COPPE
Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia
UFRJ

UFRJ
UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

Attention is All You Need

Vaswani A. et al., NeurIPS 2017

TRANSFORMERS

Breno T. Tostes

16 de maio de 2025

Introdução

Modelos passados

Até então usava-se modelos baseados em LSTMs e CNNs para a área de NLP, cada um com suas vantagens.

- Seq2seq Learning with Neural Networks; I. Sutskever, NeurIPS 2014
- Learning Phrase Representations using RNN Encoder–Decoder for SMT; K. Cho e Y. Bengio, EMNLP 2014
- NMT by Jointly Learning to Align and Translate; K. Cho e Y. Bengio, ICLR 2015
- Outros

Word Embedding

Representação de palavras através do **contexto** em que aparecem: determina um vetor *denso* que será bom em prever as palavras que surgem ao seu entorno a partir de vetores do tipo **one-hot**. Ex.:

$$\text{linguística} = \left[-0.976, 0.286, \cdots, -0.438 \right]$$

Word2Vec: Um dos principais modelos de NLP para geração de embeddings a partir de Redes Neurais.

- Algoritmos : Skip-grams (SG) ou Continuous Bag of Words (CBOW)
- Métodos de treino: Hierarchical softmax e Negative Sampling

Exemplo

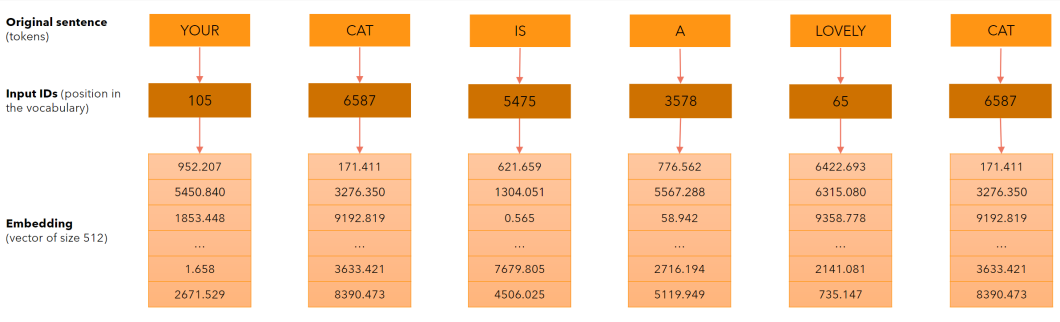


Figura 1: Representação intuitiva de Word Embedding

Intuição Visual

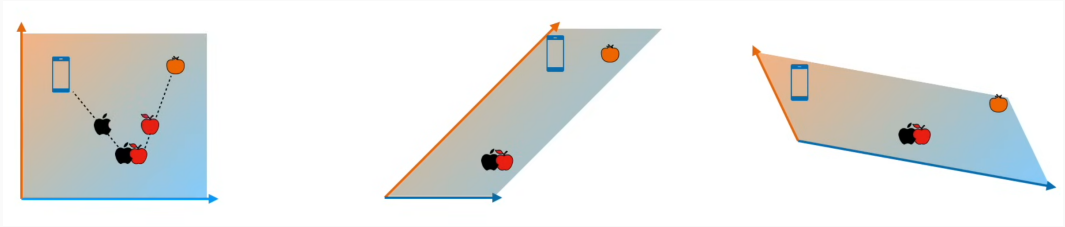


Figura 2: Visualização de embeddings bons e ruins para separação simbólica

Modelo de Transformer

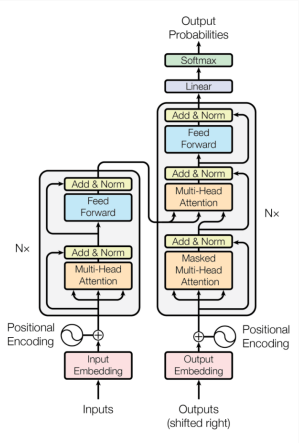
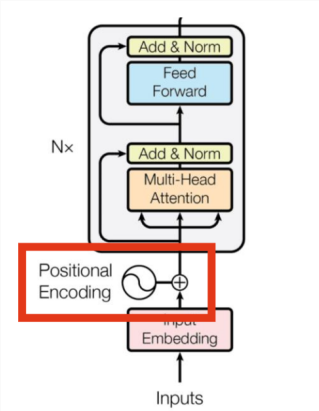


