VINI CARIDÁ

Human-Centered Data & AI

VINI CARIDÁ
Human-Centered Data & AI

Vinicius Caridá, Ph.D.

- Head of Digital Customer Service Platforms, PCP, WFM, Data and AI - Itaú Unibanco
- MBA Professor - FIAP

@vinicius caridá @vfcarida @vinicius caridá @vfcarida @vinicius caridá @vfcarida

Processamento de Linguagem Natural (NLP)

Parte 1 de 3 relembrando...

Vinicius F. Caridá

Data Prep NLP

- Remoção de ruído
- Normalização
- Tokenização
- Vocabulário

Representação Esparsa

- One Hot encoding
- Bag of Words
- TF-IDF

Representação Densa (Embeddings)

- Word2Vec (CBOW e Skip-gram)
- FastText
- ELMo

Os últimos Jedi Language Models

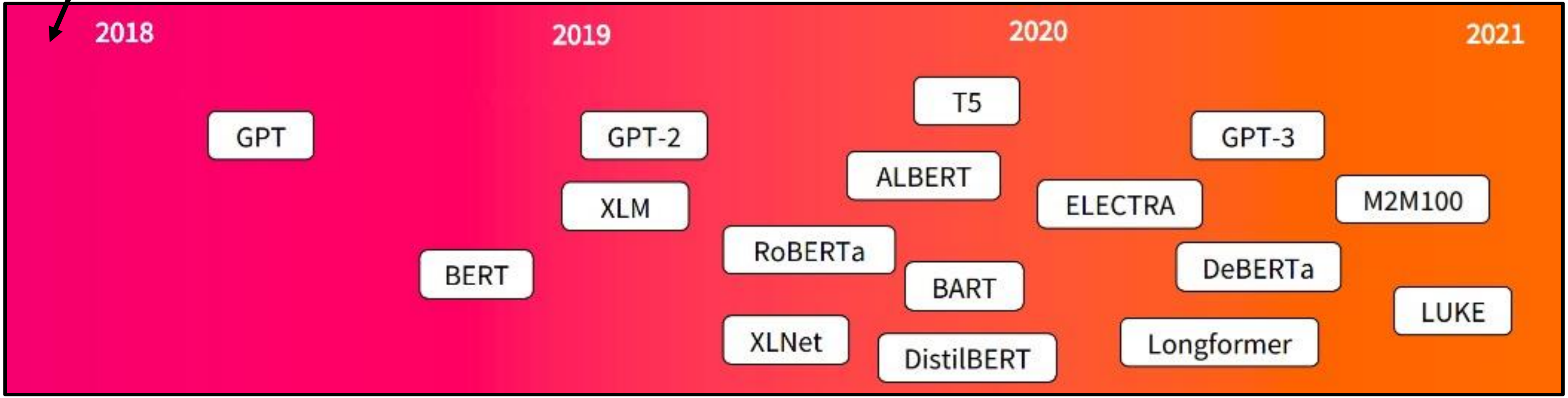

Attention Is All You Need

citações: **+31k**

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

6 Dec 2017

Após a publicação do artigo ***Attention Is All You Need*** que apresentou a arquitetura Transformers, embeddings contextuais tiveram grande avanço comparado com o algoritmo **ELMo**.



fonte: *hugging face*

Processamento de Linguagem Natural (NLP)

Parte 2 de 3

Vinicius F. Caridá

Agenda

80
Min.

BERT

- Pré treino e Fine Tuning
- Tokenizers & Embeddings
- Rorschach test com BERT Viz
- Hands-on (Notebook)

BERT

BERT

Pre-training of Deep Bidirectional Transformers for Language Understanding

https://arxiv.org/abs/1810.04805

Human-Centered Data & AI

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

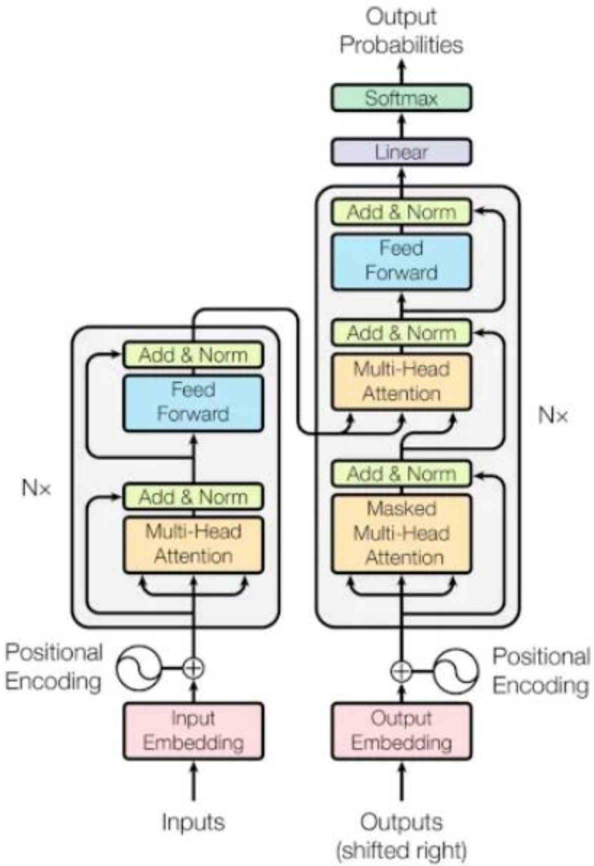
Łukasz Kaiser* Google Brain lukaszkaiser@google.com

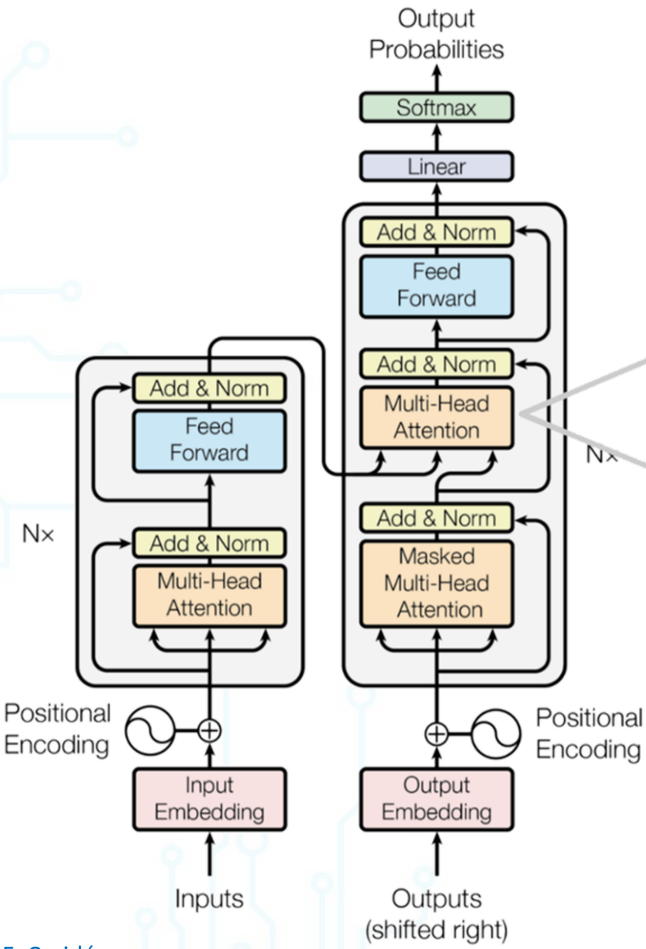
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

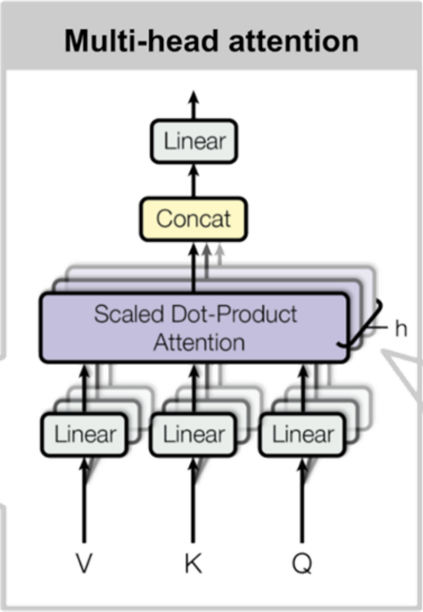
Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task,

https://arxiv.org/abs/1706.03762

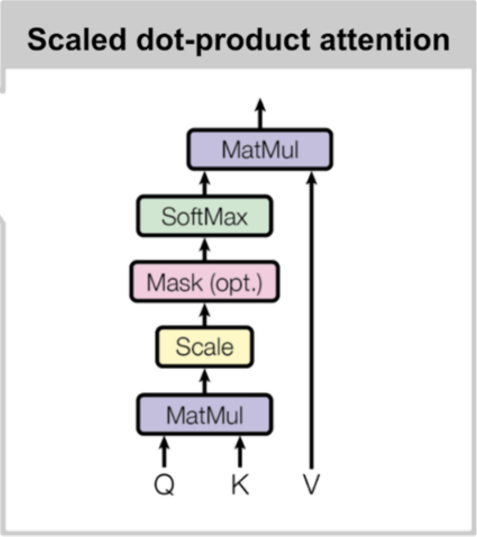






Zoom-In!

