# Sentiment Analysis

Aiko Ramalho 1820892
Gabriel MacGregor 1820591

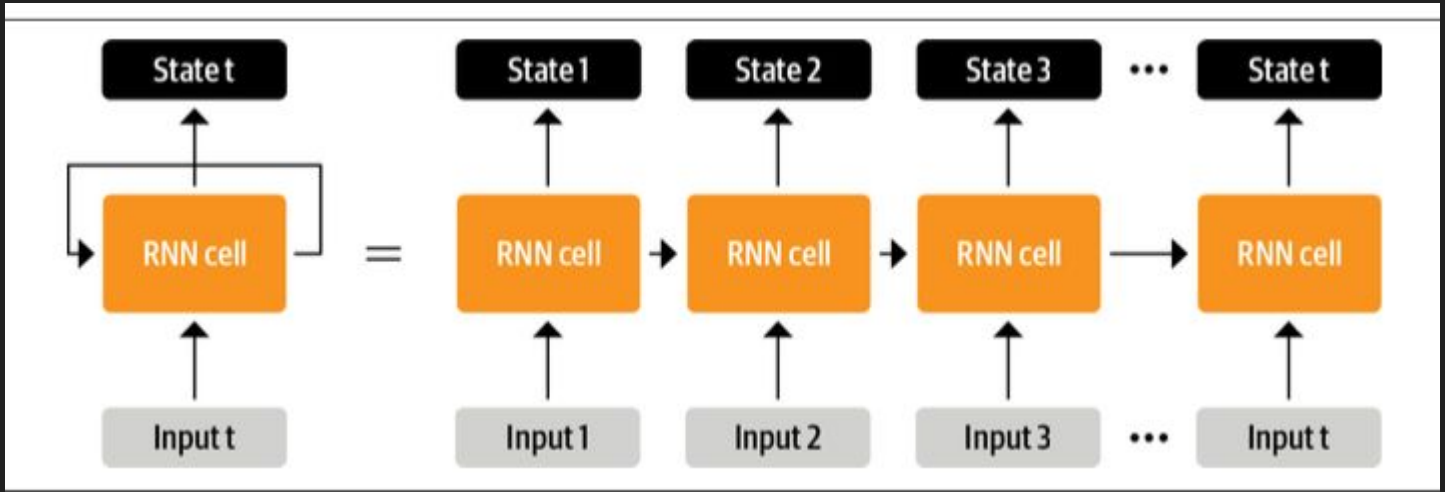
# Um breve overview sobre Transformers

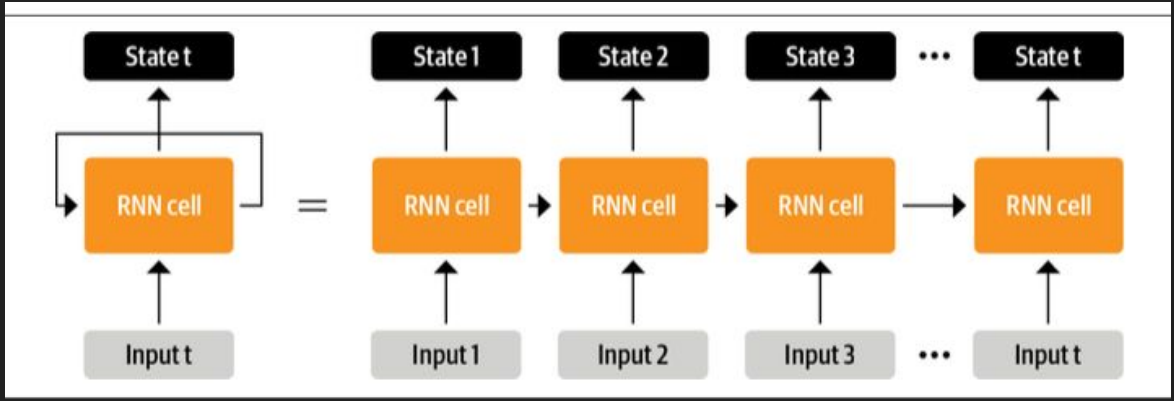

Em 2017, alguns pesquisadores do Google publicaram um artigo que propunha uma nova arquitetura de redes neurais.

Em paralelo, um método muito efetivo de transfer learning chamado ULMFiT mostrava que treinar LSTM networks em um corpus muito grande e diverso poderia produzir classificadores de texto muito performáticos com pouco dados classificados

# O Framework Encoder-Decoder

Antes dos Transformers, arquiteturas recorrentes como LSTMs eram as mais avançadas dentro da NLP. Essas arquiteturas implementavam um loop de feedback nas conexões da rede que permitiam que a informação se propagasse de um passo para o outro, tornando-as ideais para modelagem de dados sequenciais como texto.

