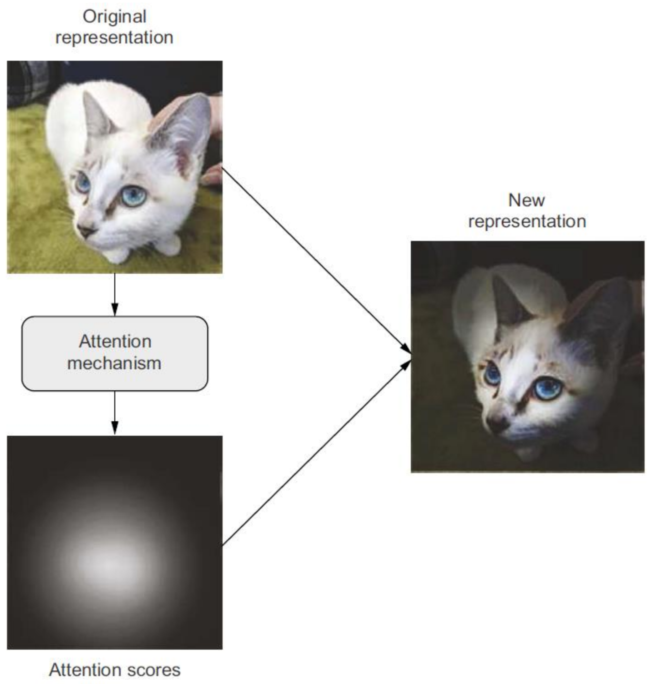
Module 7 – Part 1
Transformers prerequisites
Why Attention is ALL you need?



Jon M. Huntsman School of Business
UtahStateUniversity

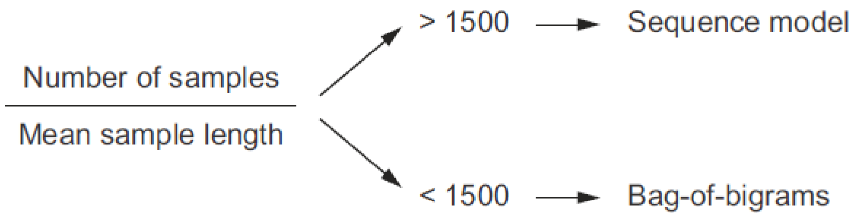
Road map!

- Module 1- Introduction to Deep Learning
- Module 2- Setting up Deep Learning Environment
- Module 3- Machine Learning review (ML fundamentals + models)
- Module 4- Deep Neural Networks (NN and DNN)
- Module 5- Deep Computer Vision (CNN, R-CNN, YOLO, FCN)
- Module 6- Deep Sequence Modeling (RNN, LSTM)
- **Module 7- Transformers (Attention is all you need!)**
- Module 8- Deep Generative Modeling (AE, VAE, GAN)
- Module 9- Deep Reinforcement Learning (DQN, PG)

Jon M. Huntsman School of Business
Utah State University

Transformers for NLP

- Starting in 2017 (Attention is all you need!), transformers started overtaking RNN across most NLP tasks.
- NLP architecture depends on word representation method
  - Discard order and treat text as an unordered set of words → bag-of-words models
  - Respect order and treat words one at a time (steps in timeseries) → recurrent models
- When to use sequence model over bag-of-words?
  - For text classification →



  - For any other NLP task → Transformers

JON M. HUNTSMAN SCHOOL OF BUSINESS
UtahStateUniversity

Pedram Jahangiry

Transformers vs other sequence models

- Transformer architecture is technically order-agnostic, yet it injects word-position information into the representations it processes (hybrid approach)
- Transformers simultaneously look at different parts of a sentence (unlike RNNs) while still being order-aware.

The cat, sat on the mat.

NLP Models	Word order awareness	Context awareness (cross-word interactions)
Bag of unigrams	No	No
Bag of Bigrams	Very limited	No
RNN	Yes	No
Transformer	Yes	Yes

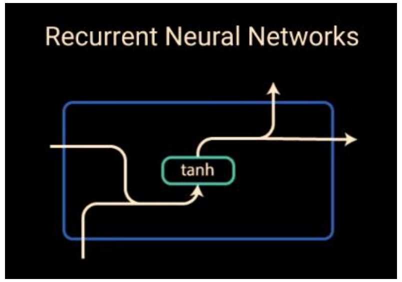
Jon M. Huntsman School of Business
UtahStateUniversity

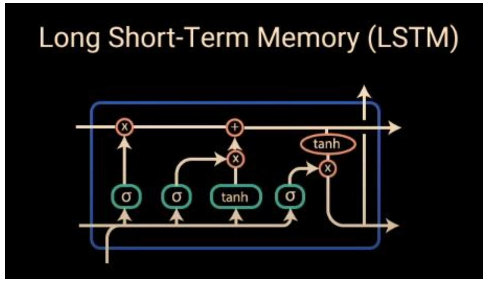
Pedram Jahangiry

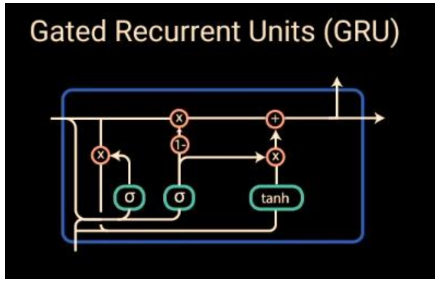
Sequence Modeling Design Criteria

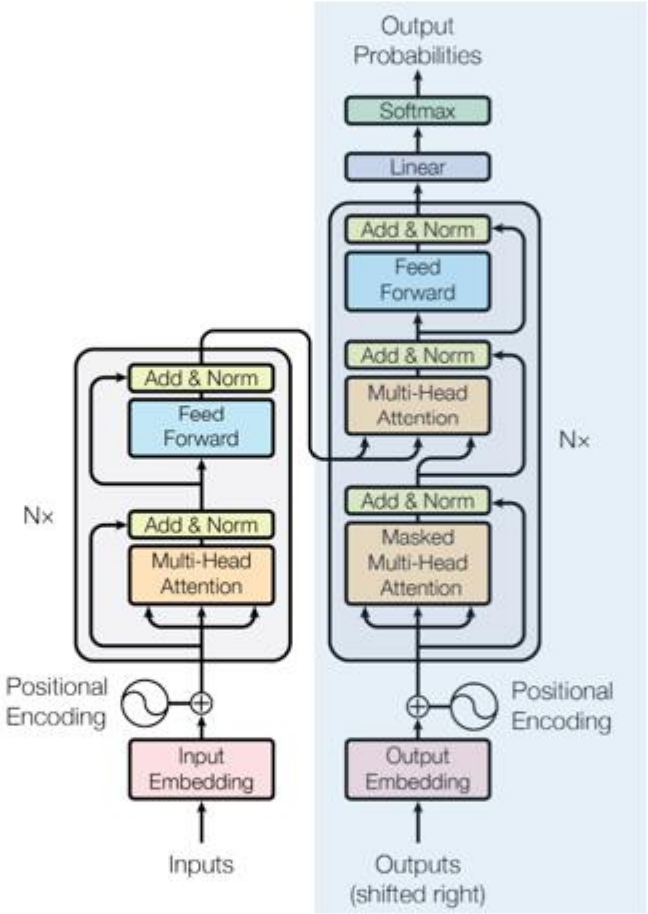
To model sequence data efficiently, we need an architecture that:

- Preserve the order
- Account for long-term dependencies
- Handle different input-length
- Share parameters across the sequence









Jon M. Huntsman School of Business
Utah State University

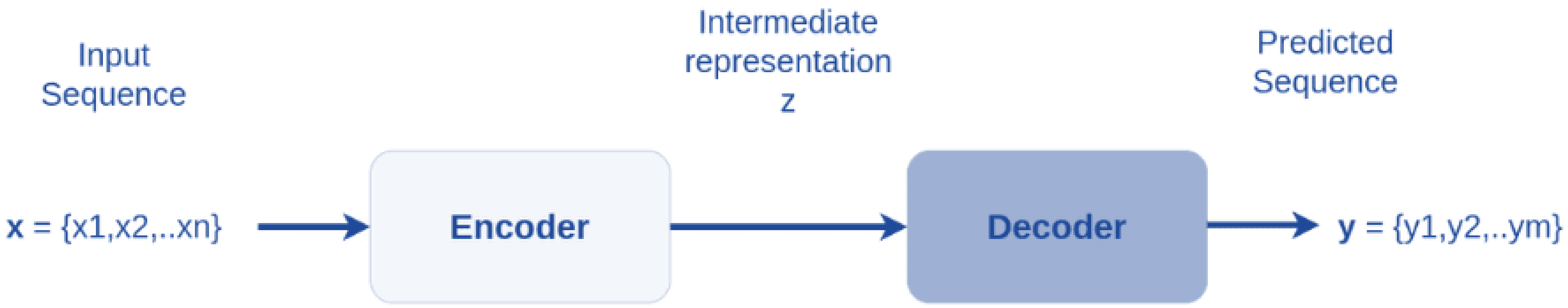
Pedram Jahangiry

Applications of Transformers

- Transformers are taking NLP, Computer vision and reinforcement learning by storm.
- <u>NLP applications:</u>
  - Machine translation, text generation, text summarization, text classification, chatbots, questions answering etc.
  - BERT, GPT
- <u>Computer vision applications:</u>
  - Image captioning, object detection and segmentation
  - ViT (vision transformers)
- <u>Reinforcement learning applications:</u>
  - Game playing, robotics and autonomous driving

Sequence-to-sequence modeling

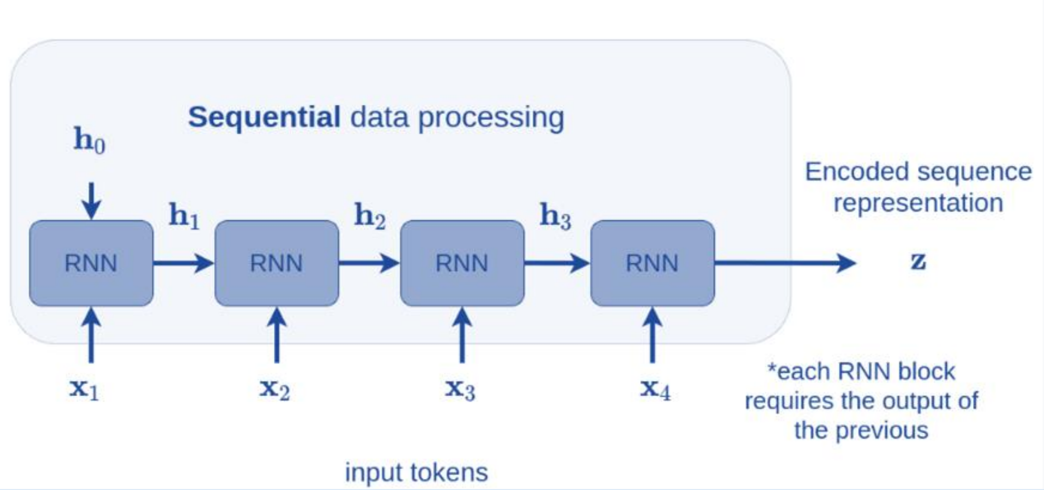
- The goal is to transform an input sequence (source) to a new one (target).

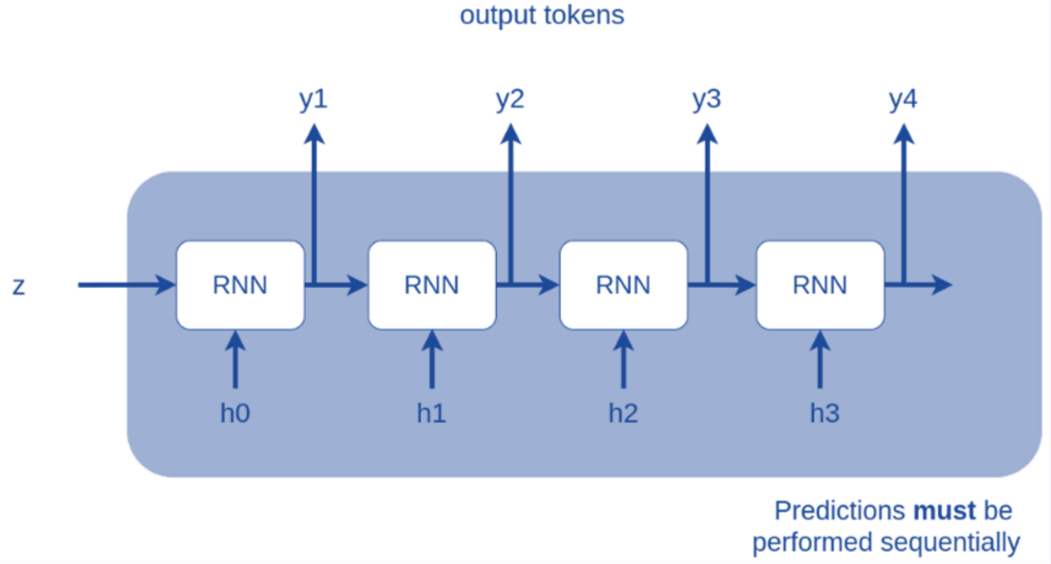


Encoder – Decoder

- Recurrent Neural Networks (RNNs) were the prevailing method for sequence-to-sequence learning until Transformers demonstrated superior performance.