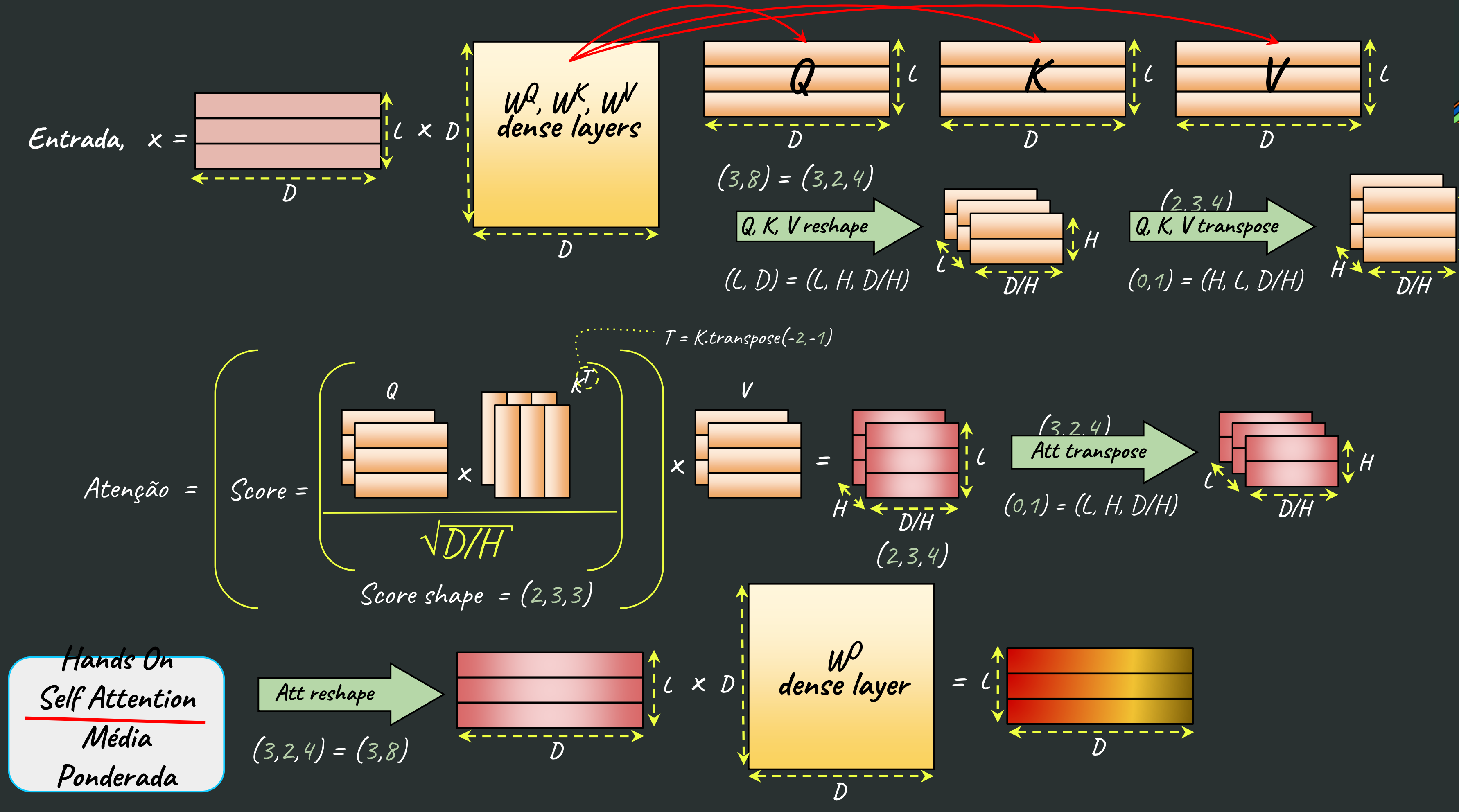
$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

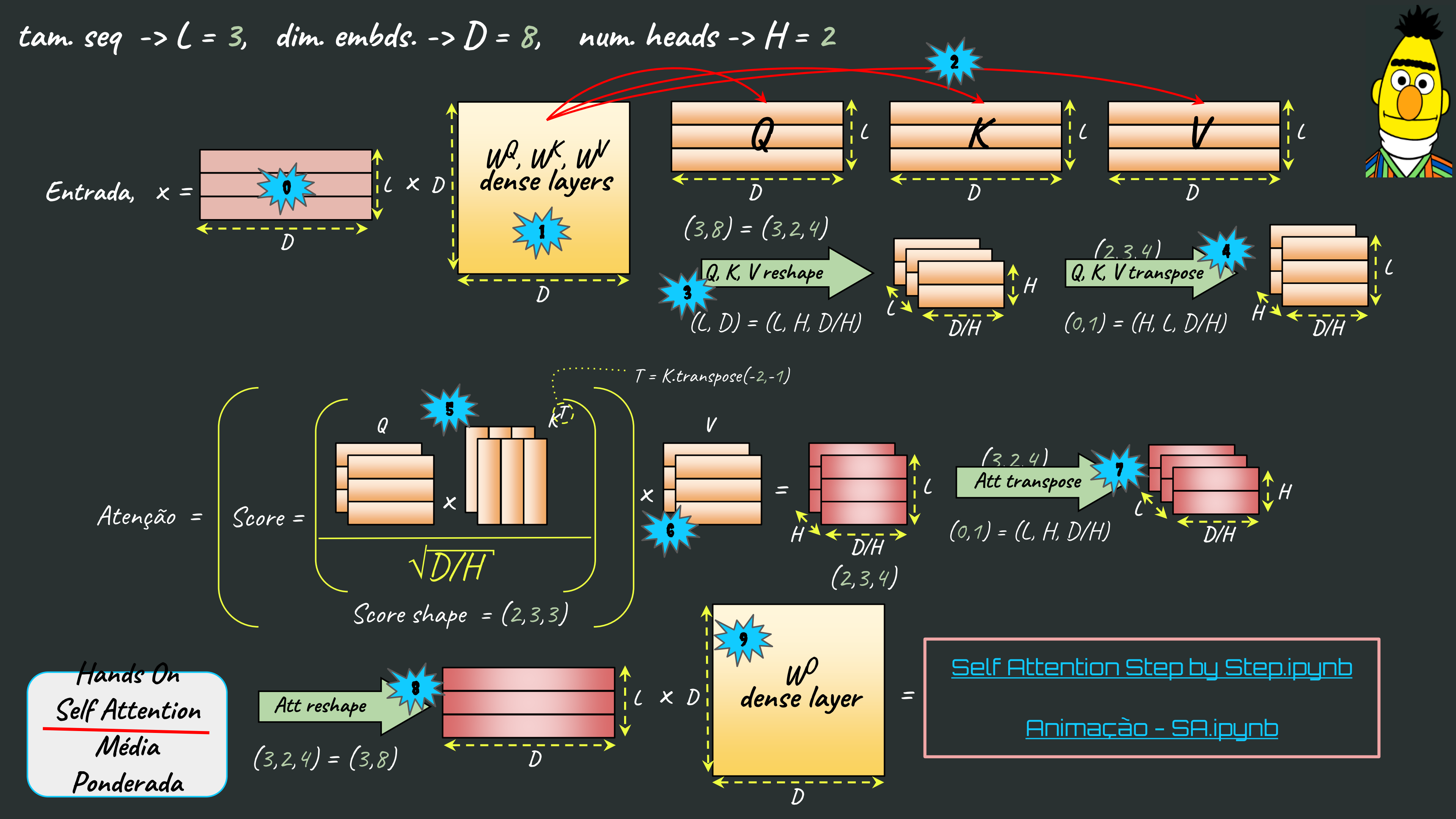


Zoom-In!

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

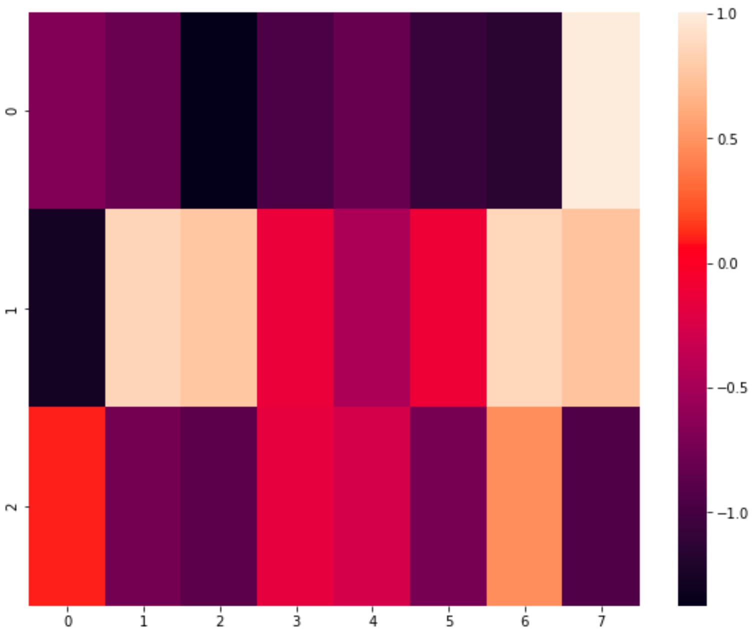
tam. seq -> $L = 3$, dim. embds. -> $D = 8$, num. heads -> $H = 2$

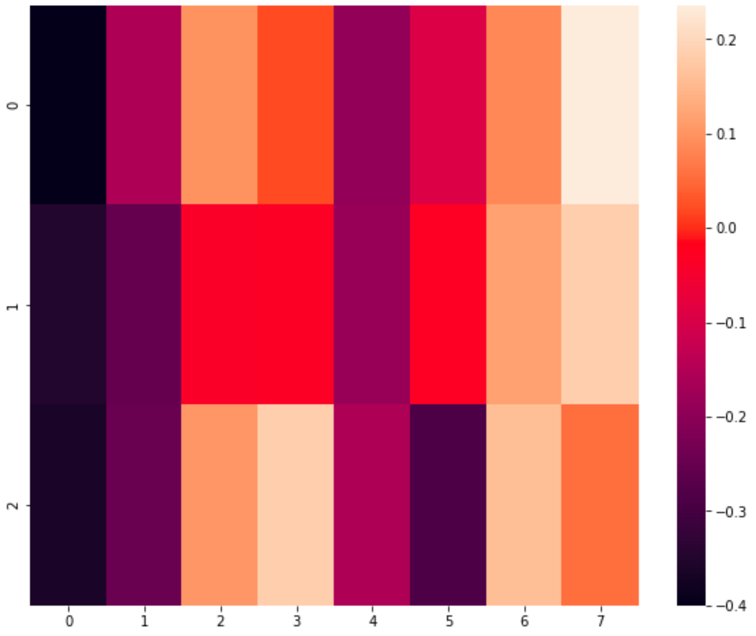




tam. seq -> L = 3, dim. embds. -> D = 8, num. heads -> H = 2
Entrada, x =
L x D
D
W^Q, W^K, W^V dense layers
Q
K
V
(3,8) = (3,2,4)
Q, K, V reshape
(L, D) = (L, H, D/H)
H
D/H
(2,3,4)
Q, K, V transpose
(0,1) = (H, L, D/H)
T = K.transpose(-2,-1)
Atenção =
Score =
x
$\sqrt{D/H}$
Score shape = (2,3,3)
(3,2,4)
Att transpose
(0,1) = (L, H, D/H)
(2,3,4)
Hands On
Self Attention
Média
Ponderada
Att reshape
(3,2,4) = (3,8)
W^O dense layer
Self Attention Step by Step.ipynb
Animação - SA.ipynb

```
text_tokens = ['Quero', 'um', 'sorvete']
text_ids = [27,11,83] # tokens sintéticos
```





- Embeddings
- ATT

DeepMind Attention explained:

https://www.youtube.com/watch?v=AIiwuClvH6k&t=5559s

The Illustrated Transformer

o/illustrated-transformer/

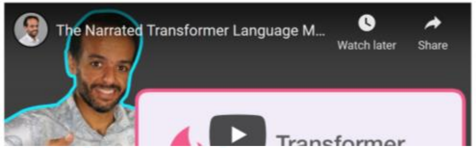
The Illustrated Transformer

Discussions: Hacker News (65 points, 4 comments), Reddit r/MachineLearning (29 points, 3 comments)
Translations: Chinese (Simplified), French 1, French 2, Japanese, Korean, Russian, Spanish, Vietnamese
Watch: MIT's Deep Learning State of the Art lecture referencing this post

In the previous post, we looked at Attention – a ubiquitous method in modern deep learning models. Attention is a concept that helped improve the performance of neural machine translation applications. In this post, we will look at **The Transformer** – a model that uses attention to boost the speed with which these models can be trained. The Transformer outperforms the Google Neural Machine Translation model in specific tasks. The biggest benefit, however, comes from how The Transformer lends itself to parallelization. It is in fact Google Cloud's recommendation to use The Transformer as a reference model to use their Cloud TPU offering. So let's try to break the model apart and look at how it functions.