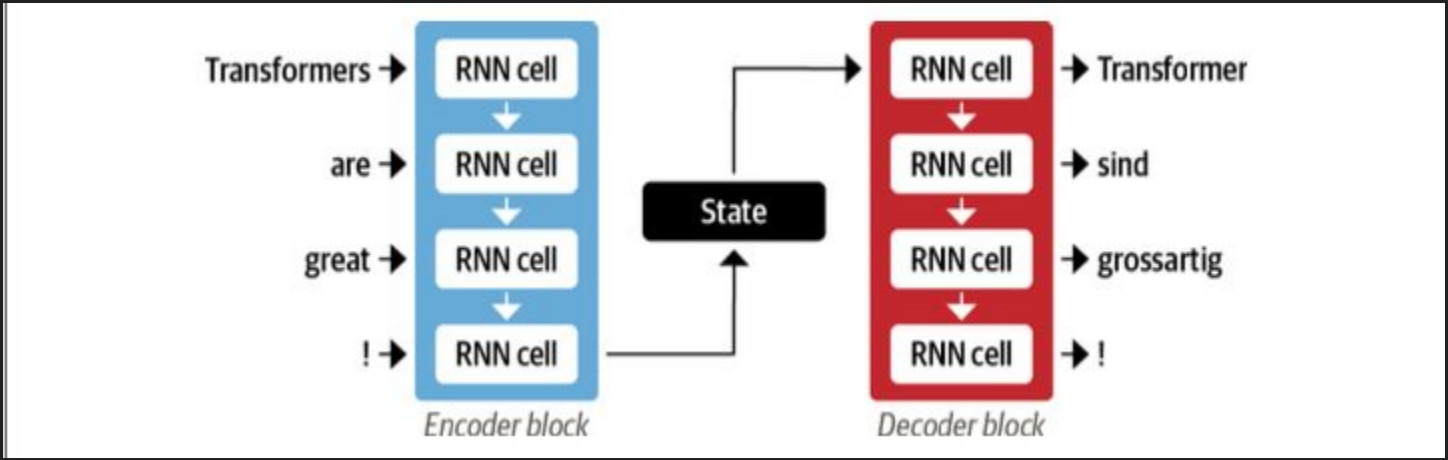





As RNNs tinham um papel importante em aplicações de tradução de texto, onde elas implementaram uma arquitetura encoder-decoder. O trabalho do encoder é fazer o encoding da informação para uma representação numérica que é chamada de *last hidden state*. O state é passado para o decoder que gera a sequência de output.

Em geral, encoder-decoder pode ser qualquer tipo de arquitetura de NN que consiga modelar sequências.

# Information Bottleneck



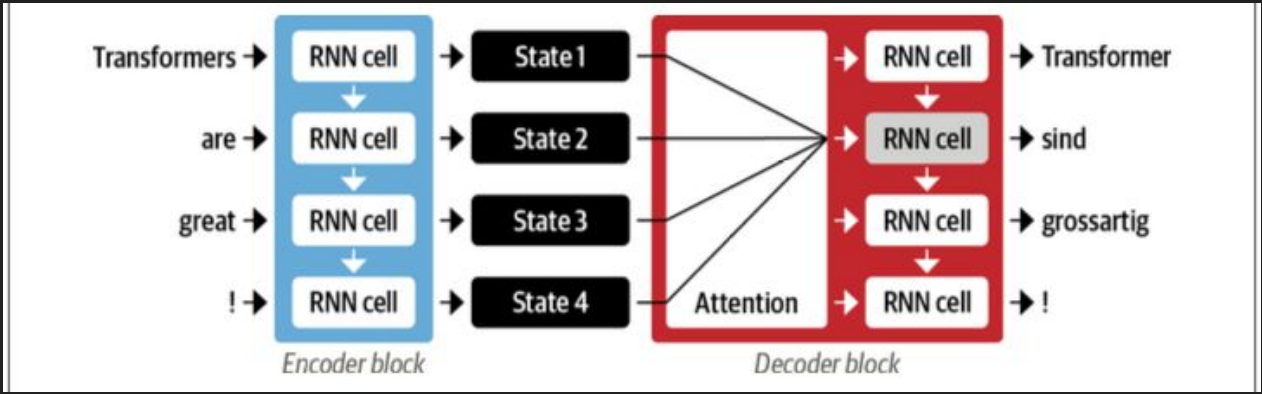

Uma fraqueza dessa arquitetura é que o hidden state final do encoder cria uma information bottleneck pois tem de representar o significado da sequência de input inteira, já que isso é tudo que o decoder tem acesso quando gera o output.