Figura 3: Modelo proposto: Encoder-Decoder com Transformer

Positional Encoding

Positional Encoding



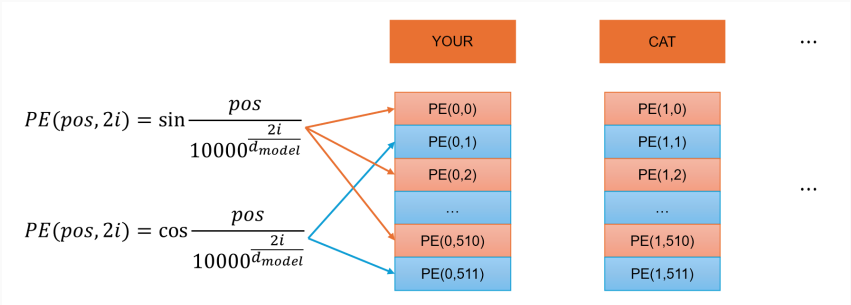
Positional Encoding

Até agora fizemos cada Token carregar o significado de palavras isoladas de um contexto de uso, isto é, como se tivéssemos representações de dicionário.

- Gostaríamos que tokens próximos dentro de frases tivessem embeddings que carregassem essa informação.
- Ao mesmo tempo, palavras distantes devem parecer distantes em seus valores.
- O Positional Encoding deve formar um padrão aprendível pelo modelo.

Como funciona?

Positional Encoding será uma função calculada **apenas uma vez** por posição. As senóides garantem que não ocorrerá repetição, uma vez que alterna a amplitude do sinal conforme a feature!



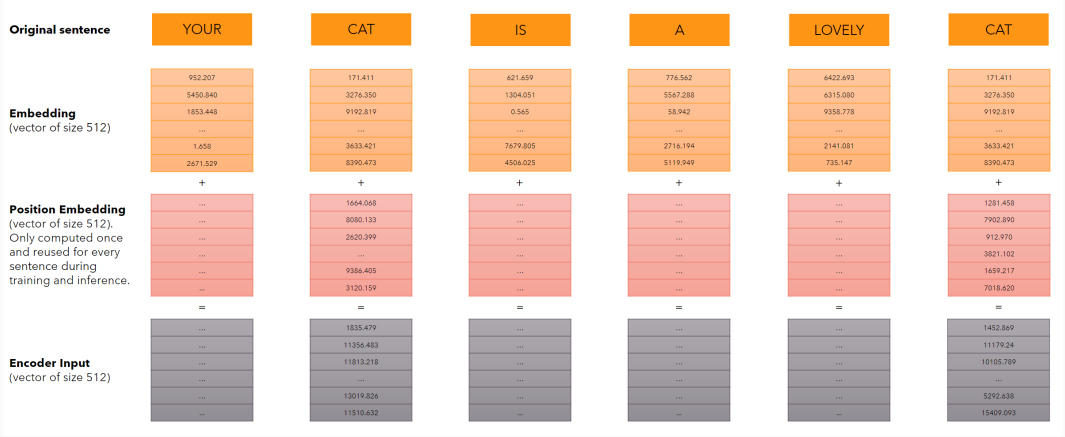
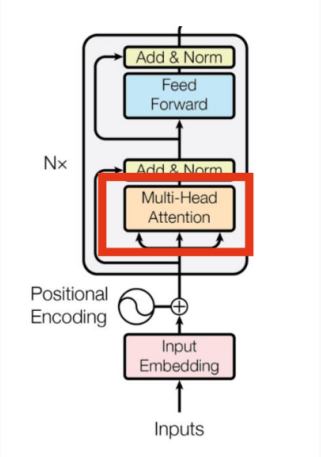


Figura 4: Retomando o exemplo anterior. O input do modelo será o agregado de informações.