The Transformer was proposed in the paper Attention is All You Need. A TensorFlow implementation of it is available as a part of the Tensor2Tensor package. Harvard's NLP group created a guide annotating the paper with PyTorch implementation. In this post, we will attempt to oversimplify things a bit and introduce the concepts one by one to hopefully make it easier to understand to people without in-depth knowledge of the subject matter.

2020 Update: I've created a "Narrated Transformer" video which is a gentler approach to the topic:



o/illustrated-transformer/

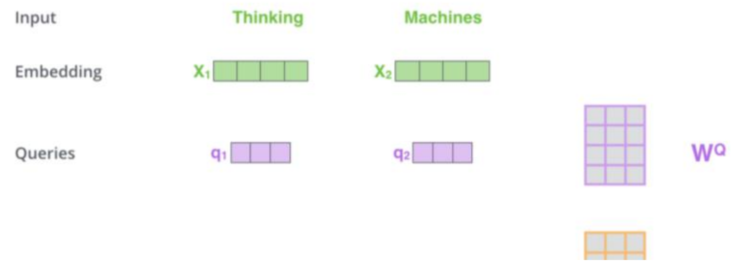
Be sure to check out the Tensor2Tensor notebook where you can load a Transformer model, and examine it using this interactive visualization.

Self-Attention in Detail

Let's first look at how to calculate self-attention using vectors, then proceed to look at how it's actually implemented – using matrices.

The **first step** in calculating self-attention is to create three vectors from each of the encoder's input vectors (in this case, the embedding of each word). So for each word, we create a Query vector, a Key vector, and a Value vector. These vectors are created by multiplying the embedding by three matrices that we trained during the training process.

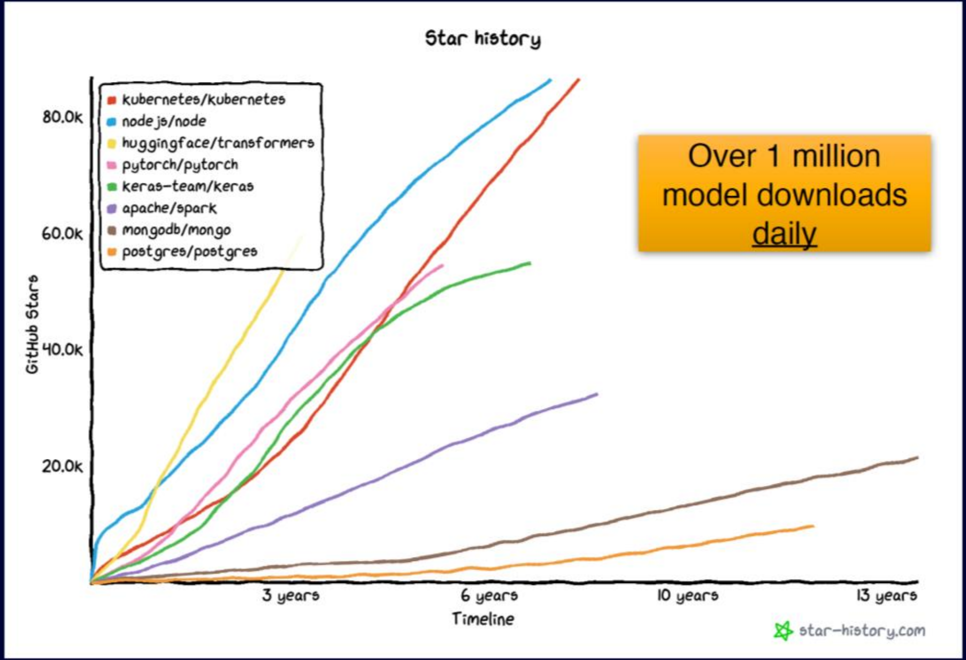
Notice that these new vectors are smaller in dimension than the embedding vector. Their dimensionality is 64, while the embedding and encoder input/output vectors have dimensionality of 512. They don't HAVE to be smaller, this is an architecture choice to make the computation of multiheaded attention (mostly) constant.



https://jalammar.github.io/illustrated-transformer/

Transformers: one of the fastest-growing open source projects

https://github.com/huggingface/transformers/



Over 1 million model downloads daily

"Transformers are emerging as a general-purpose architecture for ML"

https://www.stateof.ai/

RNN and CNN usage down, Transformers usage up

https://www.kaggle.com/kaggle-survey-2021

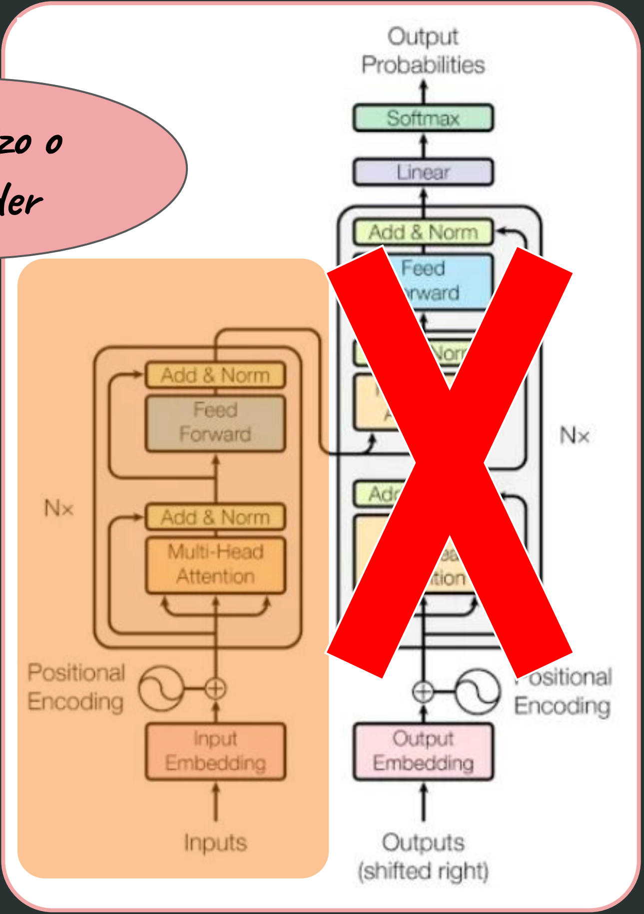
BERT

Pre-training of Deep Bidirectional Transformers for Language Understanding

https://arxiv.org/abs/1810.04805

BERT

VINI CARIDÁ
Human-Centered Data & AI



BERT - GLUE Tasks

- **CoLA** Se a sentença está escrita gramaticalmente correta
- **SST-2** Análise de reviews de filmes
- **MRPC** Se a sentença é paráfrase
- **STS-B** Similaridade entre 2 sentenças
- **QQP** Se 2 questões são similares
- **MLNI** Se a sentença "B" é uma contradição de uma sentença "A"
- **QNLI** Se uma sentença "B" contém uma resposta para a sentença "A"
- **RTE** Se a sentença "B" está vinculada com uma sentença "A"
- **WNLI** Se uma sentença "B" substitui corretamente o pronome de uma sentença "A"

BERT - GLUE Score

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

BERT - GLUE Tasks

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Benchmark ~ 90 avg.
(DeBERTA, Microsoft)

SuperGLUE

Paper

Rank	Name	Model	URL	Score
1	ERNIE Team - Baidu	ERNIE		90.9
2	DeBERTa Team - Microsoft	DeBERTa / TuringNLRv4		90.8
3	HFL iFLYTEK	MacALBERT + DKM		90.7
4	Alibaba DAMO NLP	StructBERT + TAPT		90.6
5	PING-AN Omni-Sinitic	ALBERT + DAAF + NAS		90.6
6	T5 Team - Google	T5		90.3
7	Microsoft D365 AI & MSR AI & GATECH	MT-DNN-SMART		89.9
8	Huawei Noah's Ark Lab	NEZHA-Large		89.8
9	Zihang Dai	Funnel-Transformer (Ensemble B10-10-10H1024)		89.7
10	ELECTRA Team	ELECTRA-Large + Standard Tricks		89.4

SuperGLUE

leaderboard

Text to model

sentença: Quero um carro novo

Text to model

sentença: Quero um carro novo

- 1o passo: Tokenizar := [Quero][um][carro][novo]

Text to model

sentença: Quero um carro novo

- 1o passo: Tokenizar := [Quero] [um] [carro] [novo]

- 2o passo:

one hot 😒

Quero	[1,0,0,0,...,0]
um	[0,1,0,0,...,0]
carro	[0,0,0,1,...,0]
novo	[0,0,1,0,...,0]

Text to model

sentença: Quero um carro novo

- 1o passo: Tokenizar := [Quero] [um] [carro] [novo]

- 2o passo:

- 2o passo:

one hot

Quero	[1,0,0,0,...,0]
um	[0,1,0,0,...,0]
carro	[0,0,0,1,...,0]
novo	[0,0,1,0,...,0]

embedding

Quero	[2.21,-3.32,...,0.89]
um	[-1.27,2.80,...,4.05]
carro	[0.37,-1.98,...,3.09]
novo	[0.77,-0.88,...,2.16]

BERT - Tokenize

Formas de tokenização

BERT - Tokenize

Formas de tokenização

sentença: Chewie, we're home!

BERT - Tokenize

Formas de tokenização

sentença: Chewie, we're home!

- por espaços: [Chewie,] [we're] [home!]

BERT - Tokenize

Formas de tokenização

sentença: Chewie, we're home!

por espaços: [Che [illegible] home!]

SUBÓTIMO

BERT - Tokenize

Formas de tokenização

sentença: Chewie, we're home!

- por espaços: [Che ... home!]

  SUBÓTIMO

- por pontuação: [Chewie][,][we]['][re][home][!]

BERT - Tokenize

Formas de tokenização

sentença: Chewie, we're home!

- por espaços: [Che ... home!]

  SUBÓTIMO

- por pontuação: [Chewie] [,] [we] ['] [re] [home] [!]