Felizmente, existe uma solução para esse problema, que é permitir que o decoder tenha acesso a todos os hidden states do encoder. O nome desse mecanismo é attention.

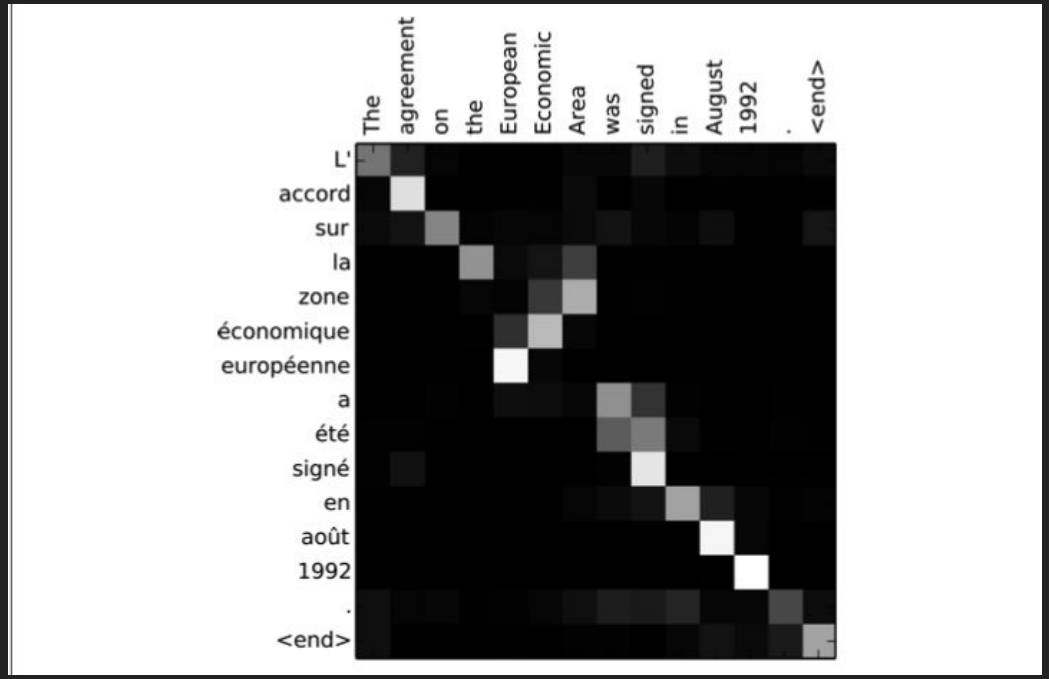
# Attention Mechanisms

A principal ideia desse mecanismo é produzir um hidden state para cada passo permitindo que o decoder tenha acesso a ele.

Entretanto, usar todos esses states ao mesmo tempo iria criar um input enorme para o decoder, então precisamos de algum mecanismo para priorizar quais states usar no processo

