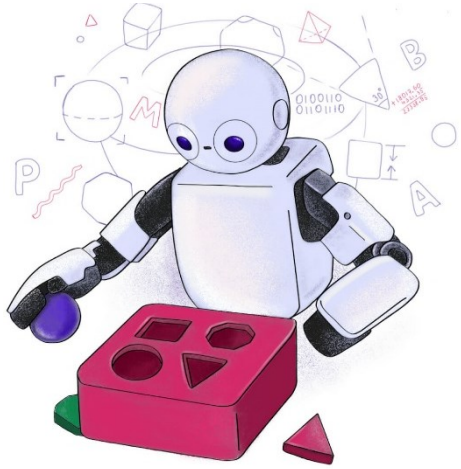
TP558 - Tópicos avançados em Machine Learning:

Automatic Speech Recognition with Transformer



Inatel

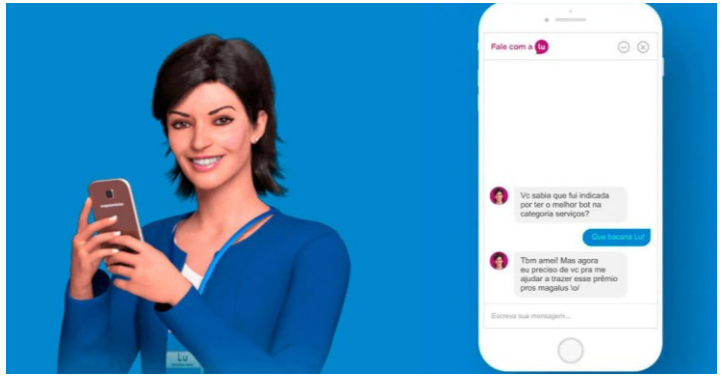
Georgino da Silva Baltazar
georgino.baltazar@dtel.inatel.br

Introdução

- O **reconhecimento automático de fala** (**ASR-*Automatic Speech Recognition***, na sigla em inglês) é a tecnologia que permite que computadores e dispositivos eletrônicos ***transcrevam a voz humana em texto de forma automática.***

Introdução

- Ela tem desempenhado um papel crucial em diversas aplicações modernas, desde assistentes virtuais, como Alexa e Siri, até sistemas de transcrição de áudio e legendagem automática de vídeos.



Introdução

- A capacidade de ***máquinas*** ***entenderem*** e ***transcreverem*** a ***fala humana*** com precisão **abre portas para uma interação mais natural e eficiente entre humanos e computadores**.

Fundamentação teórica

Evolução dos Sistemas de Reconhecimento de Fala

Data	Sistema de Reconhecimento de fala	Descrição
Décadas de 1950-1980	***Coincidência de padrões***	Os sons eram comparados com um conjunto pré-definido de padrões armazenados.
Na década de 1970	***Modelos de Markov Ocultos (HMM)***	Foram introduzidos e se tornaram a base dos sistemas do ASR.
Décadas de 1980-2000	***Modelos Estatísticos Avançados***	Os ***HMMs*** foram combinados **com técnicas de extração de características**, como a *Transformada de Fourier* e a *Transformada de Coseno Discreta*, para melhorar a precisão.
Décadas de 2010	***Redes Neurais e Aprendizado Profundo***	***Redes Neurais Profundas (DNNs):*** A introdução das DNNs trouxe um avanço significativo ao campo do ASR. As DNNs são capazes de modelar características complexas dos dados de fala, superando os HMMs em muitas tarefas.

Fundamentação teórica

Evolução dos Sistemas de Reconhecimento de Fala

Data	Sistema de Reconhecimento de fala	Descrição
Décadas de 2010	***Redes Neurais e Aprendizado Profundo***	***Redes Neurais Recorrentes (RNNs) e Long Short-Term Memory (LSTM):*** As RNNs e LSTMs foram utilizadas para modelar dependências temporais de longo alcance, melhorando a capacidade dos sistemas de reconhecer palavras em contextos variados.
2017-Presente	***Transformers***	Utilizam ***mecanismos de atenção*** que permitem a modelagem eficiente de dependências de longo alcance em dados sequenciais, sem a necessidade de processamento sequencial típico das RNNs.