¿ we're != we are ?

BERT - Tokenize

Formas de tokenização

sentença: Chewie, we're home!

- por espaços: [Che SUBÓTIMO home!]
- por pontuação: [Chev SUBÓTIMO we]['][re][home][!]

¿we're != we are ?

BERT - Tokenize

Formas de tokenização

sentença: Chewie, we're home!

- spaCy: [Chewie] [,] [we] ['re] [home] [!]

BERT - Tokenize

Formas de tokenização

sentença: Chewie, we're home!

- spaCy: [Chewie][,][we]['re][home][!]

por pontuação + por regras

BERT - Tokenize

Formas de tokenização

sentença: Chewie, we're home!

- spaCy: [Chewie][,][we]['re][home][!]

por pontuação + por regras

Qual o problema 😬?

BERT - Tokenize

Formas de tokenização

sentença: Chewie, we're home!

- spaCy: [Chewie] [,] [we] ['re] [home] [!]

por pontuação + por regras

Qual o problema 😬?

Word-level → Vocab. massivo

Transformer XL
word-level
~ 268K tokens

BERT - Tokenize

Então word-level é BIG, e se fizer char-level?

BERT - Tokenize

Então word-level é BIG, e se fizer char-level?

- **Mais simples**: reduz complexidade de memória e tempo 😏

BERT - Tokenize

Então word-level é BIG, e se fizer char-level?

- **Mais simples**: reduz complexidade de memória e tempo
- **Mais complexo**: aprendizado mais complexo de representações 😐

Ex.: aprender representação independente
do contexto do char m

BERT - Tokenize

Então word-level é BIG, e se fizer char-level?

- **Mais simples**: reduz complexidade de memória e tempo
- **Mais complexo**: aprendizado mais complexo de representações

Ex.: aprender representação independente do contexto do char m

[madeira] » [m] [a] [d] [e] [i] [r] [a] 😕

BERT - Tokenize

Então word-level é BIG, e se fizer char-level?

- **Mais simples**: reduz complexidade de memória e tempo
- **Mais complexo**: aprendizado mais complexo de representações

Ex.: aprender representação independente do contexto do char m

[madeira] » [m] [a] [d] [e] [i] [r] [a]

~?

[c] [a] [d] [e] [i] [r] [a]

Vocab. Reduzido
+
Boas Representações

- **Subword tokenizers**
  - Byte-Pair Encoding BPE
  - Unigram Subword
  - Wordpiece
  - Sentence piece

BERT - Tokenize

Vocab. Reduzido
+
Boas
Representações

- **Subword tokenizers**
  - Byte-Pair Encoding BPE
  - Unigram Subword
  - Wordpiece
  - Sentence piece

```
("mau",10), ("baú",5), ("itaú",7),("uau",10)
```

pré-tokenização

- BPE :=

BERT - Tokenize

Vocab. Reduzido
+
Boas
Representações

- **Subword tokenizers**
  - Byte-Pair Encoding BPE
  - Unigram Subword
  - Wordpiece
  - Sentence piece

```
("mau",10), ("baú",5), ("itaú",7),("uau",10)
```
pré-tokenização

```
["a", "b", "i", "m", "t", "u", "ú"]
```
vocab. base

- BPE :=

BERT - Tokenize

Vocab. Reduzido
+
Boas
Representações

- **Subword tokenizers**
  - Byte-Pair Encoding BPE
  - Unigram Subword
  - Wordpiece
  - Sentence piece

- BPE :=

```
("mau",10), ("baú",5), ("itaú",7),("uau",10)
["a", "b", "i", "m", "t", "u", "ú"]
("m" "a" "u", 10), ("b" "a" "ú", 5), ("i" "t" "a" "ú", 7),("u" "a" "u", 4)
```

pré-tokenização
vocab. base

BERT - Tokenize

Vocab. Reduzido
+
Boas Representações

- **Subword tokenizers**
  - Byte-Pair Encoding BPE
  - Unigram Subword
  - Wordpiece
  - Sentence piece

- BPE :=

```
("mau",10), ("baú",5), ("itaú",7),("uau",10)
```
pré-tokenização

```
["a", "b", "i", "m", "t", "u", "ú"]
```
vocab. base

```
("m" "a" "u", 10), ("b" "a" "ú", 5), ("i" "t" "a" "ú", 7),("u" "a" "u", 4)
```

```
("m" "au", 10), ("b" "aú", 5), ("i" "t" "aú", 7),("u" "au", 4)
```

BERT - Tokenize

Vocab. Reduzido
+
Boas
Representações

- **Subword tokenizers**
  - Byte-Pair Encoding BPE
  - Unigram Subword
  - Wordpiece
  - Sentence piece

- BPE :=

```
("mau",10), ("baú",5), ("itaú",7),("uau",10)
```
pré-tokenização

```
["a", "b", "i", "m", "t", "u", "ú"]
```
vocab. base

```
("m" "a" "u", 10), ("b" "a" "ú", 5), ("i" "t" "a" "ú", 7),("u" "a" "u", 4)
("m" "au", 10), ("b" "aú", 5), ("i" "t" "aú", 7),("u" "au", 4)
["a", "b", "i", "m", "t", "u", "ú", "au", "aú"]
```
novo vocab. base

BERT - Tokenize

Vocab. Reduzido
+
Boas
Representações

- Subword tokenizers
  - Byte-Pair Encoding BPE
  - Unigram Subword
  - Wordpiece
  - Sentence piece

- BPE := treino

```
("mau",10), ("baú",5), ("itaú",7),("uau",10)
```
pré-tokenização

```
["a", "b", "i", "m", "t", "u", "ú"]
```
vocab. base

```
("m" "a" "u", 10), ("b" "a" "ú", 5), ("i" "t" "a" "ú", 7),("u" "a" "u", 4)
("m" "au", 10), ("b" "aú", 5), ("i" "t" "aú", 7),("u" "au", 4)
["a", "b", "i", "m", "t", "u", "ú", "au", "aú"]
```
novo vocab. base

⋮

vocab. base final

BERT - Tokenize

Vocab. Reduzido
+
Boas
Representações

- **Subword tokenizers**
  - Byte-Pair Encoding BPE
  - Unigram Subword
  - Wordpiece
  - Sentence piece

- BPE := treino

("mau",10), ("baú",5), ("itaú",7),("uau",10) — pré-tokenização

["a", "b", "i", "m", "t", "u", "ú"] — vocab. base

("m" "a" "u", 10), ("b" "a" "ú", 5), ("i" "t" "a" "ú", 7),("u" "a" "u", 4)

("m" "au", 10), ("b" "aú", 5), ("i" "t" "aú", 7),("u" "au", 4)

["a", "b", "i", "m", "t", "u", "ú", "au", "aú"] — novo vocab. base

⋮

vocab. base final

GPT-2
Roberta
XLM
~50K tokens

prever "auê" → ["au", "<unk>"]

BERT - Tokenize

Problema com BPE:

BPE faz a união tokens com base na frequência mais alta. Desta forma, é um algoritmo guloso.

A desvantagem da abordagem gulosa é que ela pode resultar em um vocab. ambiguo.

deep learning

de/ep le/ar/n/ing

d/ee/p le/ar/n/ing

d/e/ep le/ar/n/ing

BERT - Tokenize

Problema com BPE:

BPE faz a união tokens com base na frequência mais alta. Desta forma, é um algoritmo guloso.

A desvantagem da abordagem gulosa é que ela pode resultar em um vocab. ambiguo.

deep learning

de/ep le/ar/n/ing
d/ee/p le/ar/n/ing
d/e/ep le/ar/n/ing

Wordpiece (BERT, DistilBERT, Electra)

WordPiece é um pouco diferente do BPE, além de contar a freq. das palavras ele mede a prob. de fazer a união de chars.

Exemplo: "a" e "u" é unido se a prob. de "au" dividido por "a" e por "u" é maior do que para qualquer outra união de chars.

mBERT, BERTimbau e BERTaú - Tokenizers

- sentença: Never tell me the odds.
  - mBERT
  - BERTimbau
  - BERTaú

mBERT, BERTimbau e BERTaú - Tokenizers

- sentença: Never tell me the odds.
  - mBERT `['never', 'tell', 'me', 'the', 'odds', '.']`
  - BERTimbau
  - BERTaú

mBERT, BERTimbau e BERTaú - Tokenizers

- sentença: Never tell me the odds.
  - mBERT ['never', 'tell', 'me', 'the', 'odds', '.']
  - BERTimbau ['Ne', '##ver', 'tel', '##l', 'me', 'the', 'o', '##dd', '##s', '.']
  - BERTaú

mBERT, BERTimbau e BERTaú - Tokenizers

- sentença: Never tell me the odds.
  - mBERT `['never', 'tell', 'me', 'the', 'odds', '.']`
  - BERTimbau `['Ne', '##ver', 'tel', '##l', 'me', 'the', 'o', '##dd', '##s', '.']`
  - BERTaú `['neve', '##r', 'tel', '##l', 'me', 'the', 'od', '##ds', '.']`

mBERT, BERTimbau e BERTaú - Tokenizers

- sentença: Never tell me the odds.
  - mBERT `['never', 'tell', 'me', 'the', 'odds', '.']`
  - BERTimbau `['Ne', '##ver', 'tel', '##l', 'me', 'the', 'o', '##dd', '##s', '.']`
  - BERTaú `['neve', '##r', 'tel', '##l', 'me', 'the', 'od', '##ds', '.']`

- sentença: Quero um cartão de crédito.
  - mBERT
  - BERTimbau
  - BERTaú

mBERT, BERTimbau e BERTaú - Tokenizers

- sentença: Never tell me the odds.
  - mBERT ['never', 'tell', 'me', 'the', 'odds', '.']
  - BERTimbau ['Ne', '##ver', 'tel', '##l', 'me', 'the', 'o', '##dd', '##s', '.']
  - BERTaú ['neve', '##r', 'tel', '##l', 'me', 'the', 'od', '##ds', '.']

- sentença: Quero um cartão de crédito.
  - mBERT ['quer', '##o', 'um', 'carta', '##o', 'de', 'credito']
  - BERTimbau
  - BERTaú

mBERT, BERTimbau e BERTaú - Tokenizers

- sentença: Never tell me the odds.
  - mBERT `['never', 'tell', 'me', 'the', 'odds', '.']`
  - BERTimbau `['Ne', '##ver', 'tel', '##l', 'me', 'the', 'o', '##dd', '##s', '.']`
  - BERTaú `['neve', '##r', 'tel', '##l', 'me', 'the', 'od', '##ds', '.']`

- sentença: Quero um cartão de crédito.
  - mBERT `['quer', '##o', 'um', 'carta', '##o', 'de', 'credito']`
  - BERTimbau `['Quer', '##o', 'um', 'cartão', 'de', 'crédito']`
  - BERTaú

mBERT, BERTimbau e BERTaú - Tokenizers

- sentença: Never tell me the odds.
  - mBERT ['never', 'tell', 'me', 'the', 'odds', '.']
  - BERTimbau ['Ne', '##ver', 'tel', '##l', 'me', 'the', 'o', '##dd', '##s', '.']
  - BERTaú ['neve', '##r', 'tel', '##l', 'me', 'the', 'od', '##ds', '.']