Self Attention


Add & Norm
Feed Forward
Nx
Add & Norm
Multi-Head Attention
Positional Encoding
Input Embedding
Inputs

Atenção

- O mecanismo de Atenção surge do contexto de visão computacional

- Podemos pensar em Atenção como uma analogia para a operação de SELECT em um banco de dados:
  - Desejamos resgatar um valor (**value**) através de uma **query** para uma chave (**key**)

$$\mathsf{a}(q, k, v) = \sum_i \text{similarity}(q, k_i) \times v_i$$

Self - Attention

Partindo da entrada obtida nas etapas prévias, deseja-se aplicar o mecanismo de Atenção a uma matriz $\mathbf{X} \in \mathbb{R}^{N \times d_{model}}$ em que d_{model} é a dimensionalidade das features e N a quantidade de tokens usados.

Pergunta: Como derivar valores de **Queries, Keys** e **Values** a partir de X?

Queries, Keys e Values

Resposta: com transformações afins, isto é, **redes neurais de 1 camada**.

Deixando que o modelo aprenda as matrizes W_q, W_k e $W_v \in \mathbb{R}^{d_{model} \times d_{model}}$ para as transformações de X em Q, K e $V \in \mathbb{R}^{N \times d_{model}}$, de forma que:

$$Q = W_q X$$

$$K = W_k X$$

$$V = W_v X$$

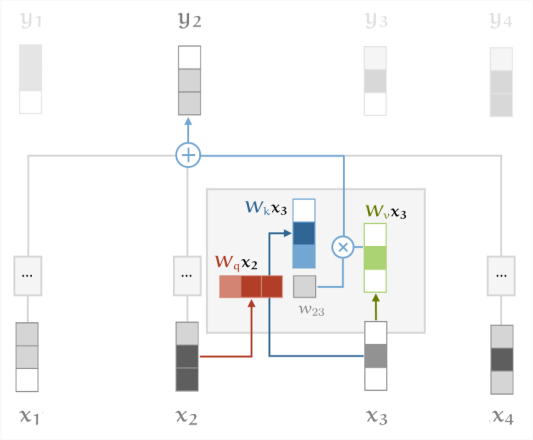


Figura 5: Com um certo grau de spoilers, pode-se observar a operação a nível atômico.

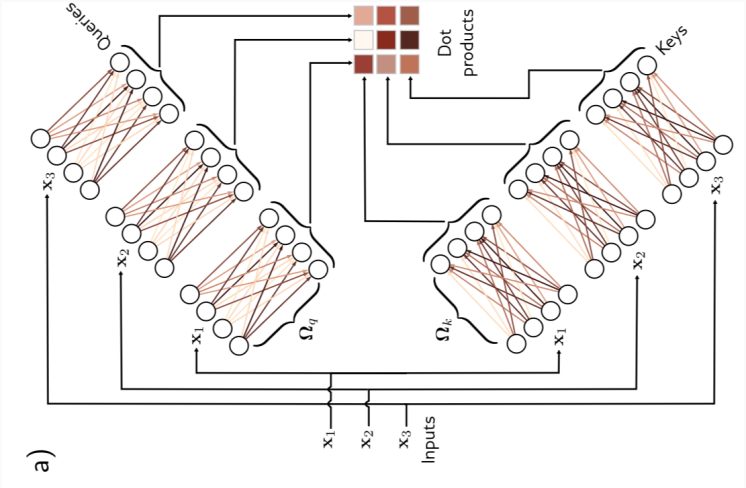


Figura 6: Outra visualização para a mesma operação. $\Omega = W$.