Encoder

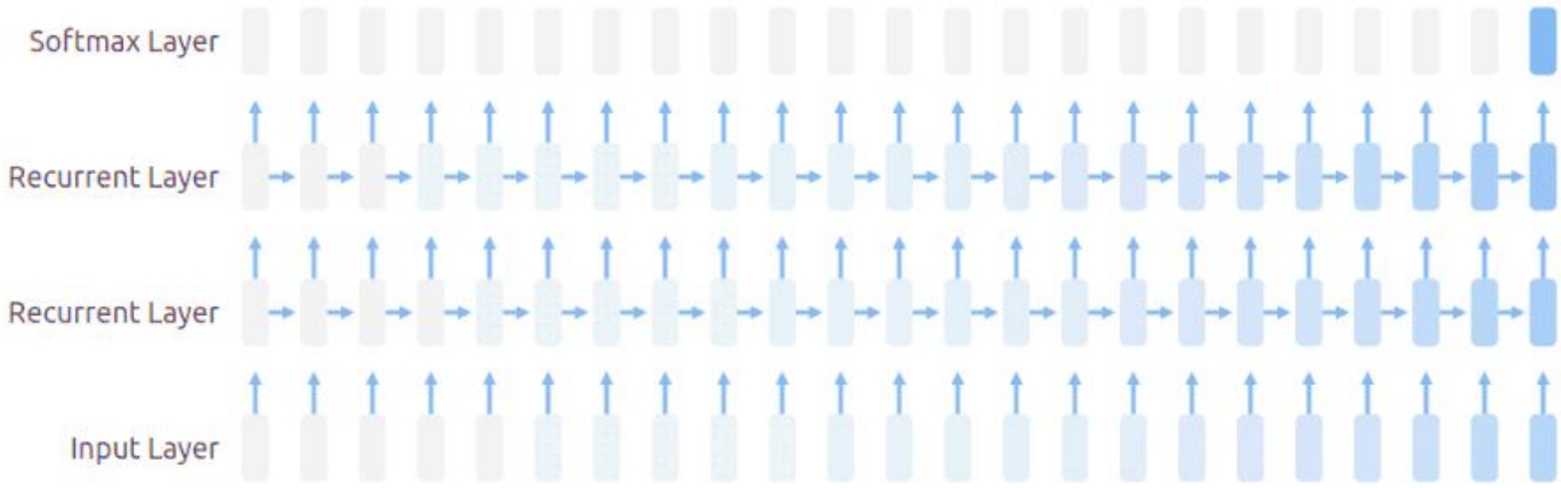




Decoder

Limitations of RNN

- Bottleneck problem: the Encoder state vector(s) must store the entire input sequence representation → Significant limitations on translatable sentence size and complexity
- RNN tends to progressively forget about the past (~100 tokens) and eventually pays more attention to the last parts of the sequence.
- Vanishing gradient problem:



Vanishing Gradient: where the contribution from the earlier steps becomes insignificant in the gradient for the vanilla RNN unit. https://distill.pub/2019/memorization-in-rnns/

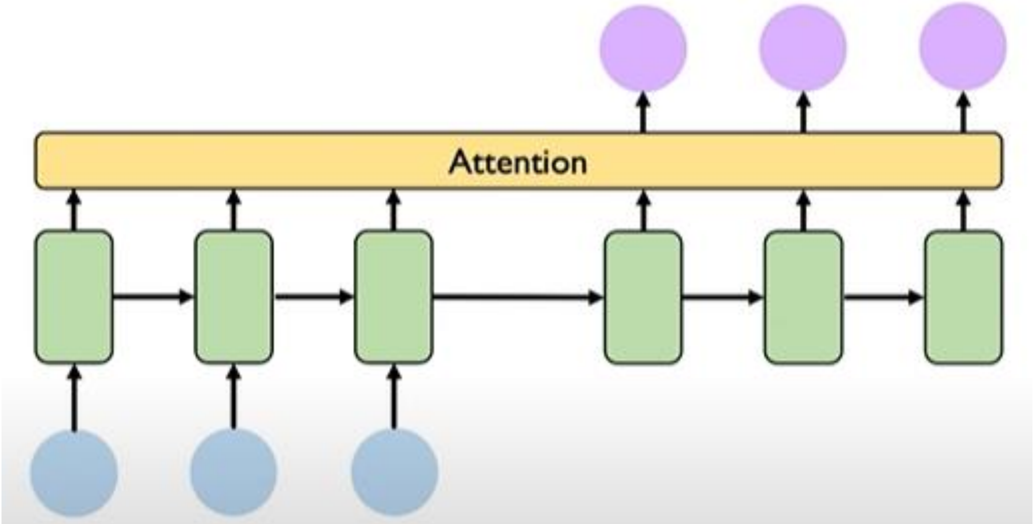
Jon M. Huntsman School of Business
Utah State University

Pedram Jahangiry

Attention is ALL you need!

Attention

- To avoid vanishing gradient, we need to form a direct connection with each timestamp.
- By letting the decoder have an attention mechanism, we relieve the encoder from the burden of having to encode all information in the source sentence (bottleneck problem)
- Attention is a mechanism that allows the model to weigh (score) and focus on specific parts of the input when generating output.
- Attention mechanism can be applied to any encoder and decoder architecture (RNN, LSTM, GRU, CNN, etc)



Attention, deep dive

- Introduction to Attention Mechanisms: Inspired by Brandon Rohrer's High-Level Example
- Attention mechanism as a selective second-order model with skips

Transformers from Scratch

Brandon Rohrer

I procrastinated a deep dive into transformers for a few years. Finally the discomfort of not knowing what makes them tick grew too great for me. Here is that dive.

Transformers were introduced in this 2017 paper as a tool for sequence transduction—converting one sequence of symbols to another. The most popular examples of this are translation, as in English to German. It has also been modified to perform sequence completion—given a starting prompt, carry on in the same vein and style. They have quickly become an indispensible tool for research and product development in natural language processing.

Before we start, just a heads-up. We're going to be talking a lot about matrix multiplications and touching on backpropagation (the algorithm for training the model), but you don't need to know any of it beforehand. We'll add the concepts we need one at a time, with explanation.

Attention, deep dive

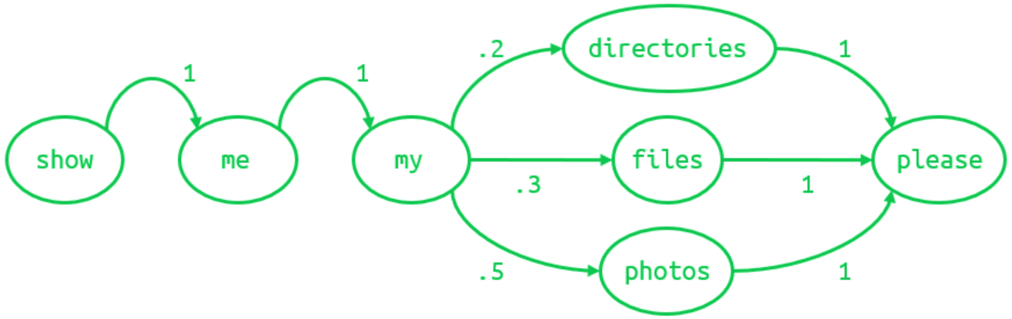
- What does it mean mathematically?
- Vocabulary: collection of symbols in each sequence
- Converting symbols to numbers: One-hot encoding
- Dot product can be used to measure similarity
- One-hot vectors can pull out a particular row of a matrix!

$$\underset{A}{\begin{bmatrix}1&0&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}}\underset{B}{\begin{bmatrix}.2&.9\\.7&0\\.8&.3\\.1&.4\end{bmatrix}}=\begin{bmatrix}.2&.9\\.1&.4\\.8&.3\end{bmatrix}$$

Brandon Rohrer: https://e2eml.school/blog.html

First order sequence model

- Example:
  - Show me my directories please
  - Show me my files please
  - Show me my photos please



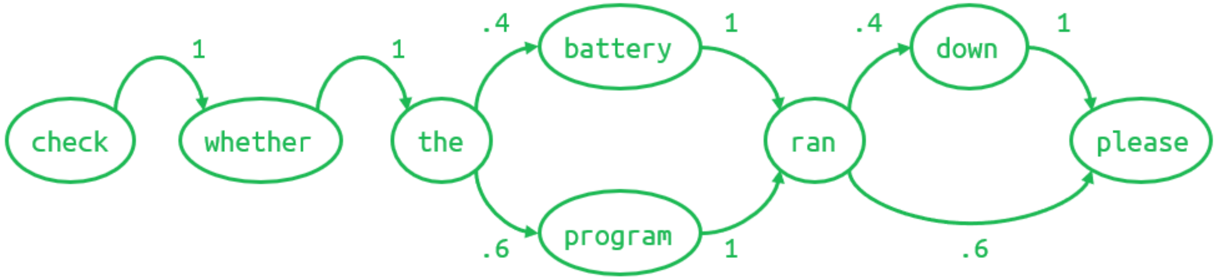
- Vocabulary size = 7 {directories, files, me, my, photos, please, show}.
- Markov chain transition model
- Matrix form

$$
\begin{array}{ccccccc}
\text{directories} & \text{files} & \text{me} & \text{my} & \text{photos} & \text{please} & \text{show} \\
\hline
0 & 0 & 0 & 1 & 0 & 0 & 0
\end{array}
\;
\begin{array}{ccccccc}
\text{directories} & \text{files} & \text{me} & \text{my} & \text{photos} & \text{please} & \text{show} \\
\hline
0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 \\
.2 & .3 & 0 & 0 & .5 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0
\end{array}
=
\begin{array}{ccccccc}
\text{directories} & \text{files} & \text{me} & \text{my} & \text{photos} & \text{please} & \text{show} \\
\hline
.2 & .3 & 0 & 0 & .5 & 0 & 0
\end{array}
$$

Second order sequence model

- First order Markov model only looks at the single most recent word.
- Predicting based on only one last word is hard! Let's consider the two most recent words!
- Example: (a 40/60 proportion)
  - Check whether the battery ran down please.
  - Check whether the program ran please.

- First order model:
- How can we remove the uncertainty after the word "ran"?



Second order sequence model

- Check whether the battery ran down please.
- Check whether the program ran please.
- Vocabulary: {battery, check, down, please, program, ran, the, whether} size = 8

	battery	check	down	please	program	ran	the	whether
battery	0	0	0	0	0	1	0	0
check	0	0	0	0	0	0	0	1
down	0	0	0	1	0	0	0	0
please	0	0	0	0	0	0	0	0
program	0	0	0	0	0	1	0	0
ran	0	0	.4	.6	0	0	0	0
the	.4	0	0	0	.6	0	0	0
whether	0	0	0	0	0	0	1	0

	battery	check	down	please	program	ran	the	whether
battery ran	0	0	1	0	0	0	0	0
check whether	0	0	0	0	0	0	1	0
program ran	0	0	0	1	0	0	0	0
the battery	0	0	0	0	0	1	0	0
the program	0	0	0	0	0	1	0	0
ran down	0	0	0	1	0	0	0	0
whether the	.4	0	0	0	.6	0	0	0
⋮	0	0	0	0	0	0	0	0

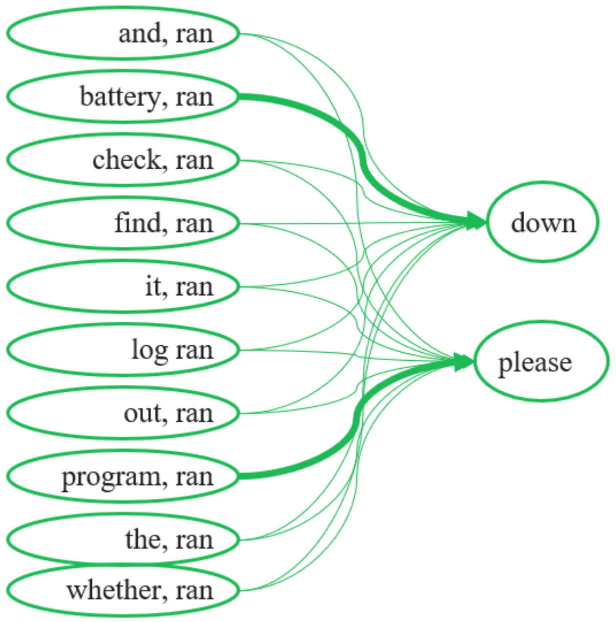
Jon M. Huntsman School of Business
Utah State University

Pedram Jahangiry

Higher order sequence models?