- sentença: Quero um cartão de crédito.
  - mBERT ['quer', '##o', 'um', 'carta', '##o', 'de', 'credito']
  - BERTimbau ['Quer', '##o', 'um', 'cartão', 'de', 'crédito']
  - BERTaú ['quero', 'um', 'cartao', 'de', 'credito']

BERT - Tokenize

Tokens especiais

- **BERTaú** - special tokens

0: [PAD]

1: [UNK]

2: [CLS]

3: [SEP]

4: [MASK]

é relacionado com o max_len ex. para max_len=10

```
"Uma sentença": [2, 335, 12286, 155, 951, 3, 0, 0, 0, 0]
"E outra":      [2, 37  , 929   , 3  , 0  , 0, 0, 0, 0, 0]
```

BERT - Tokenize

Tokens especiais

- **BERTaú** - special tokens

0: [PAD]

1: [UNK]

2: [CLS]

3: [SEP]

4: [MASK]

quando não existe a word/subword no vocab. base.
O BERTaú possui [UNK] para qualquer número
(dados anonimizados).

BERT - Tokenize

Tokens especiais

- **BERTaú** - special tokens

0: [PAD]

1: [UNK]

2: [CLS]

3: [SEP]

4: [MASK]

[CLS] Uma sentença. [SEP] E outra! [SEP]

[2, 335, 12286, 155, 951, 3, 37, 929, 3, 0]

BERT - Tokenize

Tokens especiais

- **BERTaú** - special tokens

0: [PAD]

1: [UNK]

2: [CLS]

3: [SEP]

4: [MASK] → usado durante o pré-treino e downstream tasks de **LM**

BERTaú e BERT - Tokenizers

Tokens especiais

- **BERTaú** - special tokens

  0: [PAD]

  1: [UNK]

  2: [CLS]

  3: [SEP]

  4: [MASK]

- **mBERT** - special tokens

  0: [PAD]

  100: [UNK]

  101: [CLS]

  102: [SEP]

  103: [MASK]

Uma sentença. E outra!

```
['[CLS]', 'uma', 'sente', '##n', '##ca', '.', '[SEP]', 'e', 'outra', '!', '[SEP]', , '[PAD]']
```

```
max_length=12
```

tokens

Uma sentença. E outra!

input_ids

```
['[CLS]', 'uma', 'sente', '##n', '##ca', '.', '[SEP]', 'e', 'outra', '!', '[SEP]', , '[PAD]']
```

tokens

max_length=12

Uma sentença. E outra!

[2 , 335 , 12286 , 155 , 951 , 18, 3 , 37 , 929 , 5 , 3, 0]

input_ids

['[CLS]', 'uma', 'sente', '##n', '##ca', '.', '[SEP]', 'e', 'outra', '!', '[SEP]', , '[PAD]']

tokens

max_length=12

Uma sentença. E outra!

token_type_ids

```
[ 2 , 335 , 12286 , 155 , 951 , 18, 3 , 37 , 929 , 5 , 3, 0 ]
```

input_ids

```
['[CLS]', 'uma', 'sente', '##n', '##ca', '.', '[SEP]', 'e', 'outra', '!', '[SEP]', , '[PAD]']
```

max_length=12

tokens

Uma sentença. E outra!

```
[ 0 , 0 , 0 , 0 , 0 , 0, 0, 1, 1, 1, 1, 0 ]
```

token_type_ids

```
[ 2 , 335 , 12286 , 155 , 951 , 18, 3 , 37 , 929 , 5 , 3, 0 ]
```

input_ids

```
['[CLS]', 'uma', 'sente', '##n', '##ca', '.', '[SEP]', 'e', 'outra', '!', '[SEP]', , '[PAD]']
```

```
max_length=12
```

tokens

Uma sentença. E outra!

attention_mask

[0 , 0 , 0 , 0 , 0 , 0, 0, 1, 1, 1, 1, 0]

token_type_ids

[2 , 335 , 12286 , 155 , 951 , 18, 3 , 37 , 929 , 5 , 3, 0]

input_ids

['[CLS]', 'uma', 'sente', '##n', '##ca', '.', '[SEP]', 'e', 'outra', '!', '[SEP]', , '[PAD]']

max_length=12

tokens

Uma sentença. E outra!

```
[ 1 , 1 , 1 , 1 , 1 , 1, 1, 1, 1, 1, 1, 0
]
```

attention_mask

```
[ 0 , 0 , 0 , 0 , 0 , 0, 0, 1, 1, 1, 1, 0 ]
```

token_type_ids

```
[ 2 , 335 , 12286 , 155 , 951 , 18, 3 , 37 , 929 , 5 , 3, 0 ]
```

input_ids

```
['[CLS]', 'uma', 'sente', '##n', '##ca', '.', '[SEP]', 'e', 'outra', '!', '[SEP]', , '[PAD]']
```

tokens

max_length=12

Uma sentença. E outra!

positional_encoding

[1 , 1 , 1 , 1 , 1 , 1, 1, 1, 1, 1, 1, 0
]

attention_mask

[0 , 0 , 0 , 0 , 0 , 0, 0, 1, 1, 1, 1, 0]

token_type_ids

[2 , 335 , 12286 , 155 , 951 , 18, 3 , 37 , 929 , 5 , 3, 0]

input_ids

['[CLS]', 'uma', 'sente', '##n', '##ca', '.', '[SEP]', 'e', 'outra', '!', '[SEP]', , '[PAD]']

tokens

max_length=12

Uma sentença. E outra!

```
[ 0 , 1 , 2 , 3 , 4 , 5, 6, 7, 8, 9, 10, 11
]
```

positional_encoding

```
[ 1 , 1 , 1 , 1 , 1 , 1, 1, 1, 1, 1, 1, 0
]
```

attention_mask

```
[ 0 , 0 , 0 , 0 , 0 , 0, 0, 1, 1, 1, 1, 0 ]
```

token_type_ids

```
[ 2 , 335 , 12286 , 155 , 951 , 18, 3 , 37 , 929 , 5 , 3, 0 ]
```

input_ids

```
['[CLS]', 'uma', 'sente', '##n', '##ca', '.', '[SEP]', 'e', 'outra', '!', '[SEP]', , '[PAD]']
```

tokens

```
max_length=12
```

Uma sentença. E outra!

https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

]

positional_encoding

```
[ 1 , 1 , 1 , 1 , 1 , 1, 1, 1, 1, 1, 1, 0
]
```

attention_mask

```
[ 0 , 0 , 0 , 0 , 0 , 0, 0, 1, 1, 1, 1, 0 ]
```

token_type_ids

```
[ 2 , 335 , 12286 , 155 , 951 , 18, 3 , 37 , 929 , 5 , 3, 0 ]
```

input_ids

```
['[CLS]', 'uma', 'sente', '##n', '##ca', '.', '[SEP]', 'e', 'outra', '!', '[SEP]', , '[PAD]']
```

max_length=12

tokens

Uma sentença. E outra!

transform_Layer_1

```
[ 0 , 1 , 2 , 3 , 4 , 5, 6, 7, 8, 9, 10, 11
]
```

positional_encoding

```
[ 1 , 1 , 1 , 1 , 1 , 1, 1, 1, 1, 1, 1, 0
]
```

attention_mask

```
[ 0 , 0 , 0 , 0 , 0 , 0, 0, 1, 1, 1, 1, 0 ]
```

token_type_ids

```
[ 2 , 335 , 12286 , 155 , 951 , 18, 3 , 37 , 929 , 5 , 3, 0 ]
```

input_ids

```
['[CLS]', 'uma', 'sente', '##n', '##ca', '.', '[SEP]', 'e', 'outra', '!', '[SEP]', , '[PAD]']
```

tokens

max_length=12

Uma sentença. E outra!

transform_Layer_12

.
.
.

transform_Layer_1

[0 , 1 , 2 , 3 , 4 , 5, 6, 7, 8, 9, 10, 11]

positional_encoding

[1 , 1 , 1 , 1 , 1 , 1, 1, 1, 1, 1, 1, 0]

attention_mask

[0 , 0 , 0 , 0 , 0 , 0, 0, 1, 1, 1, 1, 0]

token_type_ids

[2 , 335 , 12286 , 155 , 951 , 18, 3 , 37 , 929 , 5 , 3, 0]

input_ids

['[CLS]', 'uma', 'sente', '##n', '##ca', '.', '[SEP]', 'e', 'outra', '!', '[SEP]', , '[PAD]']

tokens

max_length=12

Uma sentença. E outra!

transform_Layer_12

.
.
.

transform_Layer_1

[0 , 1 , 2 , 3 , 4 , 5, 6, 7, 8, 9, 10, 11]

positional_encoding

[1 , 1 , 1 , 1 , 1 , 1, 1, 1, 1, 1, 1, 0]

attention_mask

[0 , 0 , 0 , 0 , 0 , 0, 0, 1, 1, 1, 1, 0]

token_type_ids

[2 , 335 , 12286 , 155 , 951 , 18, 3 , 37 , 929 , 5 , 3, 0]

input_ids

['[CLS]', 'uma', 'sente', '##n', '##ca', '.', '[SEP]', 'e', 'outra', '!', '[SEP]', , '[PAD]']

tokens

max_length=12

Uma sentença. E outra!

↑ transform_Layer_12

.
.
.