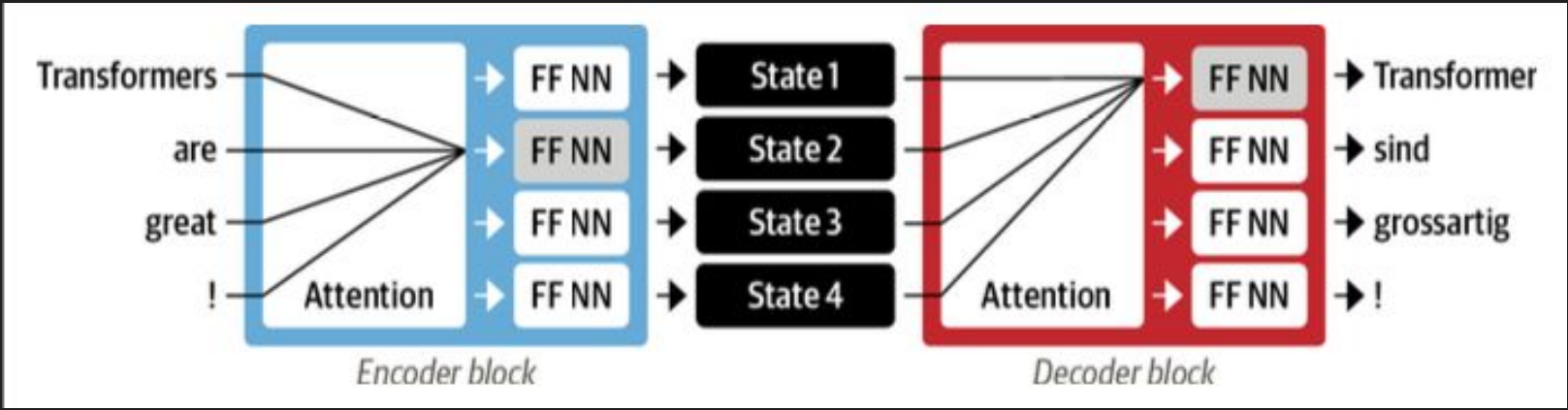


Por focar em que input tokens são mais relevantes para cada passo, os modelos baseados em attention são capazes de aprender relações não triviais entre as palavras na tradução e no texto original.

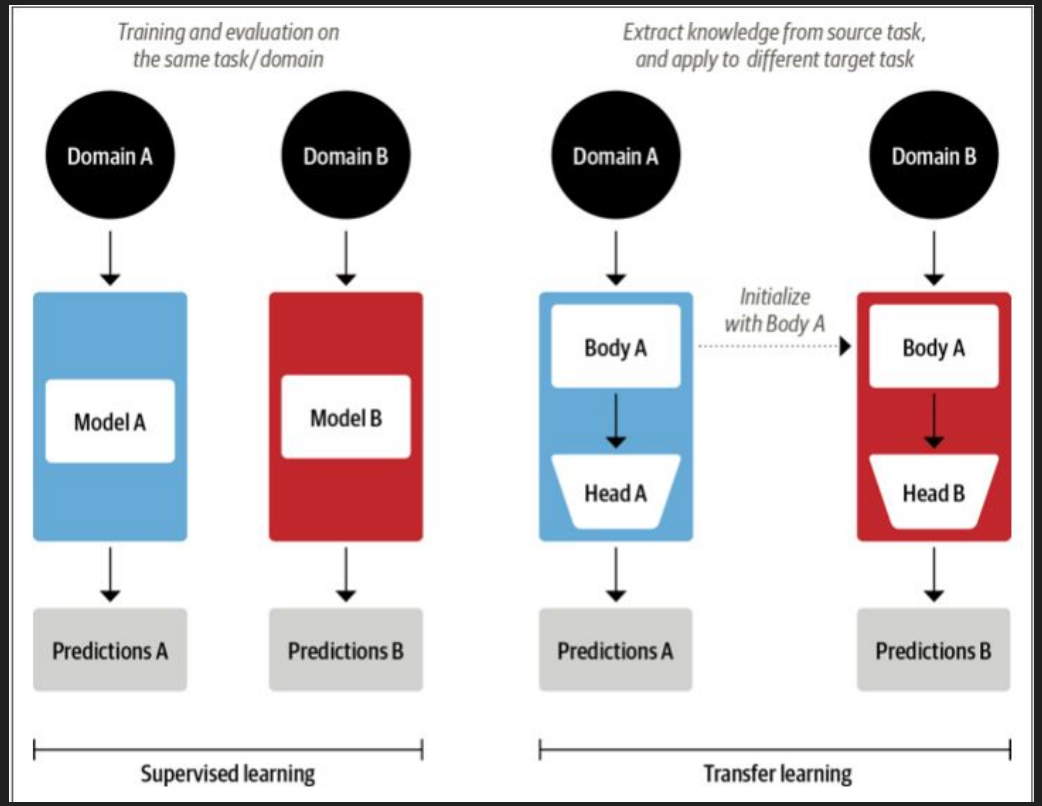


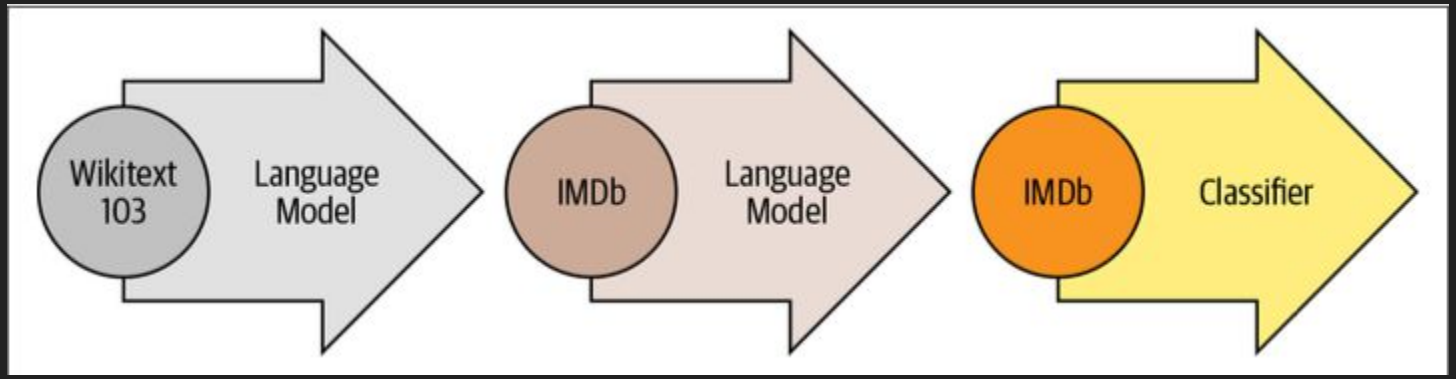
Porém ainda restava um problema: As computações eram sequenciais e não podiam ser paralelizadas durante a sequência de inputs.