- Check the program log and find out whether it ran please.
- Check the battery log and find out whether it ran down please.
- What comes after the word “ran”? It is unreasonable to investigate 9th order sequence model! (Vocab Size^9) combinations!
- Solution: Second order sequence model with skips

	and	battery	check	down	find	it	log	out	please	program	ran	the	whether
and, ran				.5					.5				
battery, ran				1					0				
check, ran				.5					.5				
down, ran				0									
find, ran				.5					.5				
it, ran				.5					.5				
log, ran				.5					.5				
out, ran				.5					.5				
please, ran													
program, ran				0					1				
ran, ran													
the, ran				.5					.5				
whether, ran				.5					.5				



Masking

- ==Masking==: Crossing out all the uninformative feature votes
- The only important rows are *battery, ran* and *program, ran*. We could ==mask== everything else!

<u>Check the program log and find out whether it ran please.</u>

	and, ran	battery, ran	check, ran	down, ran	find, ran	it, ran	log, ran	out, ran	please, ran	program, ran	ran, ran	the, ran	whether, ran
feature activities	1	0	1	0	1	1	1	1	0	1	0	1	1
*													
mask	0	1	0	0	0	0	0	0	0	1	0	0	0
=													
masked feature activities	0	0	0	0	0	0	0	0	0	1	0	0	0

Jon M. Huntsman School of Business
UtahStateUniversity

Pedram Jahangiry

Masking: selective second order model with skips

- The mask has the effect of hiding a lot of the transition matrix
- In this example, it hides the combination of ran with everything except battery and program, leaving just the features that matter
- This process of selective masking is the attention thing!

	and	battery	check	down	find	it	log	out	please	program	ran	the	whether
and, ran													
battery, ran				1					0				
check, ran													
down, ran													
find, ran													
it, ran													
log, ran													
out, ran													
please, ran													
program, ran				0					1				
ran, ran													
the, ran													
whether, ran													

Attention as Matrix Multiplication

- Stack the mask vectors for every word into a matrix (Keys)
- Use one-hot representation of the most recent word (Query) to pull out the relevant mask (attention score)
- Wight the tokens in the input sequence (Values) by the attention scores to create the context vector.

Check the program log and find out whether it ran please.

one-hot feature vector (query: Q)

0	0	0	0	0	0	0	0	0	0	0	1	0	0

*

all the masks (keys: K^T)

														and
														battery
														check
														down
														find
														it
														log
														out
														please
														program
0	1	0	0	0	0	0	0	0	0	1	0	0	0	ran
														the
														whether

=

feature-specific mask

0	1	0	0	0	0	0	0	0	0	1	0	0	0

Connecting the dots

- This high-level description can be connected to the self-attention mechanism used in Transformers as follows: selective second order model with skips
- Selective: Self-attention mechanisms learn to weigh different parts of the input based on their relevance to the current computation.
  - In the context of Transformers, this is achieved by using the dot product between the **query** and **key** vectors to calculate attention scores
- Second order: After obtaining the attention scores, the self-attention mechanism computes a weighted sum of the value vectors. This is a second order operation (attentions scores * value)
- Model with skips: Self-attention mechanisms can capture long-range dependencies in the input by effectively "skipping" over less relevant parts

Ways to compute attention

- There are several different ways to compute attention scores, including:

✓ Dot-product Attention

✓ Scaled Dot-Product Attention

✓ Additive Attention

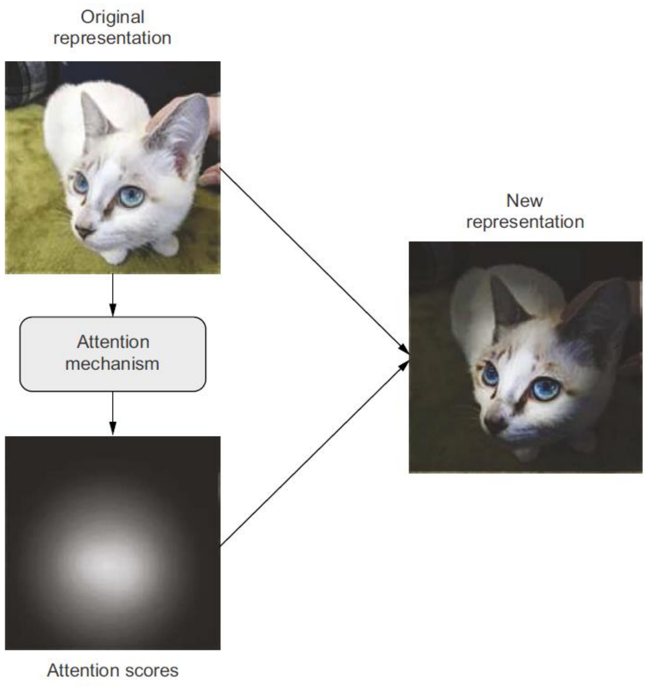
✓ Multiplicative Attention

✓ Location-Based Attention

Name	Alignment score function	Citation
Content-base attention	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$	Graves2014
Additive(*)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$	Bahdanau2015
Location-Base	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \boldsymbol{s}_t)$ Note: This simplifies the softmax alignment to only depend on the target position.	Luong2015
General	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \mathbf{W}_a \boldsymbol{h}_i$ where $\mathbf{W}_a$ is a trainable weight matrix in the attention layer.	Luong2015
Dot-Product	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$	Luong2015
Scaled Dot-Product(^)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \frac{\boldsymbol{s}_t^\top \boldsymbol{h}_i}{\sqrt{n}}$ Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.	Vaswani2017

Self-Attention

- Self attention is a variation of the attention mechanism where the input and output sequences are the same, meaning the model is attending to its own input.
- Self-Attention starts by computing scores for a set of features. High score → more relevant
- Self attention allows the model to relate different parts of the input sequence to each other, capturing dependencies and relationships within the sequence itself
- Have we seen this idea before? Max Pooling and TF-IDF

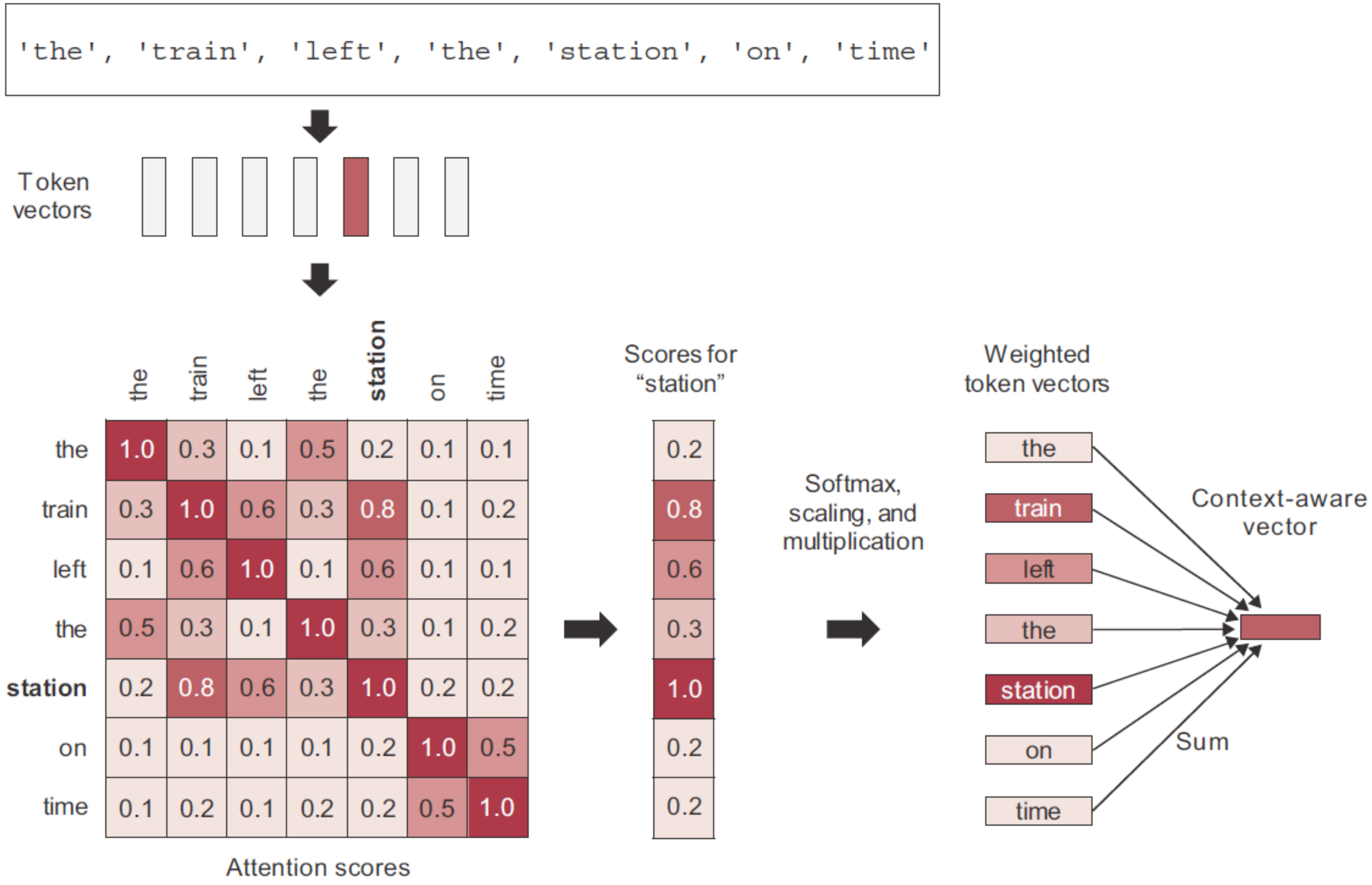


Jon M. Huntsman School of Business
Utah State University

Pedram Jahangiry

Self-Attention (context-aware representation)

- Self-attention helps to adjust the representation of a token by considering the information from related tokens in the input sequence → context awareness