Extraindo Atenção

Como indicado pelas figuras 5 e 6 e retomando a analogia original, ao realizarmos uma operação de multiplicação de matrizes:

$QK^\top \in \mathbb{R}^{N \times N}$, o resultado obtido é uma matriz de compatibilidade entre cada Token da entrada

	YOUR	CAT	IS	A	LOVELY	CAT
YOUR	0.268	0.119	0.134	0.148	0.179	0.152
CAT	0.124	0.278	0.201	0.128	0.154	0.115
IS	0.147	0.132	0.262	0.097	0.218	0.145
A	0.210	0.128	0.206	0.212	0.119	0.125
LOVELY	0.146	0.158	0.152	0.143	0.227	0.174
CAT	0.195	0.114	0.203	0.103	0.157	0.229

Extraindo Atenção II

- Ao tratar cada palavra como uma pergunta (**query**) para as outras palavras (**keys**), foi obtida uma medida de *compatibilidade* entre as palavras da sentença.

- Desejamos realizar uma **combinação convexa** entre valores (**values**) utilizando as correlações como fator ponderativo.

Extraindo Atenção III

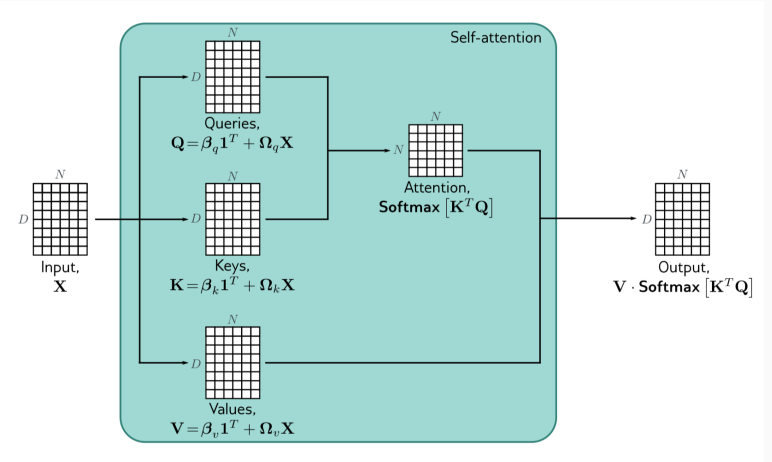
- A fim de enfatizar os vetores de maior semelhança, transformamos a matriz de compatibilidade em distribuições através de uma *softmax*

$$\textbf{Attention} = Softmax(QK^\top) = A \in \mathbb{R}^{N \times N}$$

- Finalmente, aplicamos $V \in \mathbb{R}^{N \times d_{model}}$ como uma última transformação que incorpora a informação de forma útil ao modelo:

$$\textbf{Self-Attention}(Q, K, V) = Softmax(QK^\top) \cdot V$$

Overview


Self-attention
N
D
Queries,
$\mathbf{Q} = \boldsymbol{\beta}_q \mathbf{1}^T + \boldsymbol{\Omega}_q \mathbf{X}$
Keys,
$\mathbf{K} = \boldsymbol{\beta}_k \mathbf{1}^T + \boldsymbol{\Omega}_k \mathbf{X}$
Values,
$\mathbf{V} = \boldsymbol{\beta}_v \mathbf{1}^T + \boldsymbol{\Omega}_v \mathbf{X}$
Attention,
Softmax $\left[\mathbf{K}^T \mathbf{Q}\right]$
Input,
$\mathbf{X}$
Output,
$\mathbf{V} \cdot$ Softmax $\left[\mathbf{K}^T \mathbf{Q}\right]$