↑ transform_Layer_1

```
[ 0 , 1 , 2 , 3 , 4 , 5, 6, 7, 8, 9, 10, 11
]
```

↑ positional_encoding

```
[ 1 , 1 , 1 , 1 , 1 , 1, 1, 1, 1, 1, 1, 0
]
```

↑ attention_mask

```
[ 0 , 0 , 0 , 0 , 0 , 0, 0, 1, 1, 1, 1, 0 ]
```

↑ token_type_ids

```
[ 2 , 335 , 12286 , 155 , 951 , 18, 3 , 37 , 929 , 5 , 3, 0 ]
```

↑ input_ids

```
['[CLS]', 'uma', 'sente', '##n', '##ca', '.', '[SEP]', 'e', 'outra', '!', '[SEP]', , '[PAD]']
```

↑ tokens

max_length=12

Uma sentença. E outra!

transform_Layer_12

.
.
.

transform_Layer_1

[0 , 1 , 2 , 3 , 4 , 5, 6, 7, 8, 9, 10, 11]

positional_encoding

[1 , 1 , 1 , 1 , 1 , 1, 1, 1, 1, 1, 1, 0]

attention_mask

[0 , 0 , 0 , 0 , 0 , 0, 0, 1, 1, 1, 1, 0]

token_type_ids

[2 , 335 , 12286 , 155 , 951 , 18, 3 , 37 , 929 , 5 , 3, 0]

input_ids

['[CLS]', 'uma', 'sente', '##n', '##ca', '.', '[SEP]', 'e', 'outra', '!', '[SEP]', , '[PAD]']

tokens

max_length=12

Uma sentença. E outra!

transform_Layer_12

.
.
.

transform_Layer_1

[0 , 1 , 2 , 3 , 4 , 5, 6, 7, 8, 9, 10, 11]

positional_encoding

[1 , 1 , 1 , 1 , 1 , 1, 1, 1, 1, 1, 1, 0]

attention_mask

[0 , 0 , 0 , 0 , 0 , 0, 0, 1, 1, 1, 1, 0]

token_type_ids

[2 , 335 , 12286 , 155 , 951 , 18, 3 , 37 , 929 , 5 , 3, 0]

input_ids

['[CLS]', 'uma', 'sente', '##n', '##ca', '.', '[SEP]', 'e', 'outra', '!', '[SEP]', , '[PAD]']

tokens

max_length=12

Uma sentença. E outra!

Prediction

'[CLS]'

transform_Layer_12

transform_Layer_1

[0 , 1 , 2 , 3 , 4 , 5, 6, 7, 8, 9, 10, 11]

positional_encoding

[1 , 1 , 1 , 1 , 1 , 1, 1, 1, 1, 1, 1, 0]

attention_mask

[0 , 0 , 0 , 0 , 0 , 0, 0, 1, 1, 1, 1, 0]

token_type_ids

[2 , 335 , 12286 , 155 , 951 , 18, 3 , 37 , 929 , 5 , 3, 0]

input_ids

['[CLS]', 'uma', 'sente', '##n', '##ca', '.', '[SEP]', 'e', 'outra', '!', '[SEP]', , '[PAD]']

tokens

max_length=12

Uma sentença. E outra!

BERT - transfer learning

Pre training + Fine Tuning = Transfer Learning

$$$$ $

BERT - transfer learning

Pre training + Fine Tuning = Transfer Learning

$$$$ $

dia.org/wiki/GPT-2#:~:text=While%20the%20cost%20of%20training,cost%20cannot%20be%20estimated%20accurately.

zero-shot setting.

Training [edit]

Since the transformer architecture enabled massive parallelization, GPT-series models could be trained on larger corpora than previous NLP models. While the initial GPT model demonstrated that the approach was viable, GPT-2 would further explore the emergent properties of networks trained on extremely large corpora. *CommonCrawl*, a large corpus produced by web crawling and previously used in training NLP systems,[60] was considered due to its large size, but was rejected after further review revealed large amounts of unintelligible content.[8][60] Instead, OpenAI developed a new corpus, known as *WebText*; rather than scraping content indiscriminately from the World Wide Web, WebText was generated by scraping only pages linked to by Reddit posts that had received at least three upvotes prior to December 2017. The corpus was subsequently cleaned; HTML documents were parsed into plain text, duplicate pages were eliminated, and Wikipedia pages were removed (since their presence in many other datasets could have induced overfitting).[8]

While the cost of training GPT-2 is known to have been $256 per hour,[61][62] the amount of hours it took to complete training is unknown; therefore, the overall training cost cannot be estimated accurately.[63] However, comparable large language models using transformer architectures have had their costs documented in more detail; the training processes for BERT and XLNet consumed, respectively, $6,912 and $245,000 of resources.[62]

https://en.wikipedia.org/wiki/GPT-2#:~:text=While%20the%20cost%20of%20training,cost%20cannot%20be%20estimated%20accurately.

BERT - transfer learning

Pre training + Fine Tuning = Transfer Learning

$$$$ $

2 tasks durante o treinamento

1. Next Sentence Prediction (NSP)
2. Masked Language Model (MLM)

BERT - NSP

Pre training + Fine Tuning = Transfer Learning

$$$$ $

1. Next Sentence Prediction (NSP)
2. Masked Language Model (MLM)

NSP: O BERT é alimentado com pares de sentenças. Metade das vezes a 2a sentença segue imediatamente a primeira e a outra metade não.

BERT - NSP

Pre training + Fine Tuning = Transfer Learning

$$$$ $

1. Next Sentence Prediction (NSP)
2. Masked Language Model (MLM)

NSP: O BERT é alimentado com pares de sentenças. Metade das vezes a 2a sentença segue imediatamente a primeira e a outra metade não.

"Help me, Obi-Wan Kenobi. You're my only hope."

BERT - NSP

Pre training + Fine Tuning = Transfer Learning

$$$$ $

1. Next Sentence Prediction (NSP)
2. Masked Language Model (MLM)

NSP: O BERT é alimentado com pares de sentenças. Metade das vezes a 2a sentença segue imediatamente a primeira e a outra metade não.

"Help me, Obi-Wan Kenobi. You're my only hope."

É NS?

BERT - MLM

Pre training + Fine Tuning = Transfer Learning

$$$$ $

MLM: Por que precisa de [MASK] 😒?

1. Next Sentence Prediction (NSP)
2. Masked Language Model (MLM)

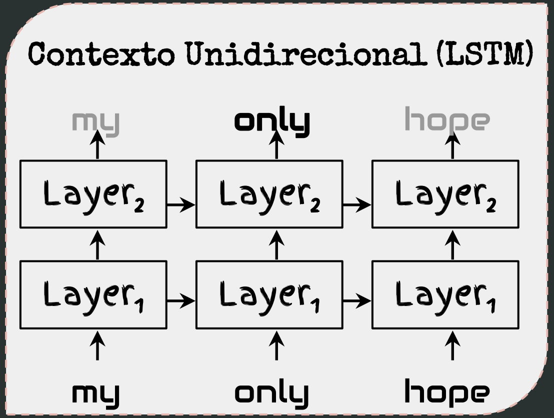
BERT - MLM

Pre training + Fine Tuning = Transfer Learning

$$$$ $

1. Next Sentence Prediction (NSP)
2. Masked Language Model (MLM)

MLM: Por que precisa de [MASK] 😐?



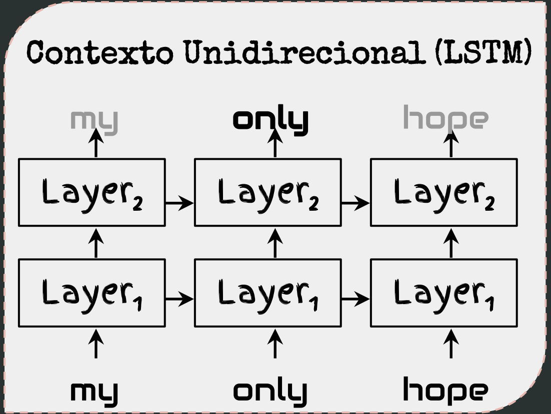
BERT - MLM

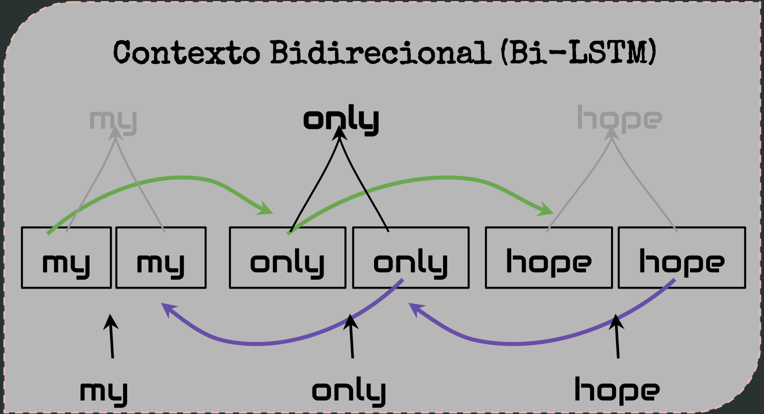
Pre training + Fine Tuning = Transfer Learning

$$$$ $

1. Next Sentence Prediction (NSP)
2. Masked Language Model (MLM)

MLM: Por que precisa de [MASK] 😉?





BERT - MLM

Pre training + Fine Tuning = Transfer Learning

$$$$ $

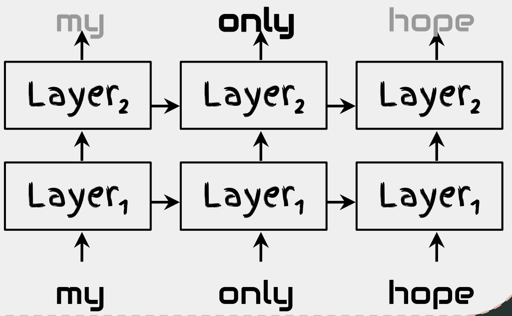
1. Next Sentence Prediction (NSP)
2. Masked Language Model (MLM)

MLM: Por que precisa de [MASK] 🤓?

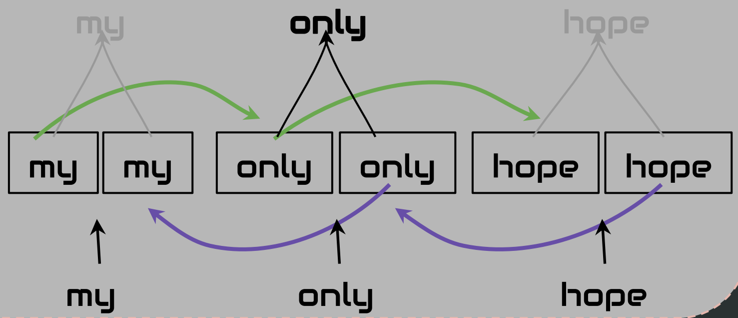
BERT

BIDIRECIONAL

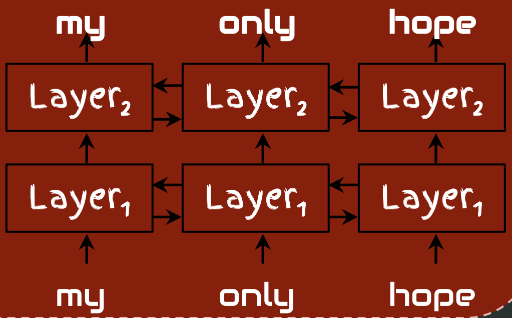
Contexto Unidirecional (LSTM)



Contexto Bidirecional (Bi-LSTM)



Contexto BIDIRECIONAL



BERT - MLM

Pre training + Fine Tuning = Transfer Learning

$$$$ $

1. Next Sentence Prediction (NSP)
2. Masked Language Model (MLM)

MLM: Ao todo 15% do dataset é transformado com MLM que possui 3 formas:

80% de 15%: "[MASK] me, Obi-Wan Kenobi. You're my only hope."

