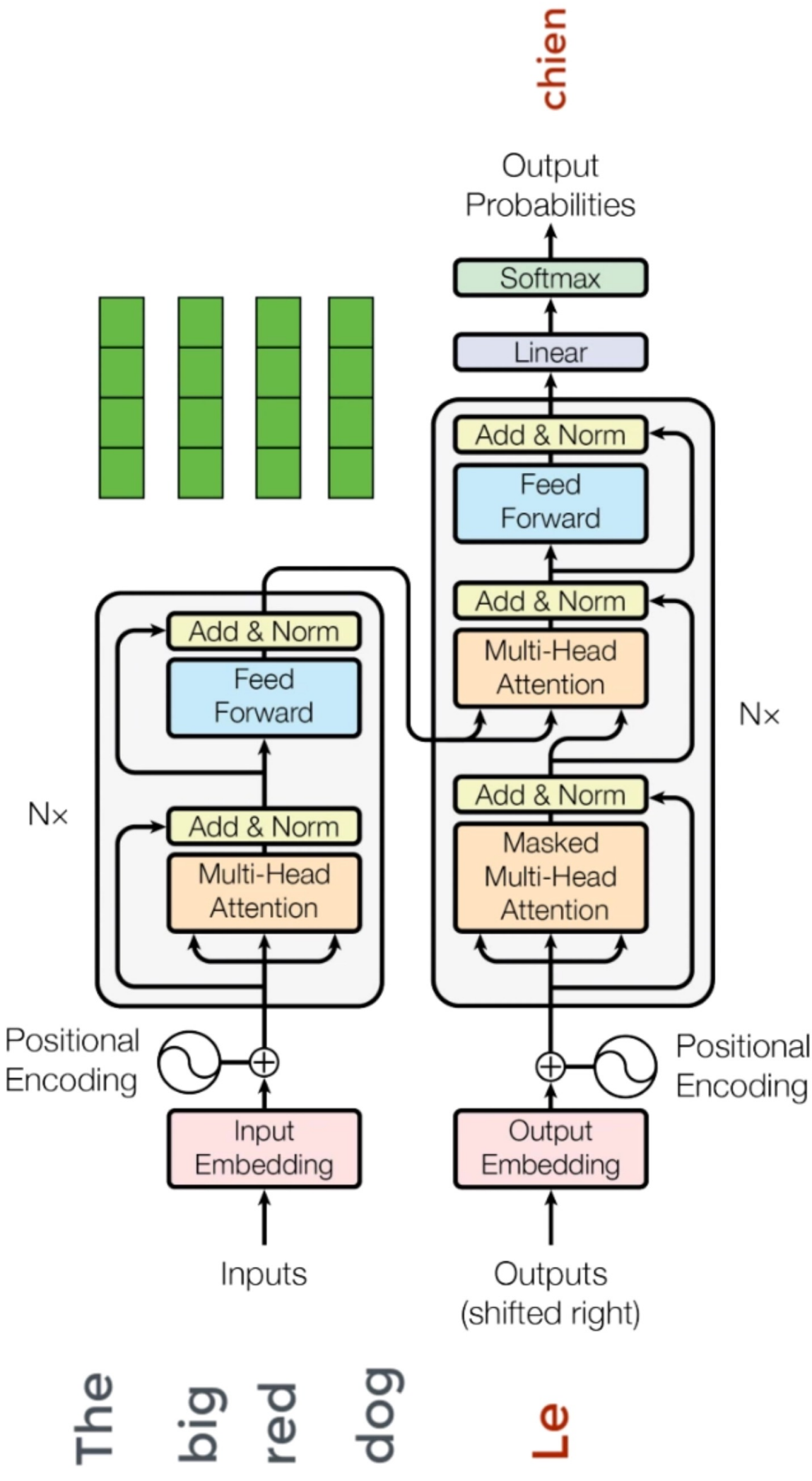
Transformer Components

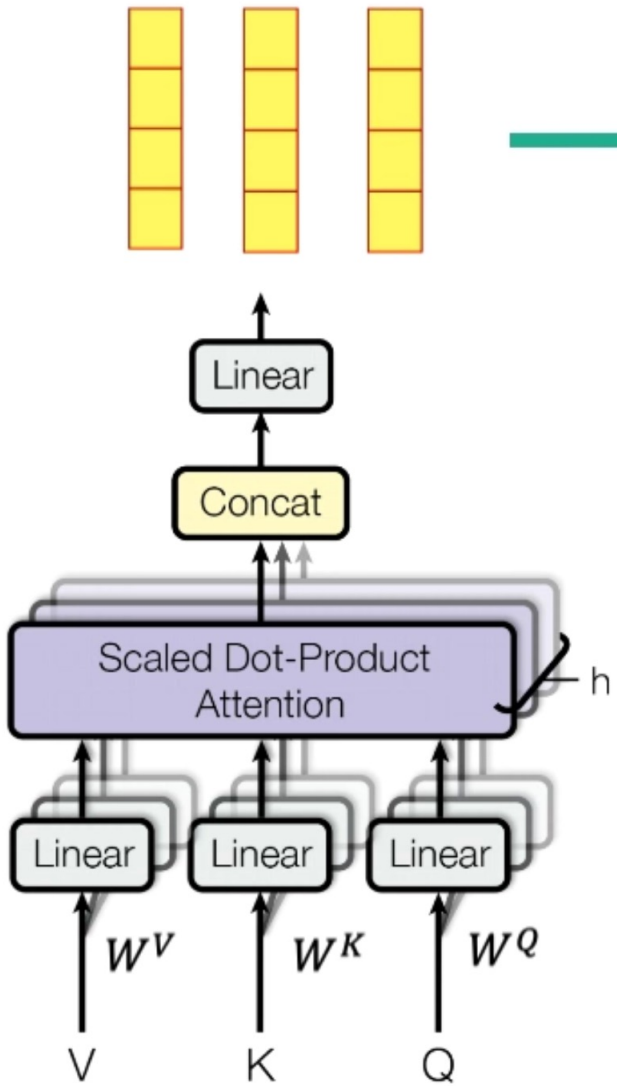
Decoder



Multi-headed Attention




chien
Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)
The
big
red
dog
Le



Feed Forward Neural Network

$$Z = softmax\left(\frac{Q.K^T}{\sqrt{Dimension\ of\ vector\ Q, K\ or\ V}}\right).V$$

Transformer Components