Exemplo

- Input: *Pizza é bom*
- Tamanho de seq.: $N = 3$
- Dimensão de Embedding: $d_{model} = 2$

$$\text{Input Embedding} = \begin{bmatrix} 0.3923 & -0.2236 \\ -0.3195 & -1.2050 \\ 1.443 & -0.6332 \end{bmatrix}$$

$$\text{PE} = \begin{bmatrix} 0.3901 & 0.9088 \\ 0.5334 & 0.7073 \\ 0.7116 & 0.2050 \end{bmatrix}$$

Exemplo

Após aplicarmos o Positional Encoding, obtemos $X \in \mathbb{R}^{N \times d_{model}}$:

$$X = \mathrm{IE} + \mathrm{PE} = \begin{bmatrix} 0.7824 & 0.6852 \\ 0.2189 & -0.4977 \\ 1.7561 & -0.4282 \end{bmatrix}$$

Utilizando X e calculando os valores de W_Q, W_K e W_V, obtemos $Q, K, V \in \mathbb{R}^{N \times d_{model}}$:

$$\mathrm{Q} = \begin{bmatrix} 0.7197 & 0.3081 \\ 0.3592 & 0.2259 \\ 0.3430 & 0.3670 \end{bmatrix} \quad \mathrm{K} = \begin{bmatrix} 0.7133 & 0.6944 \\ 0.3993 & 0.7455 \\ 0.7490 & 0.5221 \end{bmatrix} \quad \mathrm{V} = \begin{bmatrix} 0.5530 & 0.5382 \\ 0.7668 & 0.8359 \\ 0.8591 & 0.7898 \end{bmatrix}$$

Exemplo

Em seguida, calcula-se a matriz de compatibilidade $QK^\top$ e aplicamos a Softmax:

$$QK^\top = \begin{bmatrix} 0.7730 & 0.6600 & 0.6730 \\ 0.4300 & 0.4000 & 0.3950 \\ 0.2701 & 0.2300 & 0.2633 \end{bmatrix} \qquad Softmax(QK^\top) = \begin{bmatrix} 0.3902 & 0.2040 & 0.2098 \\ 0.1869 & 0.7522 & 0.1341 \\ 0.3103 & 0.1631 & 0.3633 \end{bmatrix}$$

Aplicando a distribuição obtida à matriz de valores (V), $Softmax(QK^\top)V$:

$$\mathrm{Sa}(X) = \begin{bmatrix} 0.3902 & 0.2040 & 0.2098 \\ 0.1869 & 0.7522 & 0.1341 \\ 0.3103 & 0.1631 & 0.3633 \end{bmatrix} \begin{bmatrix} 0.5530 & 0.5382 \\ 0.7668 & 0.8359 \\ 0.8591 & 0.7898 \end{bmatrix} = \begin{bmatrix} 0.5524 & 0.5462 \\ 0.7957 & 0.8353 \\ 0.6089 & 0.5903 \end{bmatrix}$$

Melhorias sobre Self Attention

Scaled Dot-Product

- As duas funções de Atenção mais utilizadas são o produto escalar e a chamada Similaridade Aditiva, que utiliza uma rede neural com 1 camada oculta para o cálculo da matriz de compatibilidade
- Apesar da similaridade aditiva desempenhar melhor para valores médios de d_{model}, não escala bem.
- Utilizar apenas o produto escalar, apesar de muito eficiente, com grandes valores para d_{model} a *Softmax* apresenta desaparecimento de gradiente.