BERT - MLM

Pre training + Fine Tuning = Transfer Learning

$$$$ $

MLM: Ao todo 15% do dataset é transformado com MLM que possui 3 formas:

1. Next Sentence Prediction (NSP)
2. Masked Language Model (MLM)

80% de 15%: "[MASK] me, Obi-Wan Kenobi. You're my only hope."

10% de 15%: "Find me, Obi-Wan Kenobi. You're my only hope."

BERT - MLM

Pre training + Fine Tuning = Transfer Learning

$$$$ $

1. Next Sentence Prediction (NSP)
2. Masked Language Model (MLM)

MLM: Ao todo 15% do dataset é transformado com MLM que possui 3 formas:

80% de 15%: "[MASK] me, Obi-Wan Kenobi. You're my only hope."

10% de 15%: "Find me, Obi-Wan Kenobi. You're my only hope."

10% de 15%: "Help me, Obi-Wan Kenobi. You're my only hope."

BERT - Pre training

Pre training + Fine Tuning = Transfer Learning

$$$$ $

Uma instância de treino consiste de um pedaço de texto com **MLM** + **NSP**. As predições de **MLM** e **NSP** são combinadas em uma única loss. De acordo com o artigo, apêndice A.2: *"The training loss is the sum of the mean masked LM likelihood and the mean next sentence prediction likelihood."*

BERT - Pre training

Pre training + Fine Tuning = Transfer Learning

$$$$ $

Uma instância de treino consiste de um pedaço de texto com **MLM** + **NSP**. As predições de **MLM** e **NSP** são combinadas em uma única loss. De acordo com o artigo, apêndice A.2: *"The training loss is the sum of the mean masked LM likelihood and the mean next sentence prediction likelihood."*

Exemplo de uma instância de treino

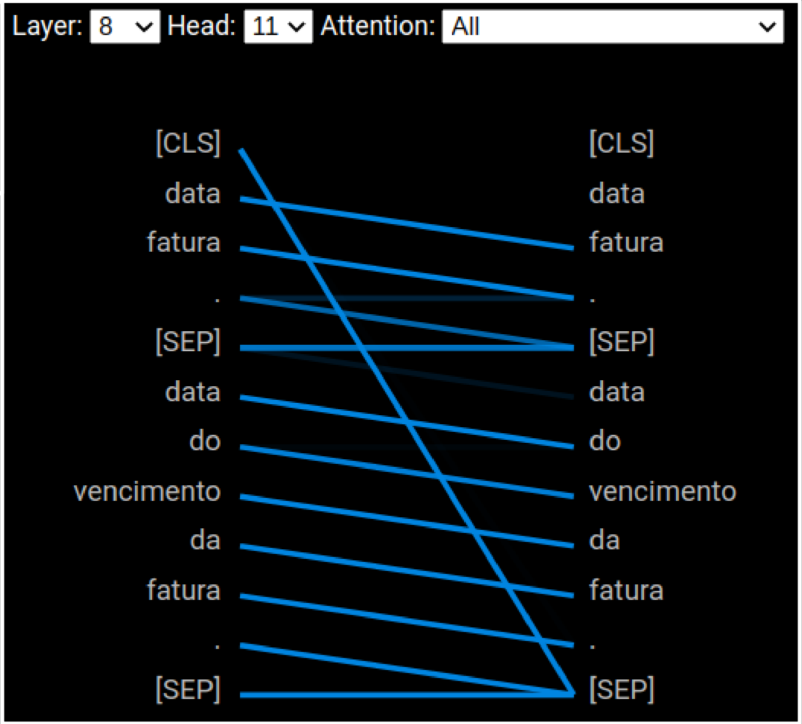
Help me, Obi-Wan Kenobi. You're my unique [MASK].

Label: IsNext

BERT - Rorschach Test

- sentença 1: Data fatura.
- sentença 2: Data do vencimento da fatura.

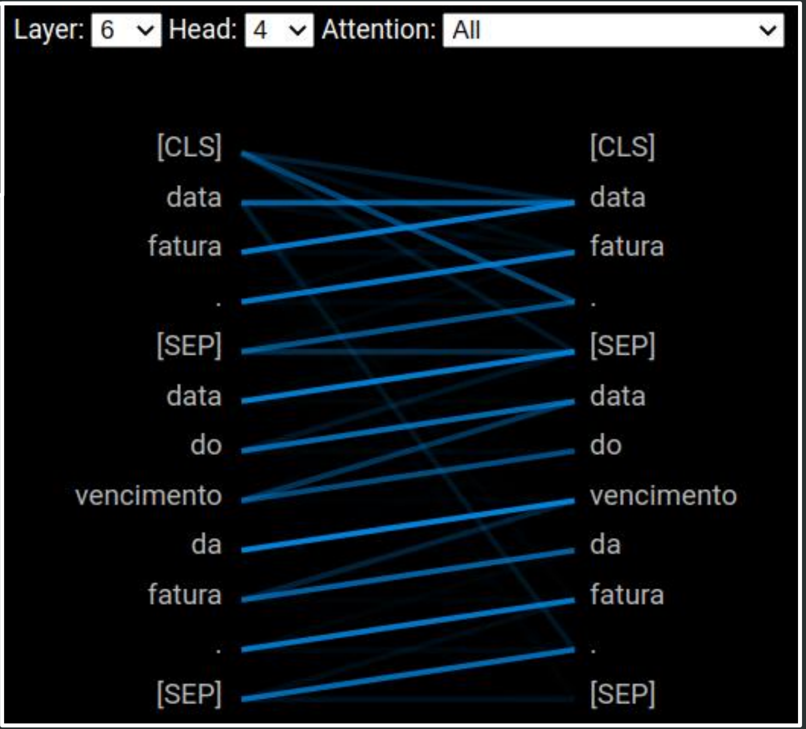
Next Word Attention



BERT - Rorschach Test

- sentença 1: Data fatura.
- sentença 2: Data do vencimento da fatura.

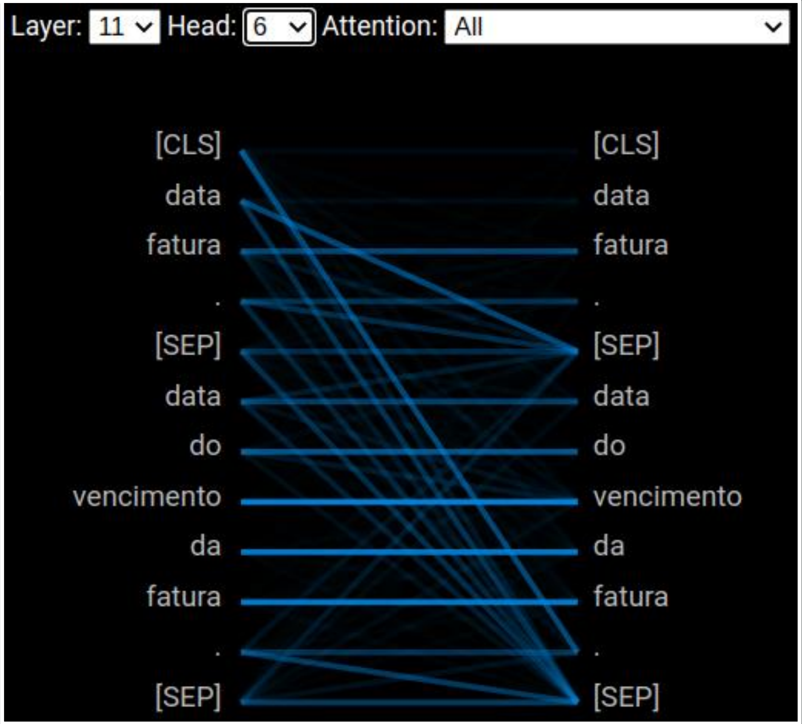
Previous Word Attention



BERT - Rorschach Test

- sentença 1: Data fatura.
- sentença 2: Data do vencimento da fatura.

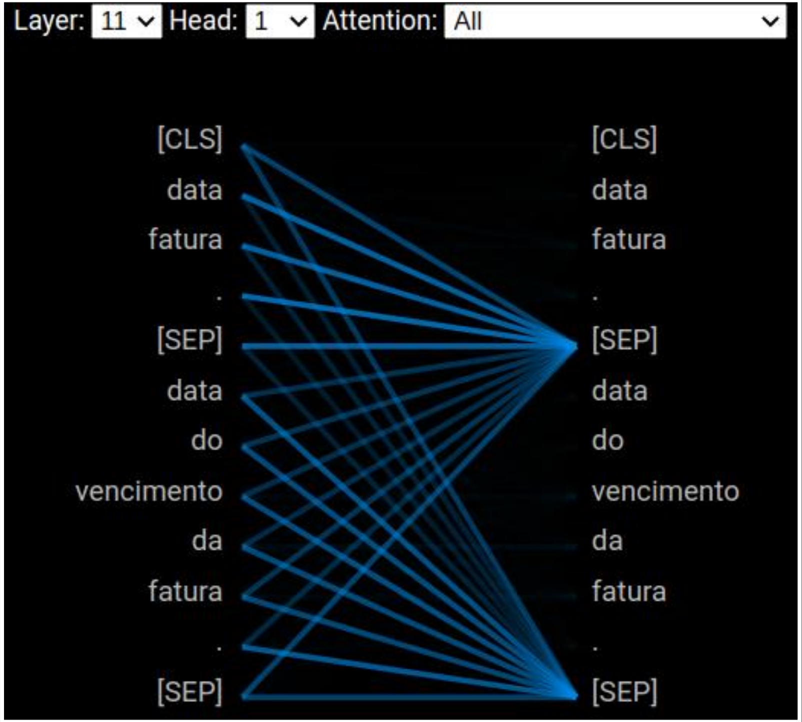
Identical Words Attention



BERT - Rorschach Test

- sentença 1: Data fatura.
- sentença 2: Data do vencimento da fatura.