Um novo modelo foi então introduzido: *self-attention*. A ideia é que esse mecanismo permite que a attention opere na mesma camada da NN.



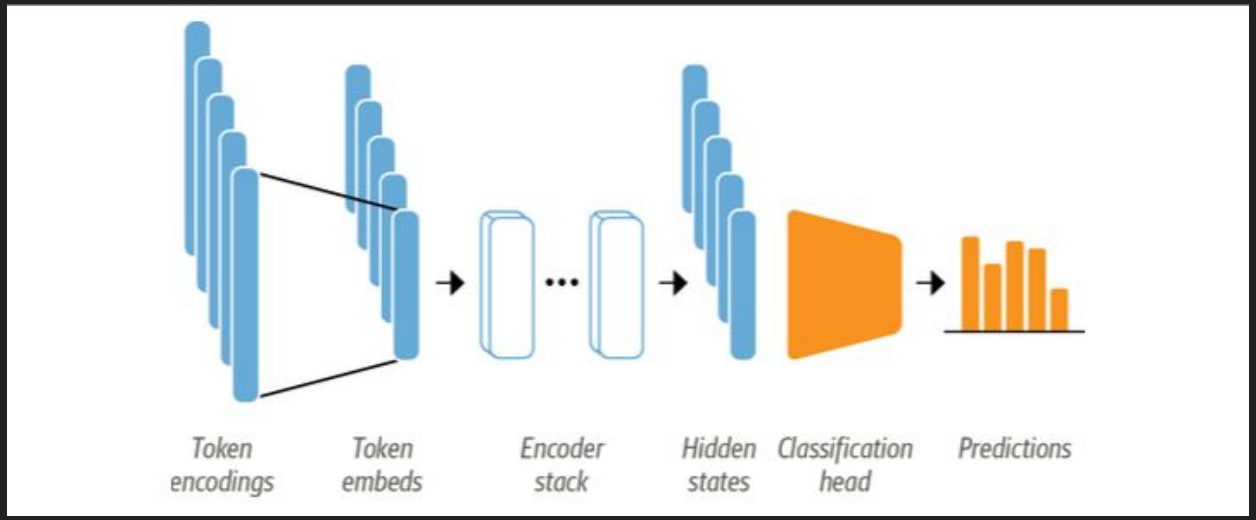
# Transfer Learning






- Pretraining -> predizer a proxima palavra baseada nas anterioes
- Domain Adaptation -> predizer proxima palavra no target corpus
- Fine-tuning -> modelo é ajustado com um classification layer para a tarefa

# Treinando um classificador de texto




Primeiro, o texto é tokenizado e representado como one-hot vectors chamados token encodings. Depois, os encodings são convertidos para embeddings, que são são vetores que moram num espaço dimensional mais reduzido. Então os embeddings são passados para as camadas de encoder block para emitir um hidden state para cada input token.

P/ pretraining, hidden state é passado para uma camada que prediz o masked input token.

P/ classification, trocamos a camada de language modeling por uma camada de classificação.

# Treinando um classificador de texto

Temos duas opções:

- Feature Extraction -> usamos os hidden states como features e treinamos um classificador em cima deles, sem modificar o modelo pretreinado
- Fine-tuning -> Treinamos todo o modelo ponta a ponta, o que também atualiza os parâmetros do modelo pré treinado.

# Bibliografia

-Natural Language Processing with Transformers, Revised Edition

- Attention is All You Need

- Knowledge Distilation [Bucila et al., 2006, Hinton et al., 2015]

# Próximos passos

- Montar um dataset real -> Scraping twitter?
- NEP on tweets
- Adaptar o modelo para português (modelo pré treinado em dataset pt-br? Multilingual transformers?)