Solução: conter o produto escalar através de um fator:

$$\frac{1}{\sqrt{d_{model}}} \longrightarrow \mathbf{SA}(Q, K, V) = Softmax\Big(\frac{QK^\top}{\sqrt{d_{model}}}\Big)V$$

Scaled Dot-Product

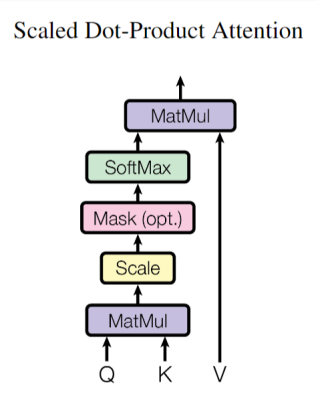


Figura 7: Esquemática de Atenção modificada

Multi-head Attention

- Comumente, mecanismos de Self-Attention são utilizados em paralelo, adquirindo o nome *Multi-head* Self Attention.
- Com essa alteração, passa-se a ter H cabeças de Atenção, tal que também dividimos Q_h, K_h e V_h
- A Atenção computada para a cabeça h é dada por:

$$\mathbf{SA}_h = V_h \cdot Softmax\left(\frac{Q_h K_h^\top}{\sqrt{d_{model}}}\right)$$

- Tipicamente dividimos Q_h, K_h e $V_h \in \mathbb{R}^{N \times \frac{d}{H}}$ (Figura 8)

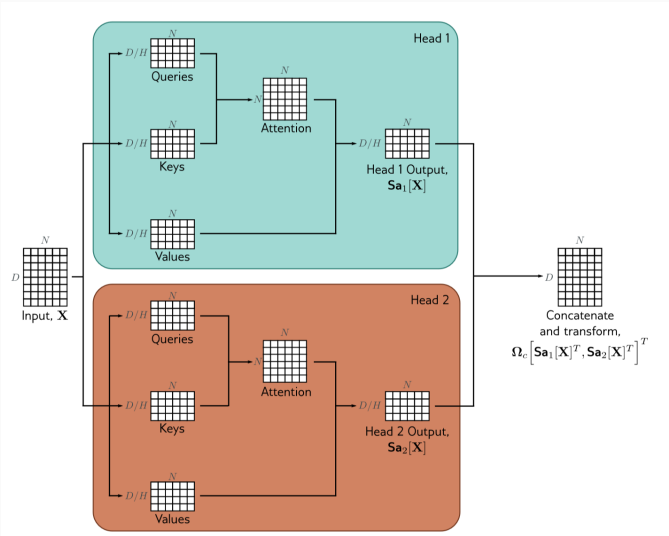


Figura 8: Diferentes matrizes W_q^h, W_k^h, W_v^h expressam novos aspectos da atenção

Ao realizar a divisão do aprendizado para incorporar diferentes aspectos/relações entre elementos da entrada, passa a ser necessária a reintegração das saídas de cada cabeça:

$$\textbf{MhSa} = Concat(Sa_1, Sa_2, \ \dots \ , Sa_n)W^O$$

Ainda, é importante notar que

$$head_i = Sa_i(QW_q^i, KW_k^i, VW_v^i)$$

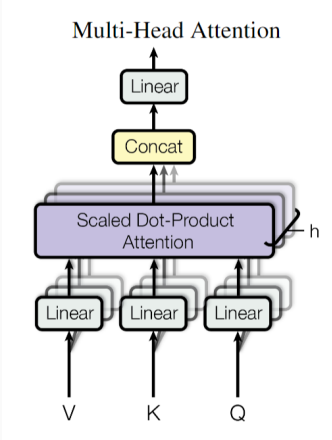


Figura 9: A separação passa a ter $W^i \in \mathbb{R}^{d_{model} \times \frac{d}{H}}$

Masked Multi-head Attention

Multi-head na qual deseja-se prevenir que alguns valores sejam desconsiderados através de uma máscara (M) atuando na probabilidade:

$$\textbf{MaskedSa}(Q, K, V) = Softmax\left(\frac{QK^\top + M}{\sqrt{d_{model}}}\right) \cdot V$$

$$M = \begin{bmatrix} 0 & -\infty & \cdots & -\infty & -\infty \\ 0 & 0 & \cdots & -\infty & -\infty \\ \vdots & & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 & -\infty \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix}$$