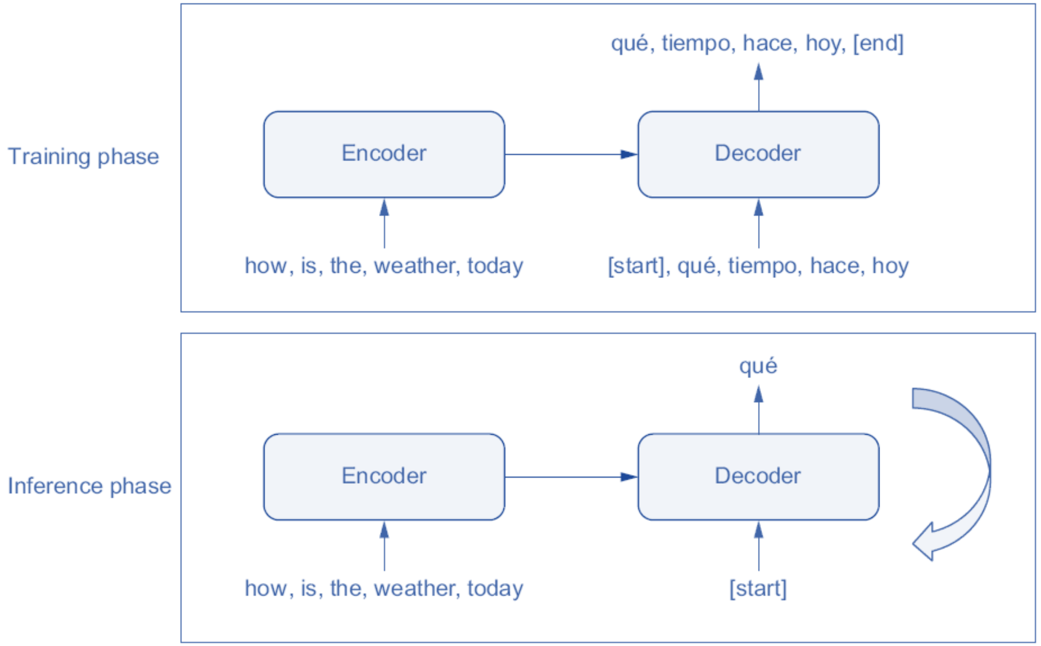


JON M. HUNTSMAN SCHOOL OF BUSINESS
UtahStateUniversity

Pedram Jahangiry

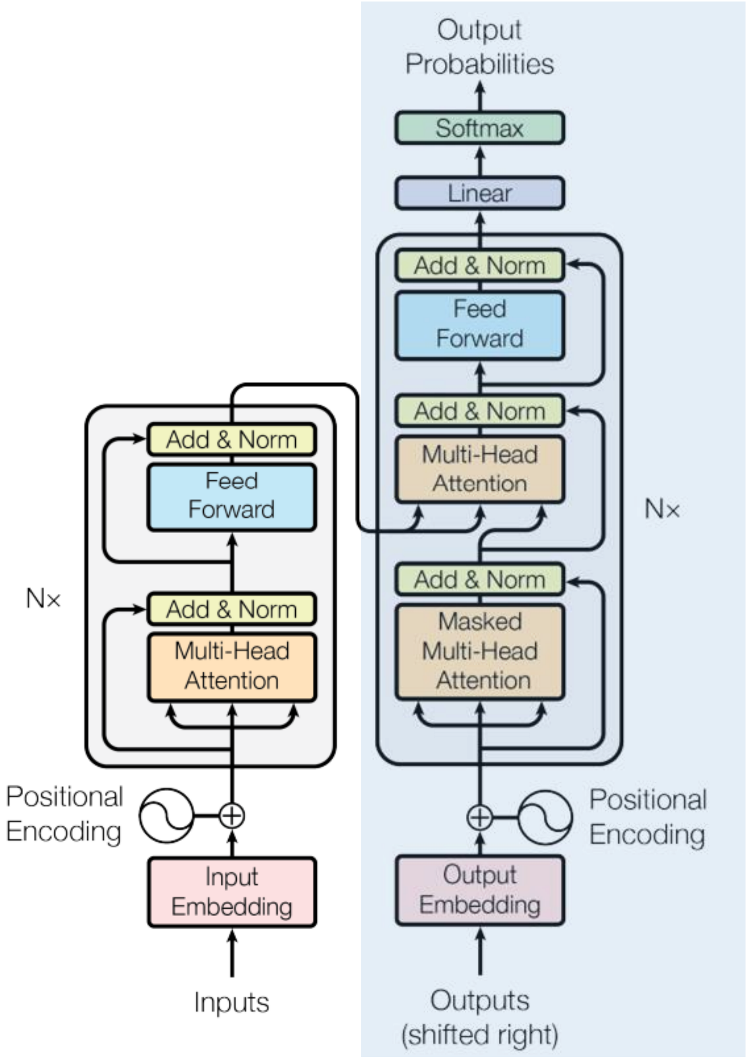
Encoder–Decoder (Machine Translation)

- An encoder model turns the source sequence into an intermediate representation.
- A decoder model is trained to predict the next token in the target sequence by looking at both previous tokens and the encoded source sequence.
- During inference, we don't have access to the target sequence (predict it from scratch) → must generate it one token at a time



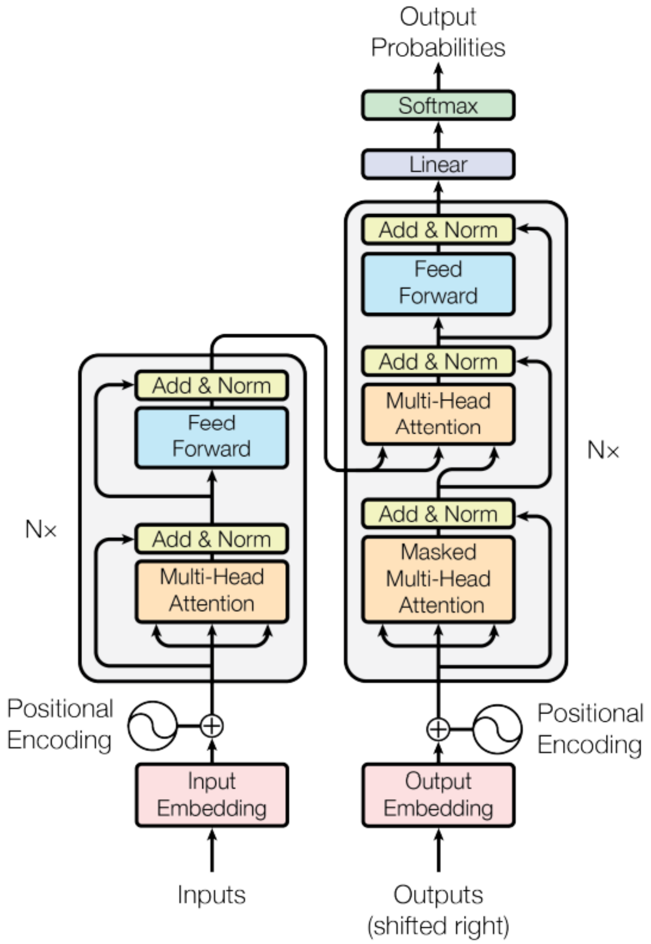
Self-Attention and Transformers

- Transformers primarily use self-attention mechanisms.
- Both the encoder and decoder layers apply self-attention to capture relationships and dependencies within the input sequence itself.
- By using self-attention, transformers can:
  1. Process inputs in parallel
  2. Identify long-range dependencies
  3. Model complex relationships more efficiently compared to RNN and CNN.



Module 7 – Part 2

Transformers Architecture

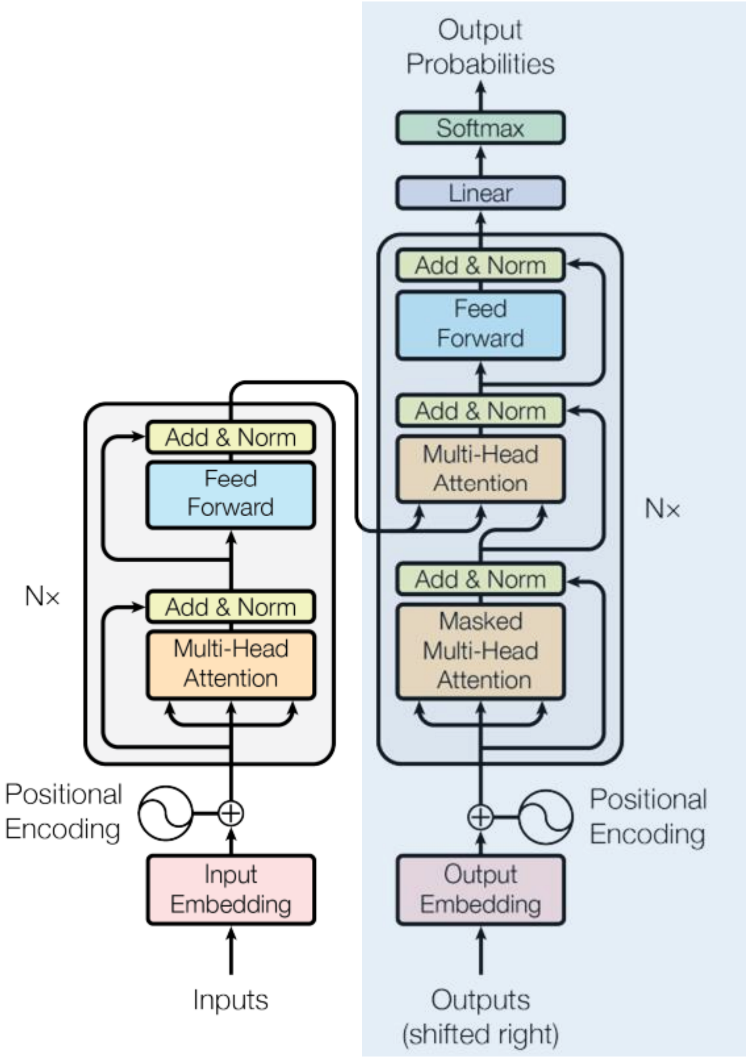


Jon M. Huntsman School of Business
Utah State University

Pedram Jahangiry

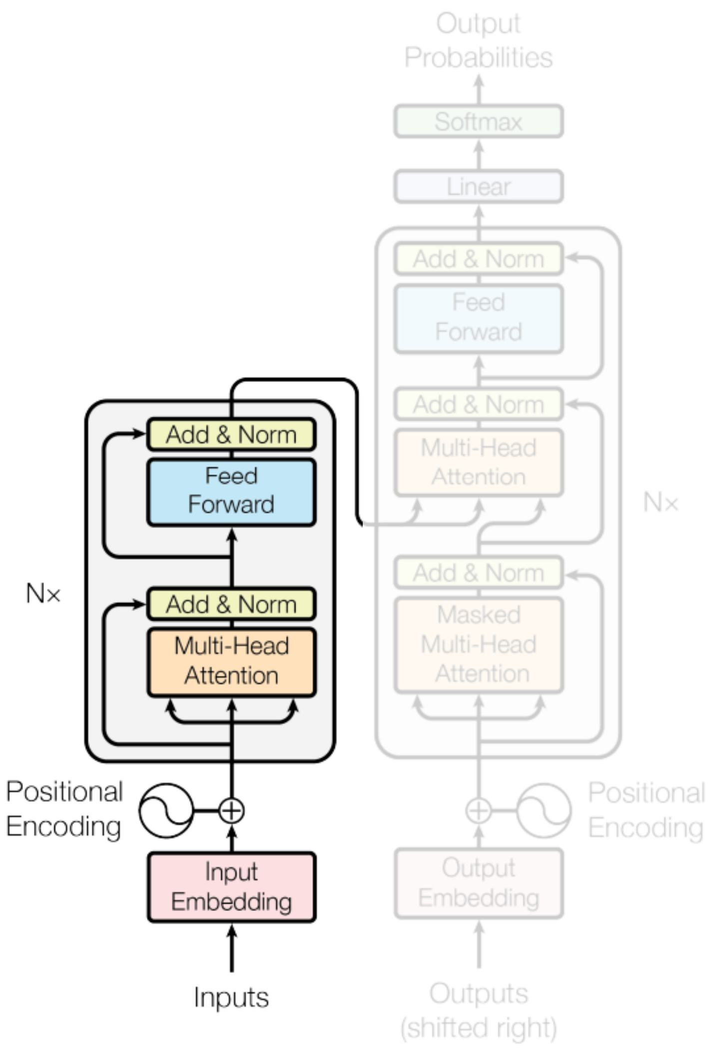
Transformers outline

- Encoder – Decoder architecture
  - Embedding and Positional Encoding
  - Self-Attention (scaled dot product attention)
  - Query-Key-Value model
  - (Masked) Multi-head attention
  - Encoder-Decoder attention
  - Residual connections
  - Layer normalization
  - Feed Forward
  - Softmax layer