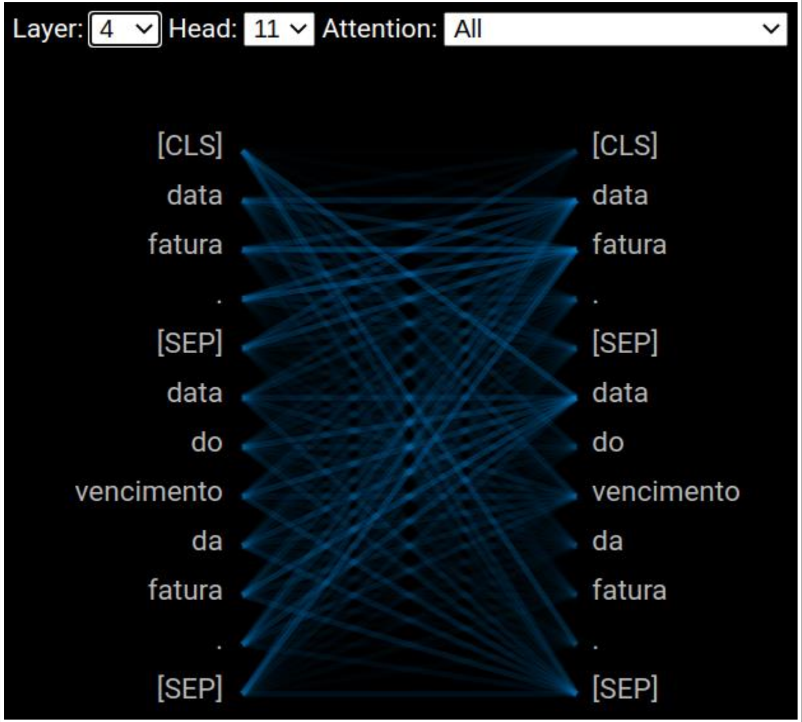
[SEP]'s Attention

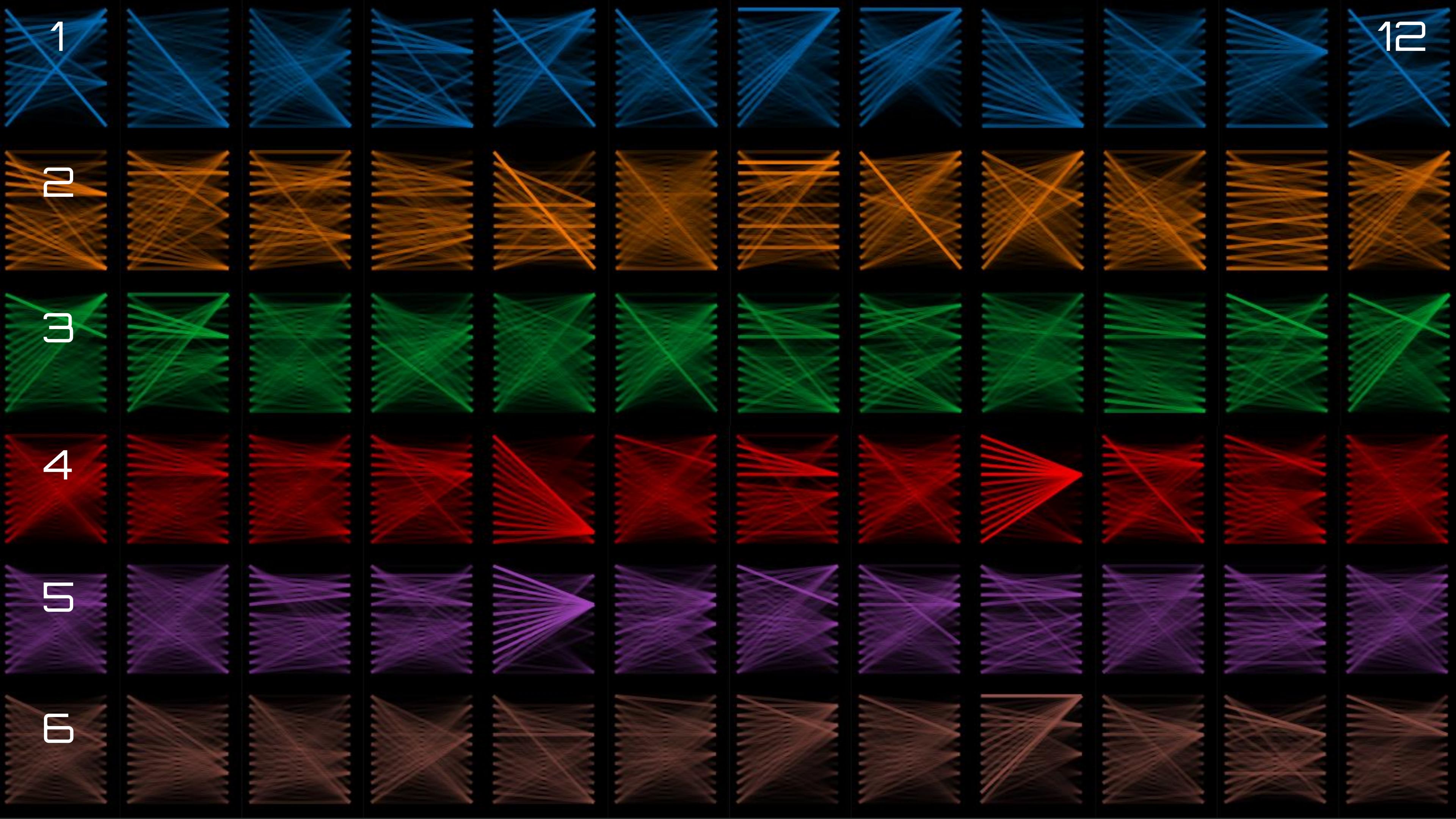


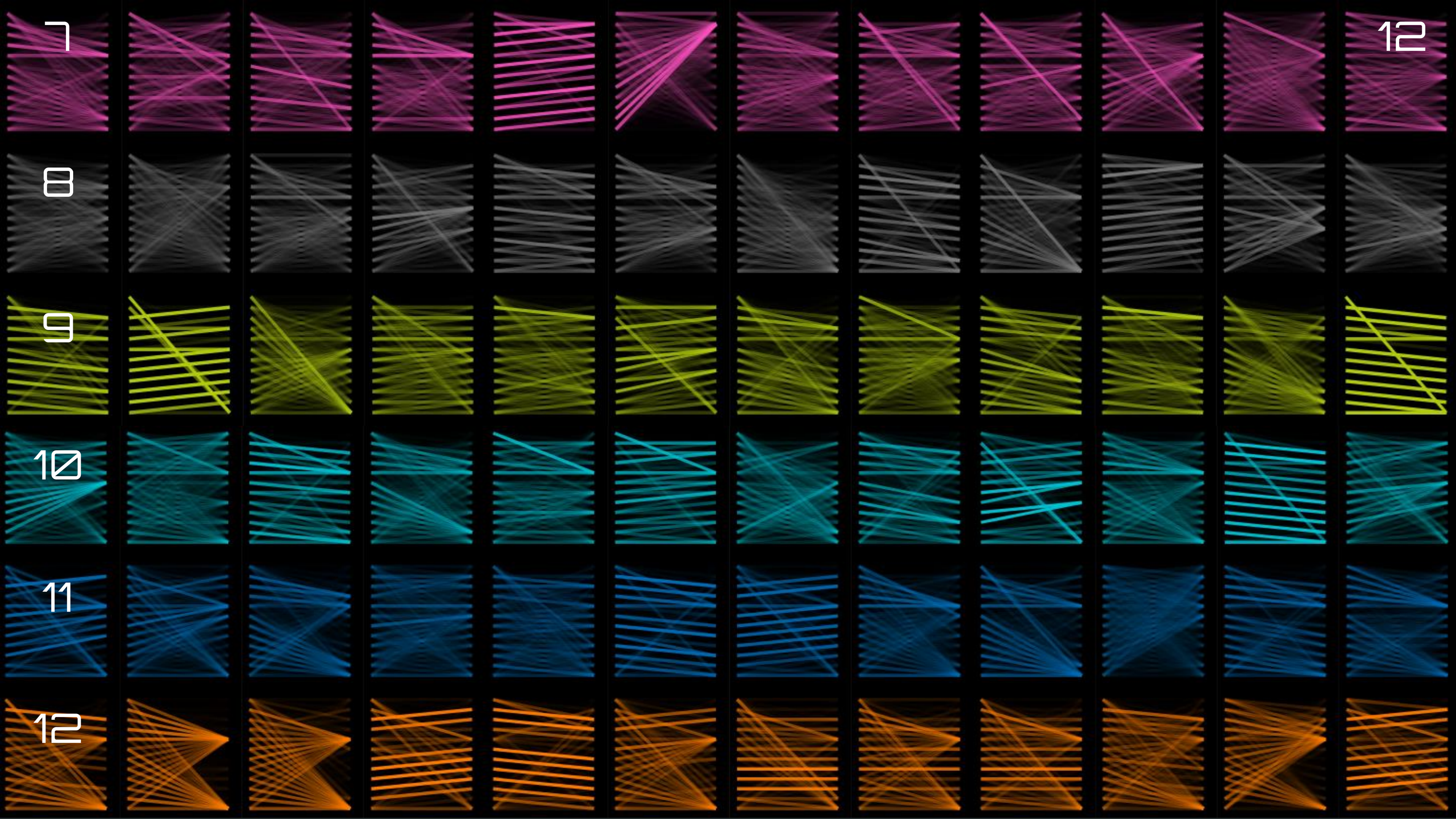
BERT - Rorschach Test

- sentença 1: Data fatura.
- sentença 2: Data do vencimento da fatura.

BoW
Attention




1
12
2
3
4
5
6


7
12
8
9
10
11
12

BERT - Attention Viz in SA

Positive Sample

Legend: Negative Neutral Positive

True Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
Positivo	Positivo (0.92)	Positivo	1.66	[CLS] atendimento muito bom , problema foi resolvido ! [SEP]

```
[('[CLS]', 0.0),('atendimento', 0.09), ('muito', 0.57),('bom',0.70), (',', 0.03),
('problema', -0.07), ('foi', 0.01), ('resolvido', -0.06), ('!', 0.38),('[SEP]', 0.0)]
```

Negative Sample

Legend: Negative Neutral Positive

True Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
Negativo	Negativo (0.03)	Neutro	-1.94	[CLS] quero cancelar , muito ruim , pessimo , nao resolveu meu problema . [SEP]

```
[('[CLS]', 0.0), ('quero', 0.10),('cancelar',0.039),(',', 0.03), ('muito', -0.19),
 ('ruim', -0.18), (',', 0.001), ('pessimo', -0.83), (',', 0.01), ('nao', -0.19),
('resolveu', -0.29), ('meu', -0.02),('problema', -0.10),('.', -0.30),('[SEP]',0.0)]
```

Referências

1. Artigo BERT: arxiv
2. BERT Viz: github-repo
3. Transformers Interpret: github-repo
4. Artigo Attention is all you need: arxiv
5. Self Attention explained: DeepMind-youtube channel
6. Artigo (livro?) de IR com Transformers: arxiv 155 pages
7. UvA DLC (tutorial 6): github page

Thanks !

Vinicius Fernandes Caridá

vfcarida@gmail.com

@vinicius caridá
@vfcarida
@vinicius caridá
@vfcarida
@vinicius caridá
@vfcarida