Encoder-Decoder Transformer

Modelo Encoder-Decoder

Modelos de Encoder-Decoder são comumente utilizados para tarefas de tradução em NLP, onde:

- **Encoder** é responsável por obter uma boa representação da sentença original
- **Decoder** computa a informação codificada e gera uma saída traduzida

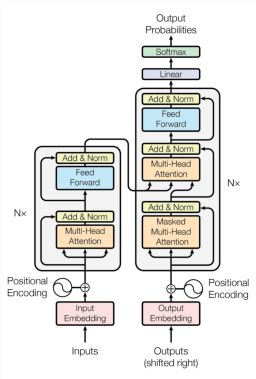


Figura 10: Encoder à esquerda e Decoder à direita.

Encoder

Após a etapa de **MhSa**, utilizamos a saída de dimensão $N \times d_{model}$ como uma grande fonte de informação contextual de cada palavra! Através das conexões residuais preservamos informação dos **Embeddings** e **Positional Encoding**.

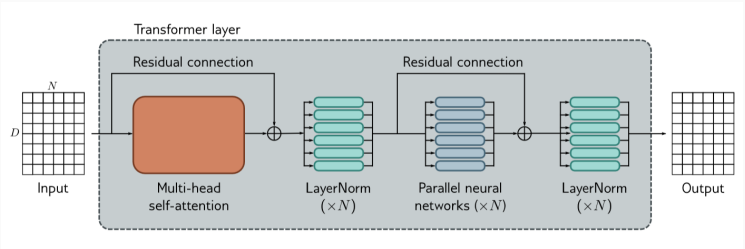


Figura 11: Conexões residuais facilitam o aprendizado através de Gradiente Descendente

Layer Normalization

Uma parte **fundamental** para o funcionamento dos Transformers devido ao *grande número de camadas interdependentes* durante o GD, as etapas de normalização servem para estabilizar e **acelerar a convergência**.

- A normalização garante que a média será $\mu = 0$ e sua variância $\sigma^2 = 1$
- Na prática, há ainda a introdução de 2 parâmetros β, γ aprendidos que introduzem flutuações, garantindo certa plasticidade ao modelo (parâmetros compensatórios)

$$h_i = \frac{\gamma}{\sigma}(h_i - \mu) + \beta, \quad \mu = \frac{1}{H}\sum_{i=1}^{H} h_i, \quad \sigma = \sqrt{\frac{1}{H}\sum_{i=1}^{H}(h_i - \mu)^2}$$

Position-Wise Feed Forward Net

Além das camadas de Atenção, cada camada de Transformer contém uma camada de FFNs que correspondem a duas transformações lineares que ajustam o aprendizado após o processo de Atenção:

$$FFN(x) = \max(0,\ xW_1 + b_1)W_2 + b_2$$

Decoder

Muito semelhante ao Encoder, utiliza-se uma camada de **Masked Multi-head Attention** em conjunto com **Teacher Forcing**.

- O uso de uma máscara com zeros no trinângulo inferior e $-\infty$ no triângulo superior faz com que o modelo leve em conta apenas informação prévia, isto é, introduz causalidade
- A prática de Teacher Forcing impede a formação de uma relação de recorrência entre um Output $t-1$ e Input t durante o treinamento. Ainda, inclui informação de embeddings no novo idioma.

	YOUR	CAT	IS	A	LOVELY	CAT
YOUR	0.268	~~0.119~~	~~0.134~~	~~0.148~~	~~0.179~~	~~0.152~~
CAT	0.124	0.278	~~0.201~~	~~0.128~~	~~0.154~~	~~0.115~~
IS	0.147	0.132	0.262	~~0.097~~	~~0.218~~	~~0.145~~
A	0.210	0.128	0.206	0.212	~~0.119~~	~~0.125~~
LOVELY	0.146	0.158	0.152	0.143	0.227	~~0.174~~
CAT	0.195	0.114	0.203	0.103	0.157	0.229