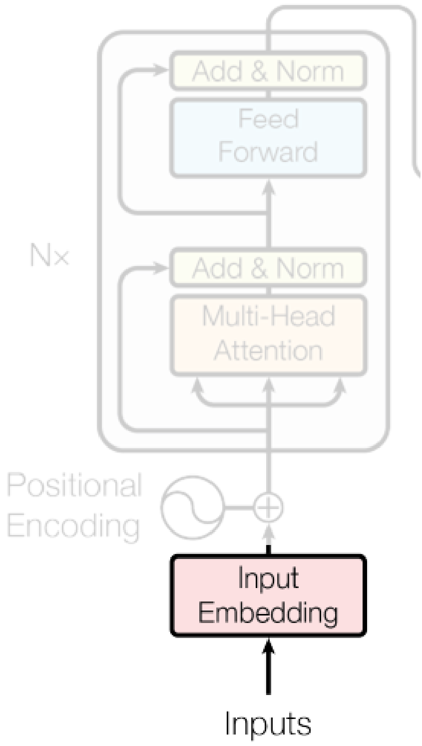
Transformer architecture

Encoder



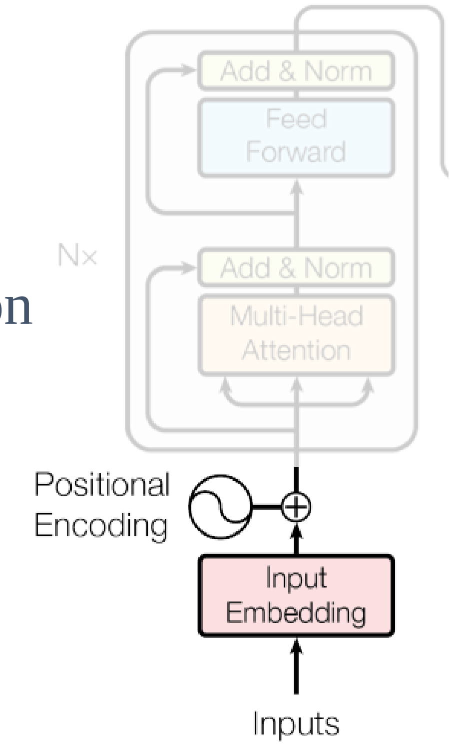
Input Embedding

- Word embedding is a numerical representation of words as vectors in a continuous, relatively low-dimensional space compared to the size of the vocabulary, which captures their semantic meaning and relationships with other words.
- Word embedding is learned through a neural network
- Notion of order is lost!
- Example: “ I love Transformers!” with embedding dim=10
  - "I" : [-0.2, 0.1, 0.3, -0.4, 0.5, -0.6, -0.1, -0.3, -0.2, 0.4]
  - "love“ : [0.3, -0.2, 0.1, 0.5, -0.4, 0.2, -0.6, -0.1, -0.3, 0.2]
  - "transformers“ : [-0.4, -0.3, 0.6, -0.1, 0.2, 0.1, 0.4, 0.5, 0.2, 0.3]
  - "!“ : [0.1, -0.4, -0.2, 0.3, 0.2, -0.1, 0.5, -0.3, -0.5, -0.4]



Positional Encoding

- Positional encoding reinject order information
- positional encoding is a set of small constants, which are added to the word embedding vector before the first self-attention layer.
- If the same word appears in a different position, the actual representation will be slightly different, depending on where it appears in the input sentence
- The model will figure out how to leverage this additional information.
- Naïve solution: My name is Pedram → {0,1,2,3}
- Better solution: Add a circular wiggle



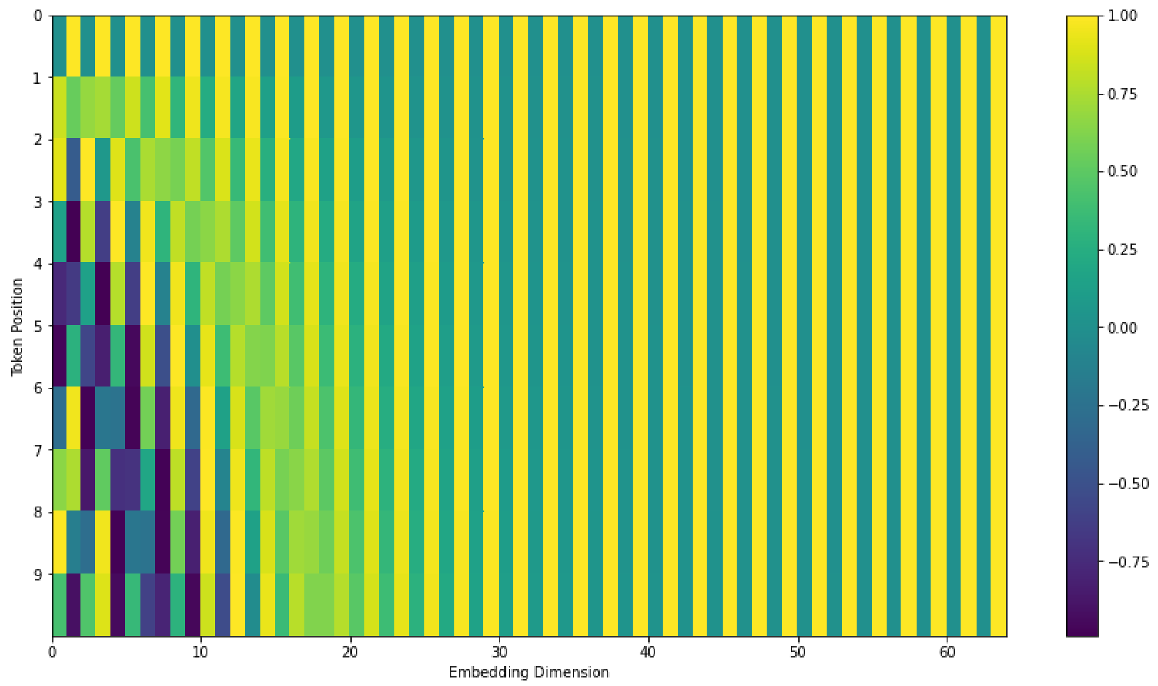
Word embedding + positional encoding = positional embedding

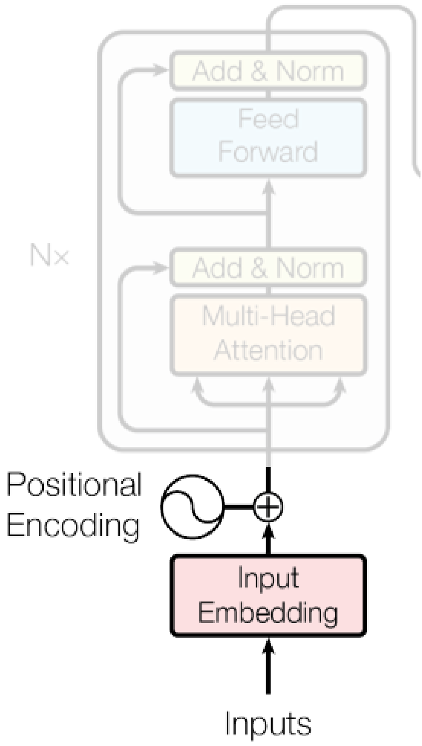
Positional Encoding

- Better solution: Sine – Cosine function

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

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

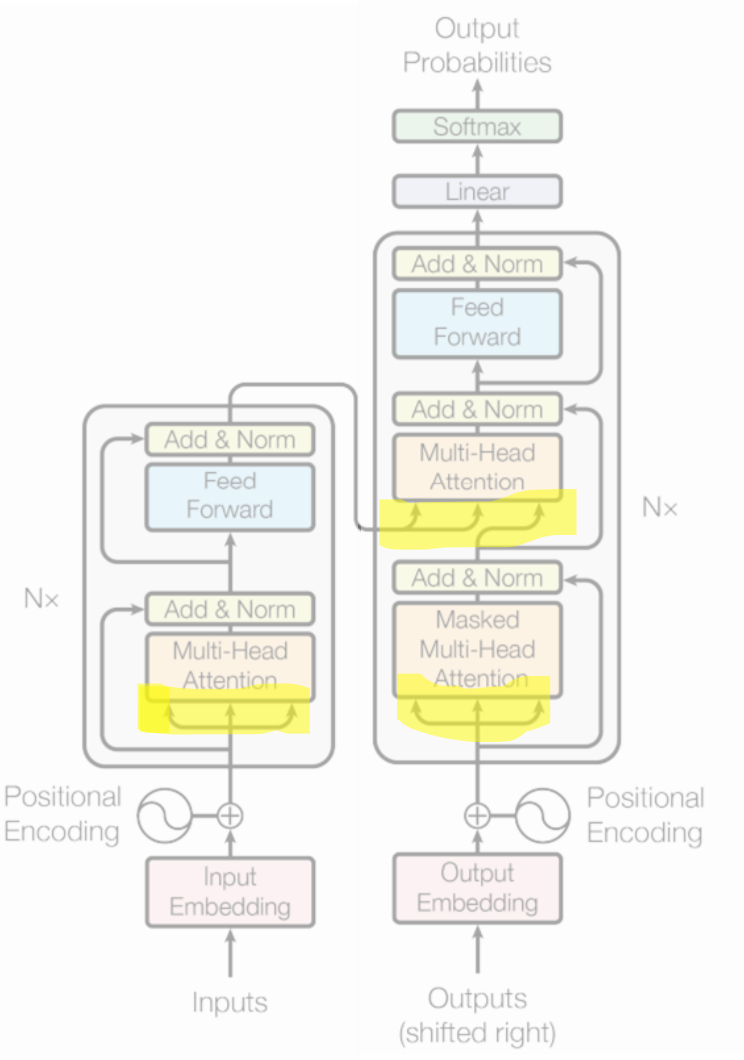




Jon M. Huntsman School of Business Utah State University

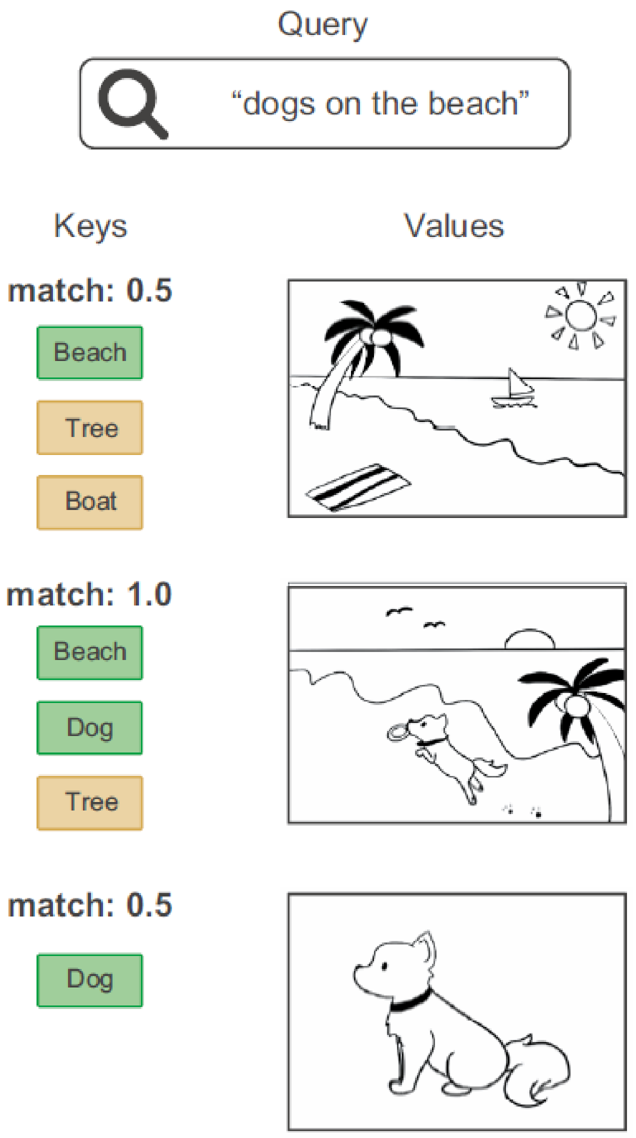
Pedram Jahangiry

Self-Attention mechanism
Query-Key-Value



The Query-Key-Value

- This terminology comes from search engines
- Query: What we are looking for?
- Value: Body of knowledge that we are trying to extract information from (database)
- Key: Set of "keywords" that describes the value in a format that can be readily compared to a query.
- The "query" is compared to a set of "keys," and the match scores are used to rank "values" (images in this example).