Figura 12: Matriz de Compatibilidade com máscara aplicada

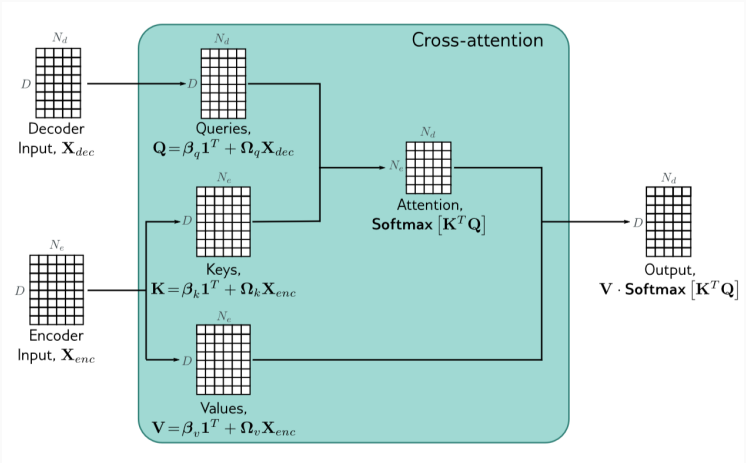
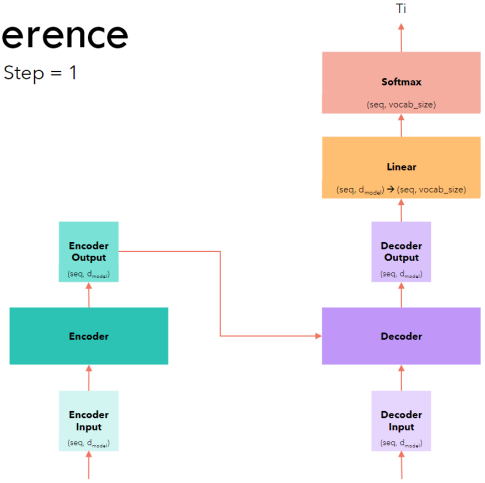


Figura 13: Carrega-se a informação do Encoder através dos **Keys** e **Values**.

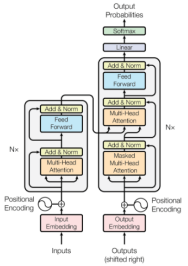
Inference

Time Step = 1



We select a token from the vocabulary corresponding to the position of the token with the maximum value.

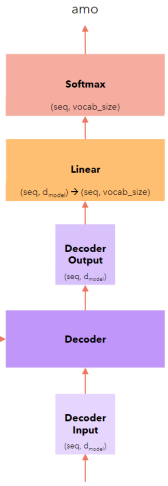
The output of the last layer is commonly known as **logits**



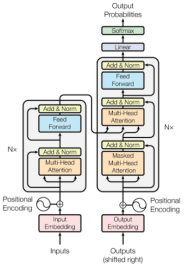
* Both sequences will have same length thanks to padding

Inference

Time Step = 2



Since decoder input now contains **two** tokens, we select the softmax corresponding to the second token.



Append the previously output word to the decoder input

Umar Jamil - https://github.com/hkproj/transformer-from-scratch-notes

Variações de Modelos

Hoje, há incontáveis variações de modelos para diferentes usos baseados neste modelo original

- BERT - Encoder Only Transformers
- GPT3 - Decoder Only Transformers
- Image Transformers
- etc.

Conclusão

- Modelo de Transformers é extremamente poderoso!
- Uma arquitetura que une o melhor dos dois mundos de LSTMs e CNNs
- Transformers operam em Embeddings de alta dimensionalidade e utilizam a informação na íntegra (não há problema do esquecimento a longo prazo)
- Treinamento em passada única - **totalmente paralelizável**
- Implementação na próxima apresentação

Obrigado!