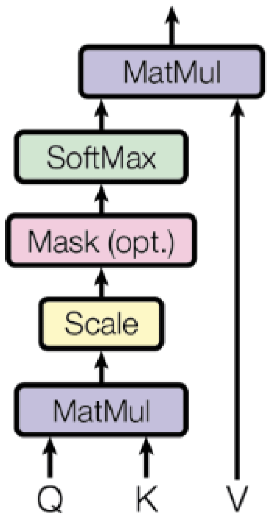


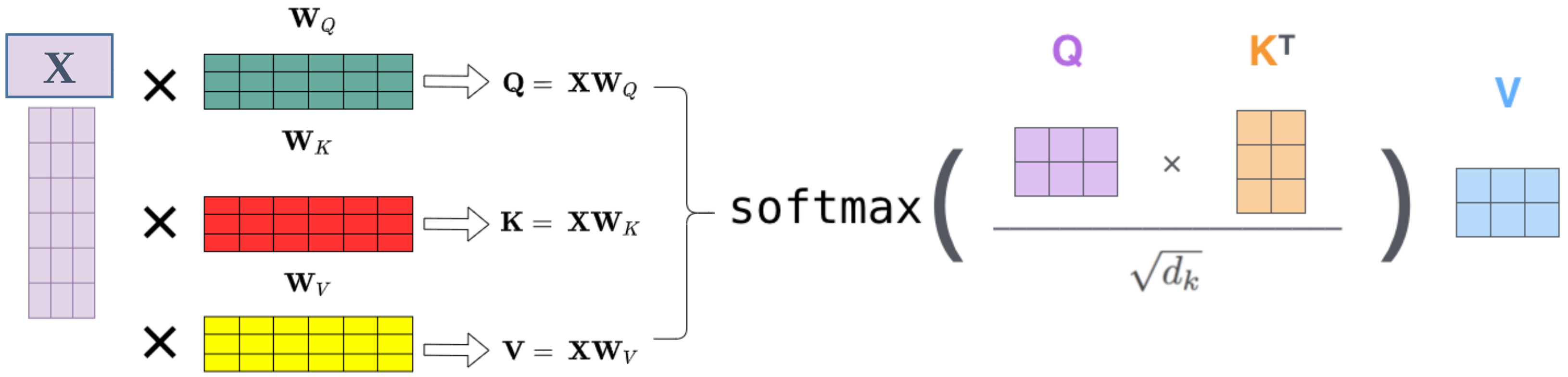
Jon M. Huntsman School of Business
Utah State University

Pedram Jahangiry

Scaled dot product attention

- Goal: Identify which part to attend to and Extracting features with high attention
- Scaled dot product attention utilizes three weight matrices, referred to as W_Q, W_K, and W_V which are learned as model parameters during training.
- These matrices serve to project the inputs into query, key, and value components of the sequence.

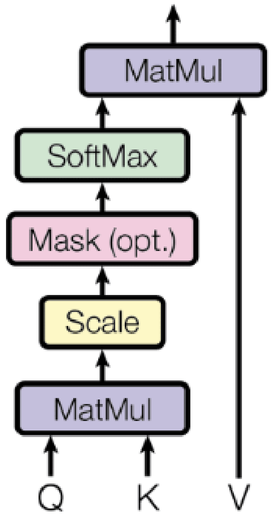




JON M. HUNTSMAN SCHOOL OF BUSINESS
UtahStateUniversity

Pedram Jahangiry

Self-Attention in Language Models

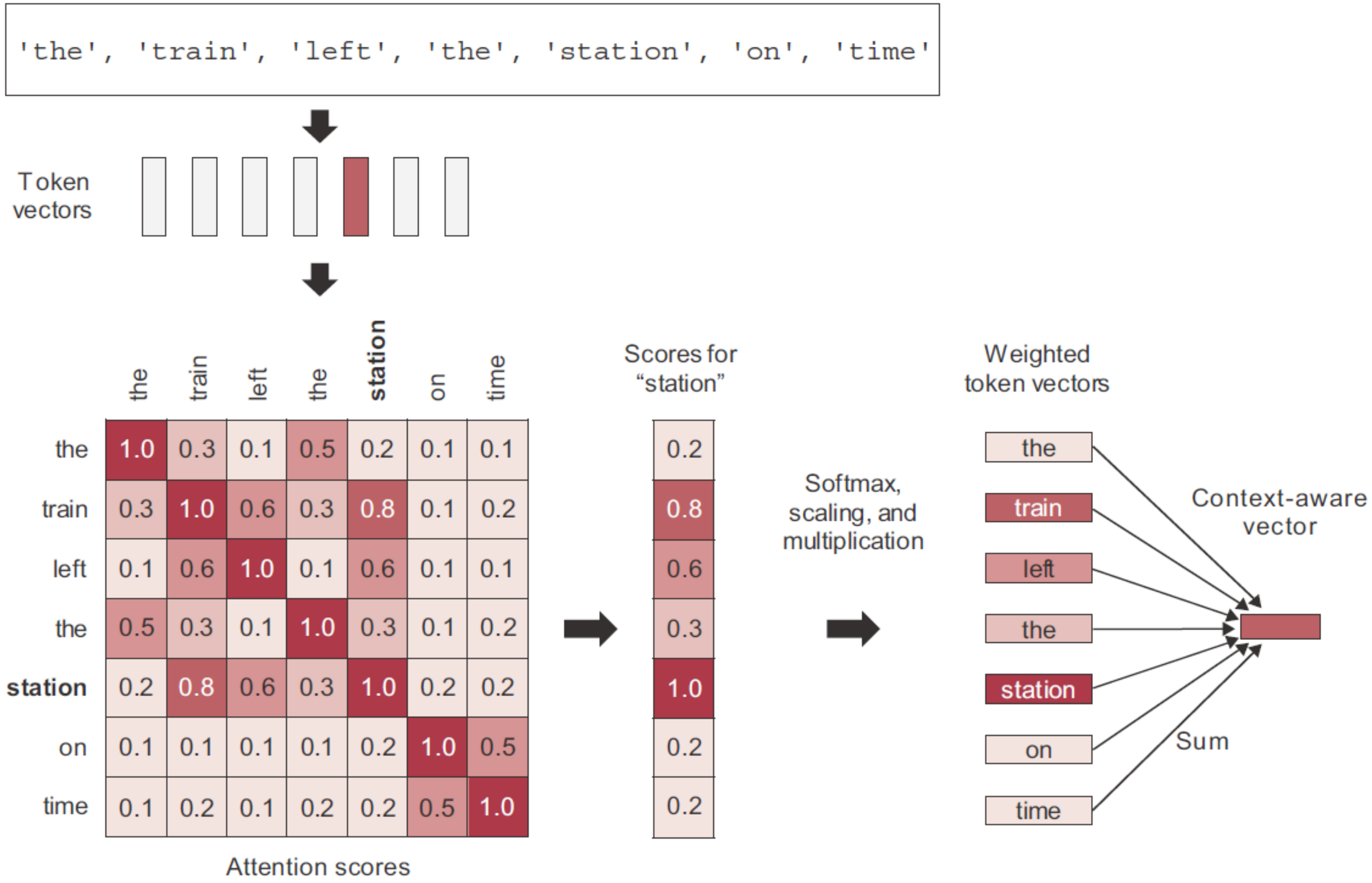


1. The input embeddings are transformed into query, key, and value vectors using different learned weight matrices. Each of these vectors is used for different purposes:
   - Query (Q): Represents the current token that the model is focusing on.
   - Key (K): Represents all the tokens in the input sequence, which are compared to the query to compute attention scores.
   - Value (V): Represents all the tokens in the input sequence, which are weighted by the attention scores to create the context vector.
2. Attention scores = $softmax(\frac{Q.K^T}{\sqrt{d_k}})$
3. The attention scores are then used to weight the value vectors, and the weighted sum of these value vectors forms the context vector for the current word.
4. This context vector is then used to generate the output.

Jon M. Huntsman School of Business Utah State University

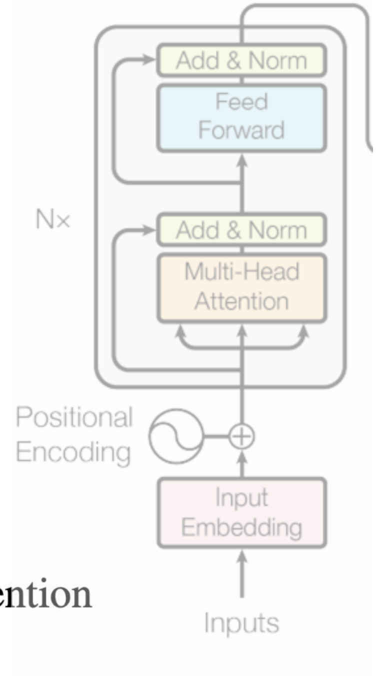
Pedram Jahangiry

Self-Attention in Language Models

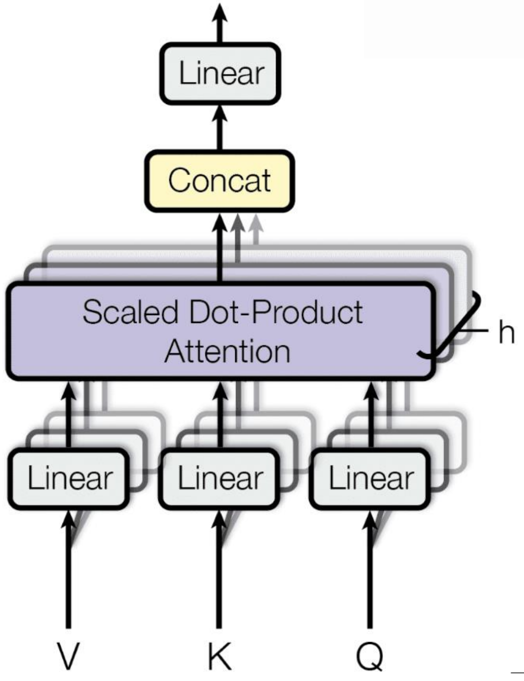


Multi-Head Attention

- Intuition: it allows the model to attend to different parts of the sequence differently each time
- Features within a self-attention head are correlated but mostly independent from those in other heads, similar to Depthwise separable convolutions treating each channel independently to obtain more expressive representations.
- Multi-head attention is simply the application of the same idea to self-attention.
- Independent heads help the layer learn different groups of features for each token.



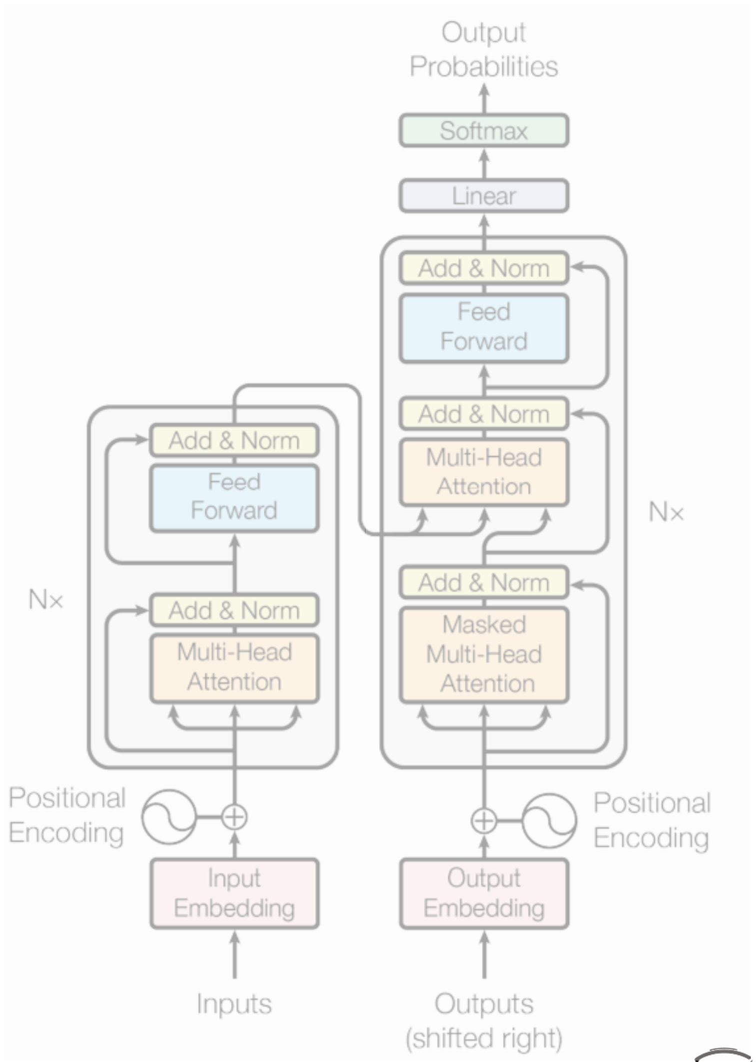
Multi-Head Attention



Add & Norm

- Residual connections are added to preserve valuable information and prevent information loss during the training process.
- Normalization layers aid gradient flow during backpropagation to improve the training process.
  - Layer Normalization instead of Batch Normalization
  - Normalizing each sequence independently

- Leveraging standard architectural patterns such as factoring outputs into multiple independent spaces, adding residual connections, and incorporating normalization layers can enhance the performance of complex models.

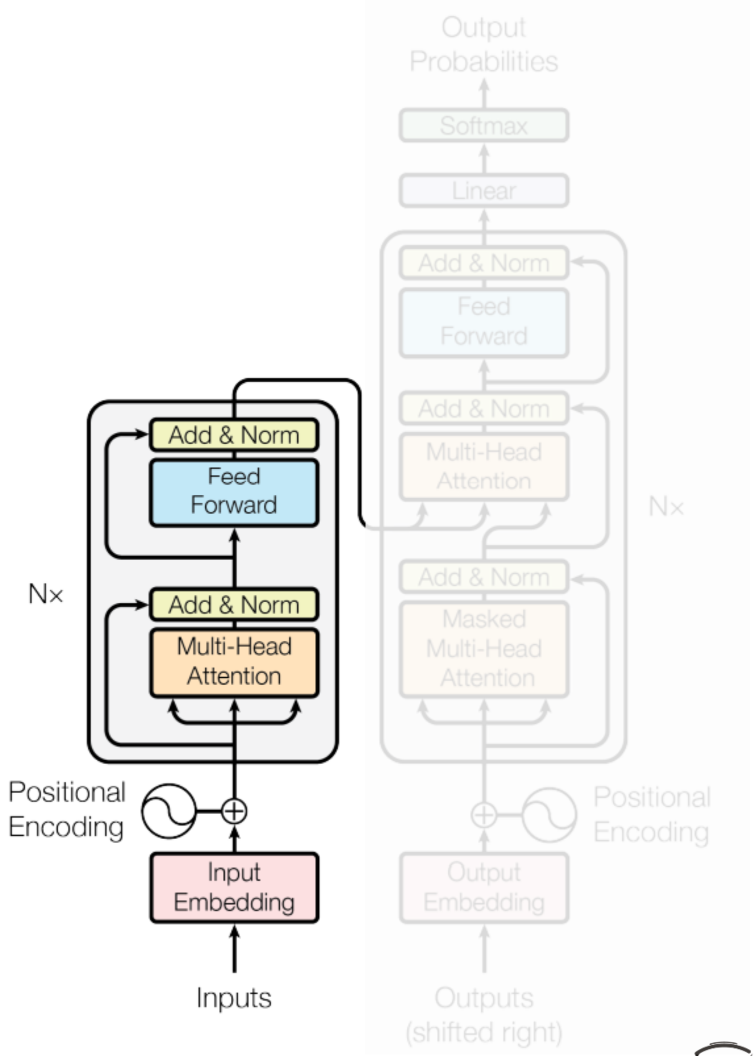


Jon M. Huntsman School of Business Utah State University

Pedram Jahangiry

Encoder summary

- Input embedding
- Positional encoding
- Encoder block:
  - Multi-head self-attention layer
  - Layer normalization
  - Residual connection
  - MLP (2 linear layers + RELU activation)
  - Second Layer normalization
  - Second residual connection
- Replicating N times (N=6 in the original paper)

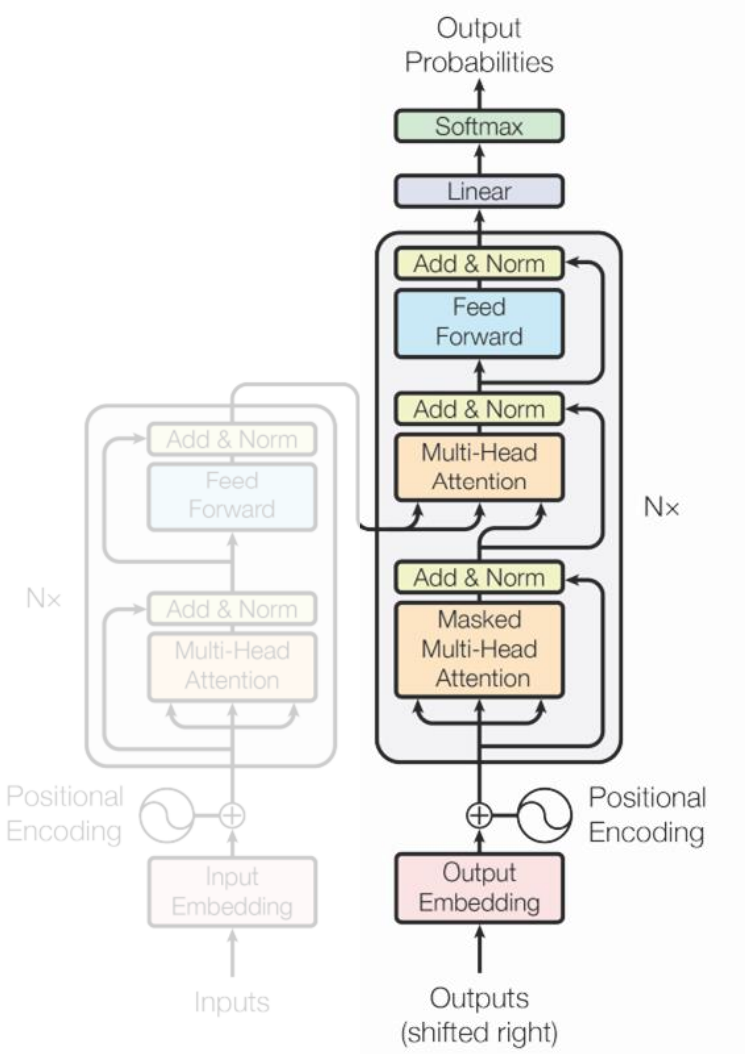


JON M. HUNTSMAN SCHOOL OF BUSINESS
UtahStateUniversity

Pedram Jahangiry

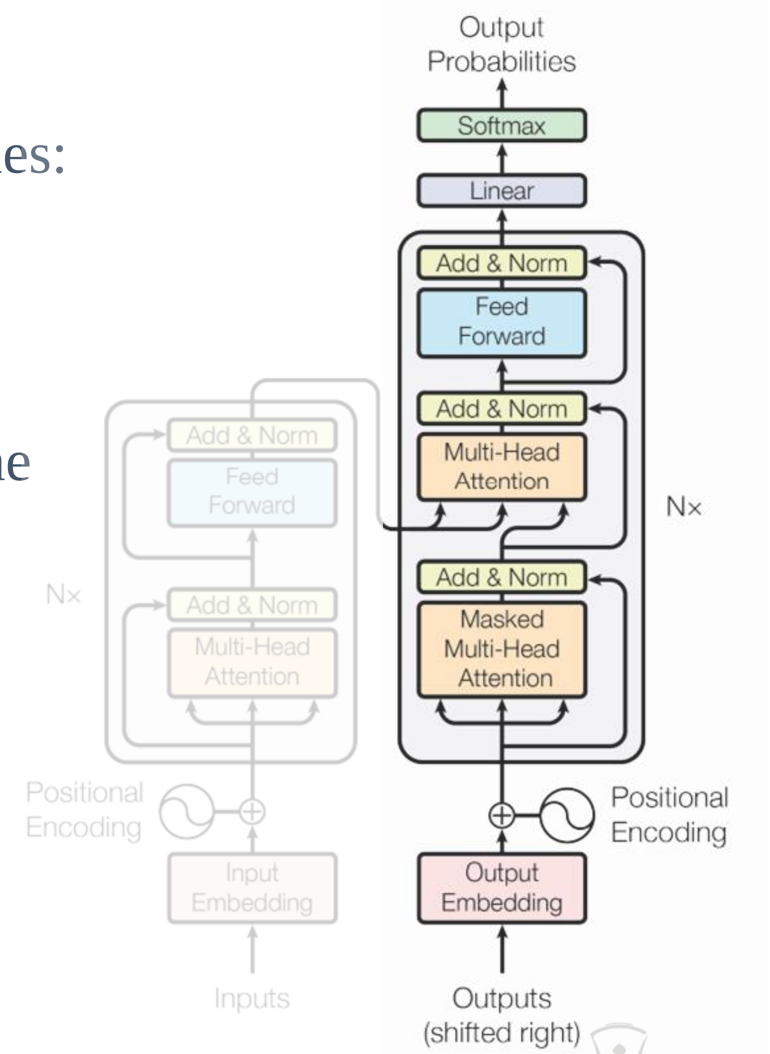
Transformer architecture

Decoder



Decoder

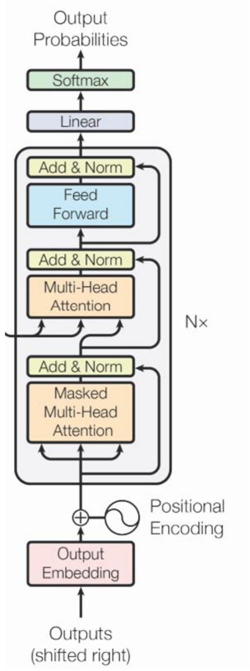
- Decoder consist all the components of encoder plus two novel ones:
  - Masked multi-head self-attention layer
  - Encoder-Decoder Attention layer
- Final decoder block output is passed through a linear layer and the probabilities are calculated with a standard softmax function
- Decoder is autoregressive:
  - Generate output one at a time
  - This is repeated until the **<end>** token is seen.



JON M. HUNTSMAN SCHOOL OF BUSINESS UtahStateUniversity

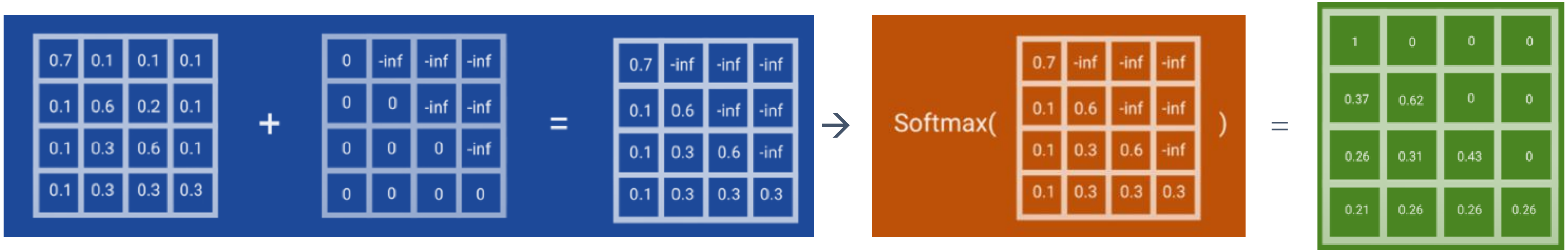
Pedram Jahangiry

Masked multi-head self-attention layer



- First Multi-headed attention computes the attention scores for the decoders input.
- For this Multi-headed attention layer, we need to apply masking to avoid cheating.

$$\text{MaskedAttention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}\left(\boxed{\frac{\mathbf{Q}\mathbf{K}^T + \mathbf{M}}{\sqrt{d_k}}}\right)\mathbf{V}$$

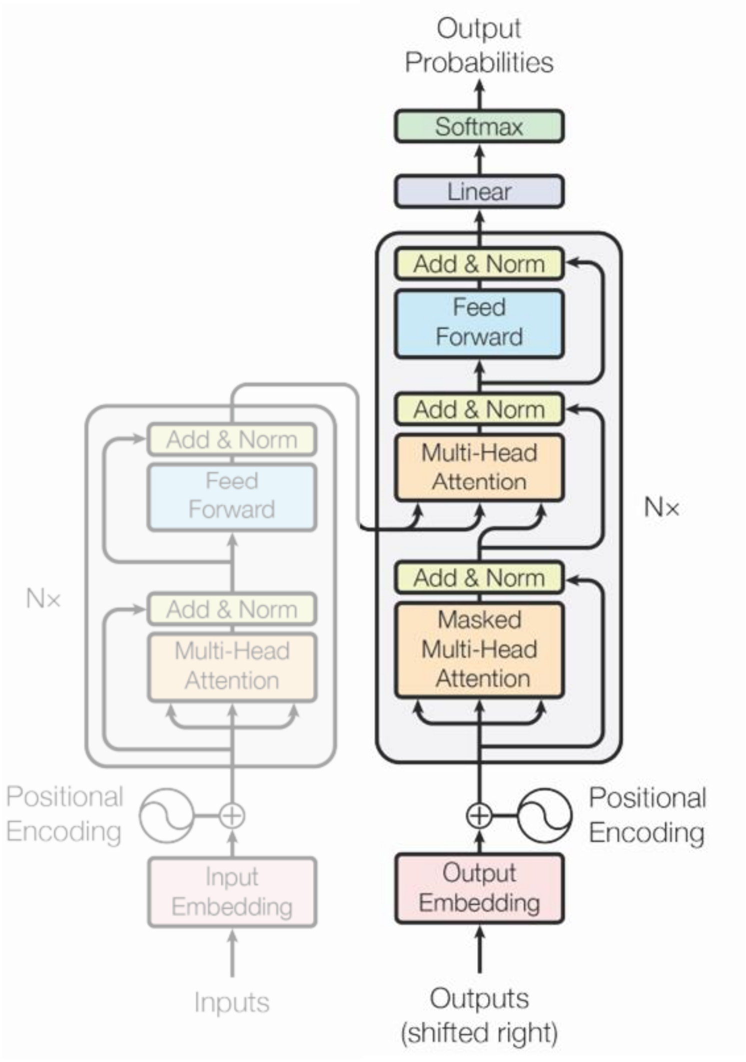


Jon M. Huntsman School of Business
UtahStateUniversity

Pedram Jahangiry

Encoder-Decoder Attention Layer

- Transformers also use a form of attention known as "encoder-decoder attention" in the decoder layers.
- This is where the magic happens, where the decoder processes the encoded representation (K,V)
- This attention mechanism allows the decoder to focus on specific parts of the input sequence encoded by the encoder when generating each output element.
- The encoder-decoder attention mechanism helps the model to align the input and output sequences better, which is particularly useful in tasks like machine translation.

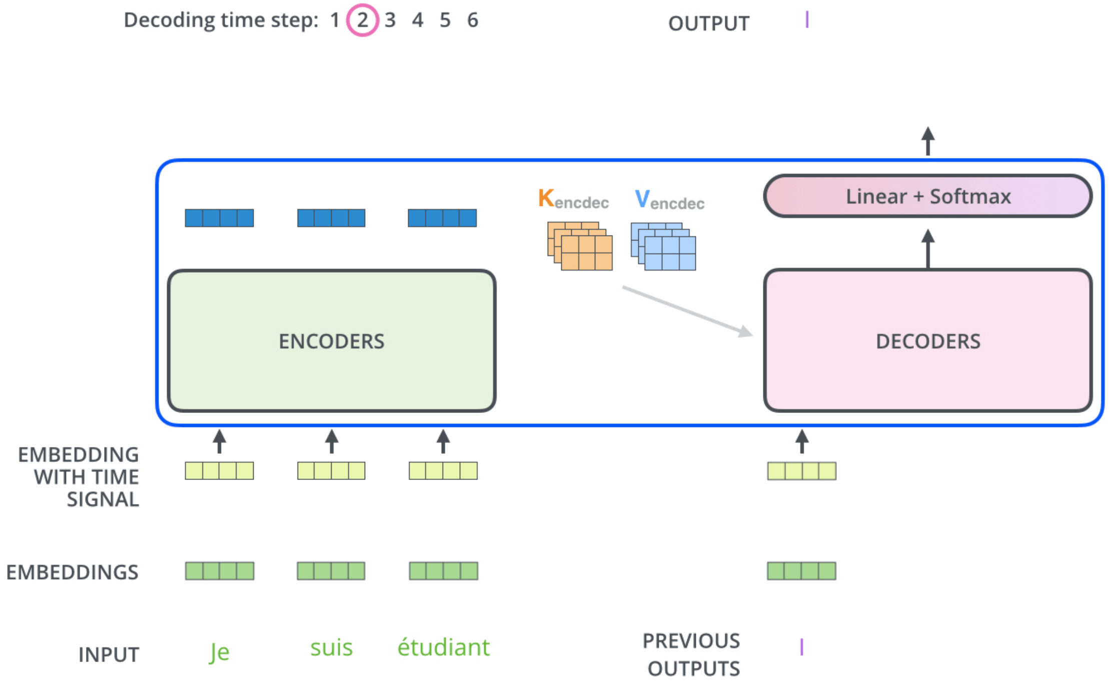


JON M. HUNTSMAN SCHOOL OF BUSINESS
UtahStateUniversity

Pedram Jahangiry

Encoder-Decoder Attention Layer (cross-attention)

- The encoder output is utilized to generate the Key and Value matrices.
- On the other hand, the Masked Multi-head attention block's output contains the newly generated sentence, represented as the Query matrix in the attention layer.
- This process matched the encoders input to the decoders input allowing the decoder to decide which encoder input is relevant to focus on.



Training

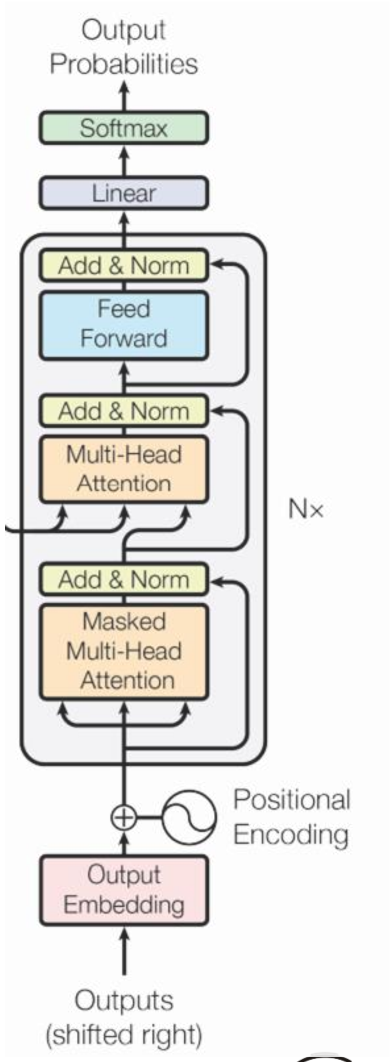
- Like any other deep learning model, training involves:
  - Feed forward, loss computation, backpropagation, parameter updates
- Cross-entropy compare two probability distributions.

Target Model Outputs

Output Vocabulary:	a	am	I	thanks	student	<eos>
position #1	0.0	0.0	1.0	0.0	0.0	0.0
position #2	0.0	1.0	0.0	0.0	0.0	0.0
position #3	1.0	0.0	0.0	0.0	0.0	0.0
position #4	0.0	0.0	0.0	0.0	1.0	0.0
position #5	0.0	0.0	0.0	0.0	0.0	1.0

Trained Model Outputs

Output Vocabulary:	a	am	I	thanks	student	<eos>
position #1	0.01	0.02	0.93	0.01	0.03	0.01
position #2	0.01	0.8	0.1	0.05	0.01	0.03
position #3	0.99	0.001	0.001	0.001	0.002	0.001
position #4	0.001	0.002	0.001	0.02	0.94	0.01
position #5	0.01	0.01	0.001	0.001	0.001	0.98

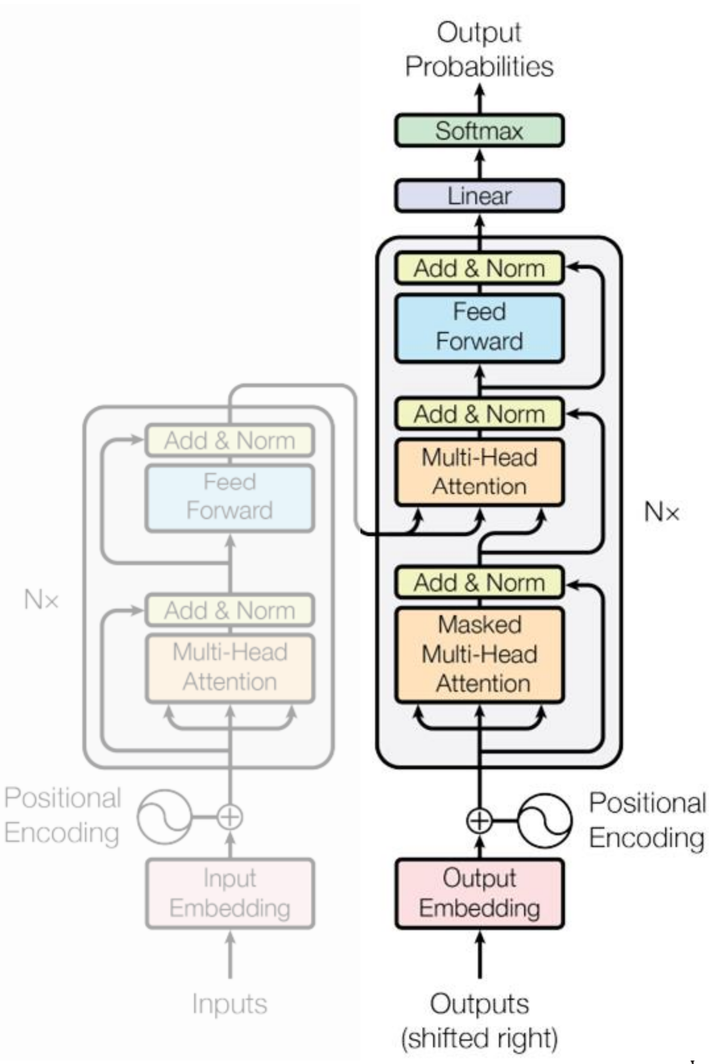


Jon M. Huntsman School of Business
UtahStateUniversity

Pedram Jahangiry

Decoder summary

- Input embedding
- Positional encoding
- Decoder block:
  - Masked Multi-head self-attention layer
  - First Normalization and residual connection
  - Encoder-Decoder Multi-head attention
  - Second Normalization and residual connection
  - MLP (2 linear layers + RELU activation)
  - Third Normalization and residual connection
- Linear layer followed by a softmax function
- Replicating N times (N=6 in the original paper)

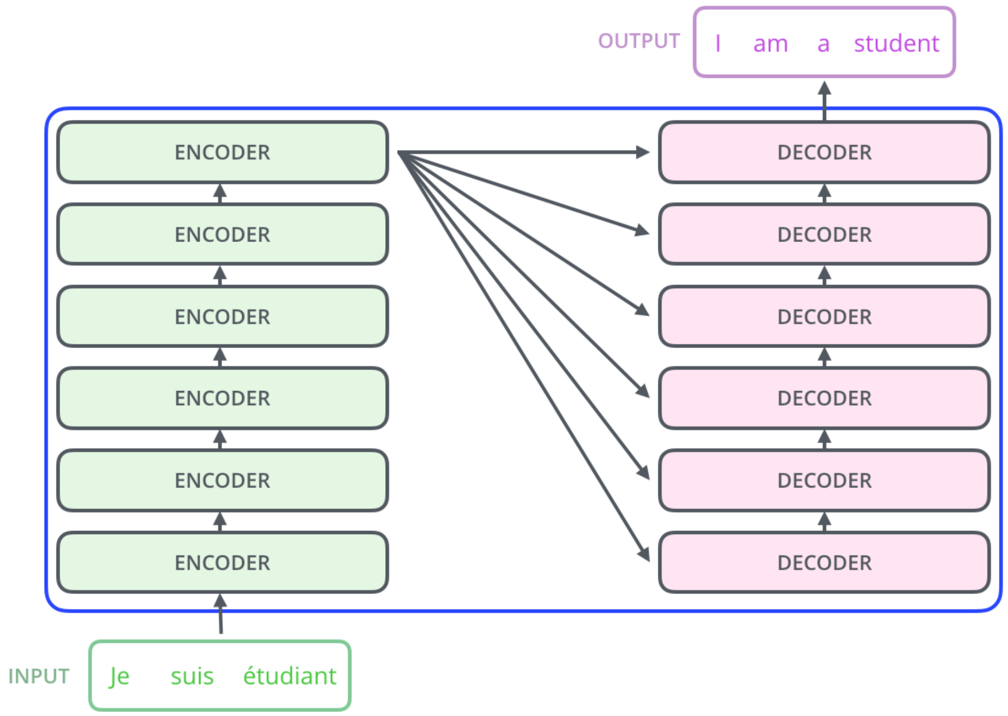


JON M. HUNTSMAN SCHOOL OF BUSINESS
UtahStateUniversity

Pedram Jahangiry

Why transformers work?

- Multi-head attention (multiple representations of the same input)
- Context awareness through self-attention (data-dependent dynamic weights)
- Stacking multiple encoder and decoder blocks (transformer blocks are shape-invariant)



References

- Attention is all you need! *Vaswani et al*
- Deep learning with Python, *Francois Chollet*
- How Transformers work in deep learning and NLP: an intuitive introduction, *Nikolas Adaloglou*
- Transformers from scratch, *Brandon Rohrer*
- The illustrated transformers, *Jay Alammar*

Road map!

- ✓ Module 1- Introduction to Deep Learning
- ✓ Module 2- Setting up Deep Learning Environment
- ✓ Module 3- Machine Learning review (ML fundamentals + models)
- ✓ Module 4- Deep Neural Networks (NN and DNN)
- ✓ Module 5- Deep Computer Vision (CNN, R-CNN, YOLO, FCN)
- ✓ Module 6- Deep Sequence Modeling (RNN, LSTM)
- ✓ Module 7- Transformers (Attention is all you need!)
- Module 8- Deep Generative Modeling (AE, VAE, GAN)
- Module 9- Deep Reinforcement Learning (DQN, PG)

Module 7 – Transformers
Modern Architectures