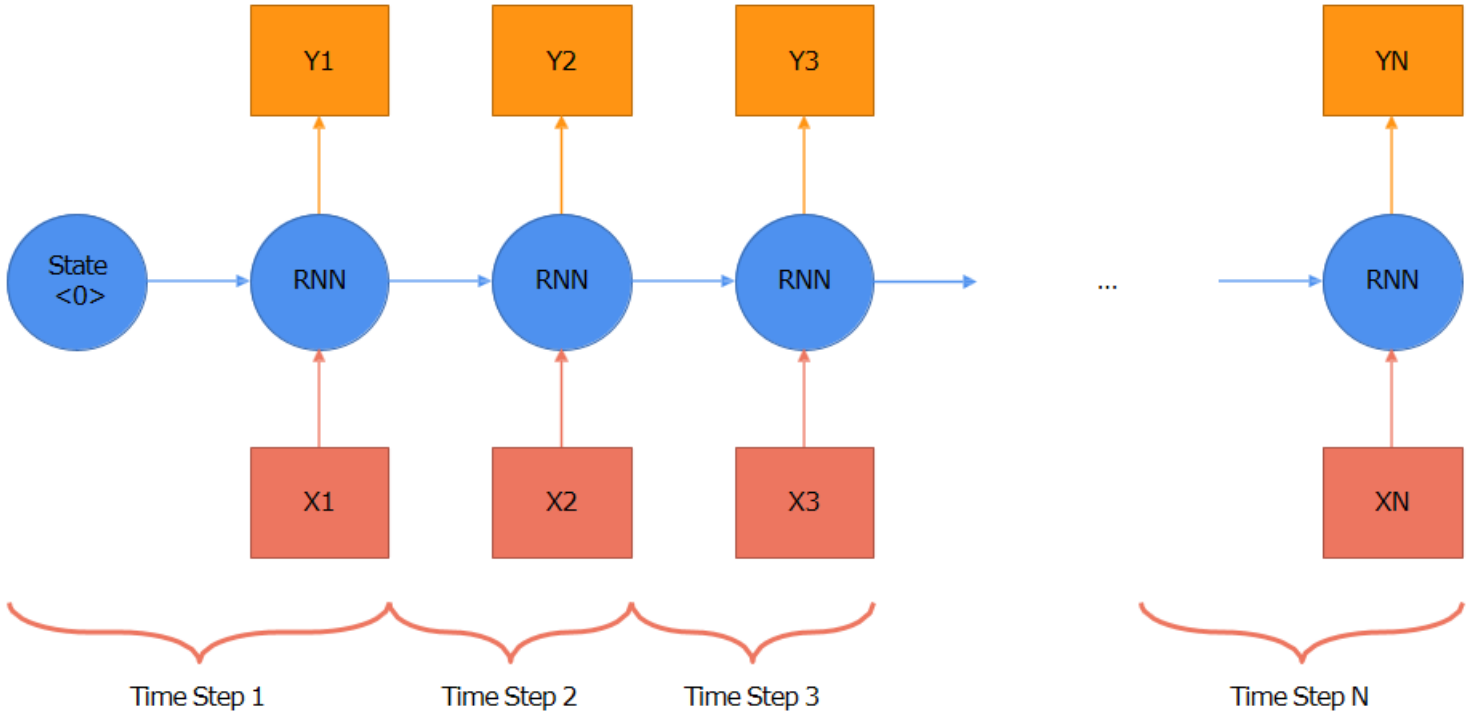
Fundamentação teórica

Evolução dos Sistemas de Reconhecimento de Fala

- A *evolução* dos *sistemas de reconhecimento de fala* reflete uma *progressão contínua* de *métodos estatísticos simples* para *modelos sofisticados de aprendizado profundo*.
- A introdução dos *Transformers* marcou uma mudança paradigmática, proporcionando melhorias substanciais na precisão e eficiência dos sistemas de ASR.
- Hoje, esses modelos continuam a ser refinados e ampliados, impulsionando avanços na interação homem-máquina e na acessibilidade tecnológica

Fundamentação teórica

Redes Neurais Recorrentes



Fundamentação teórica

Problemas com RNN (entre outros)

- Computação lenta para sequências longas
- Dificuldade em acessar informações de muito tempo atrás

Transformers

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

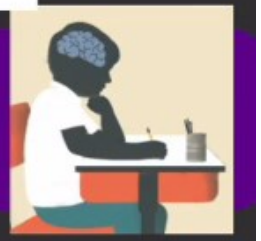
Llion Jones* Google Research llion@google.com

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

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Mecanismos de Atenção



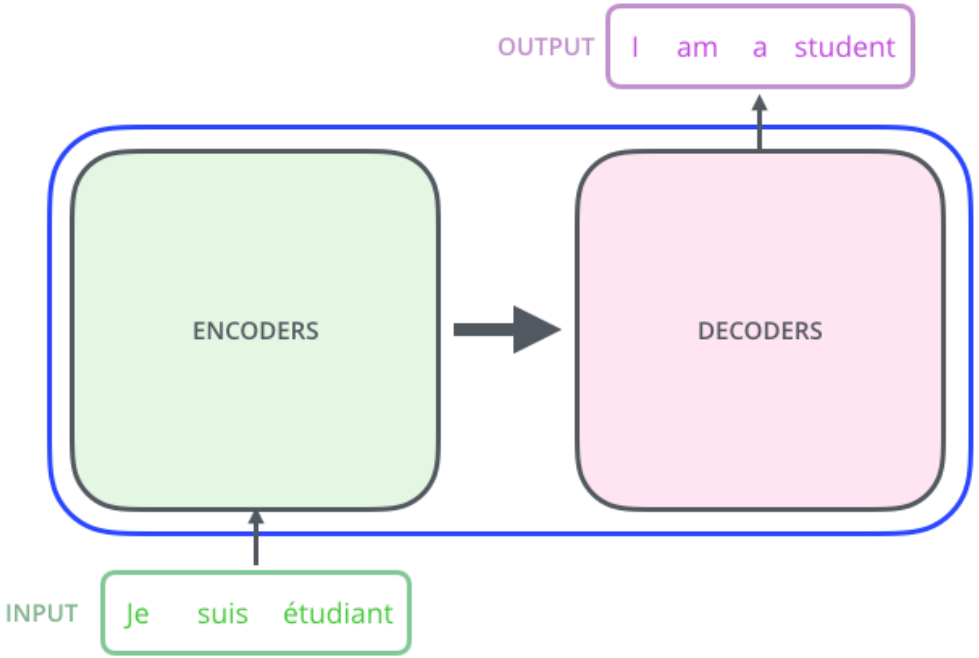
Originalmente sequence-to-sequence

+ 122 mil citações

Arquitetura e funcionamento

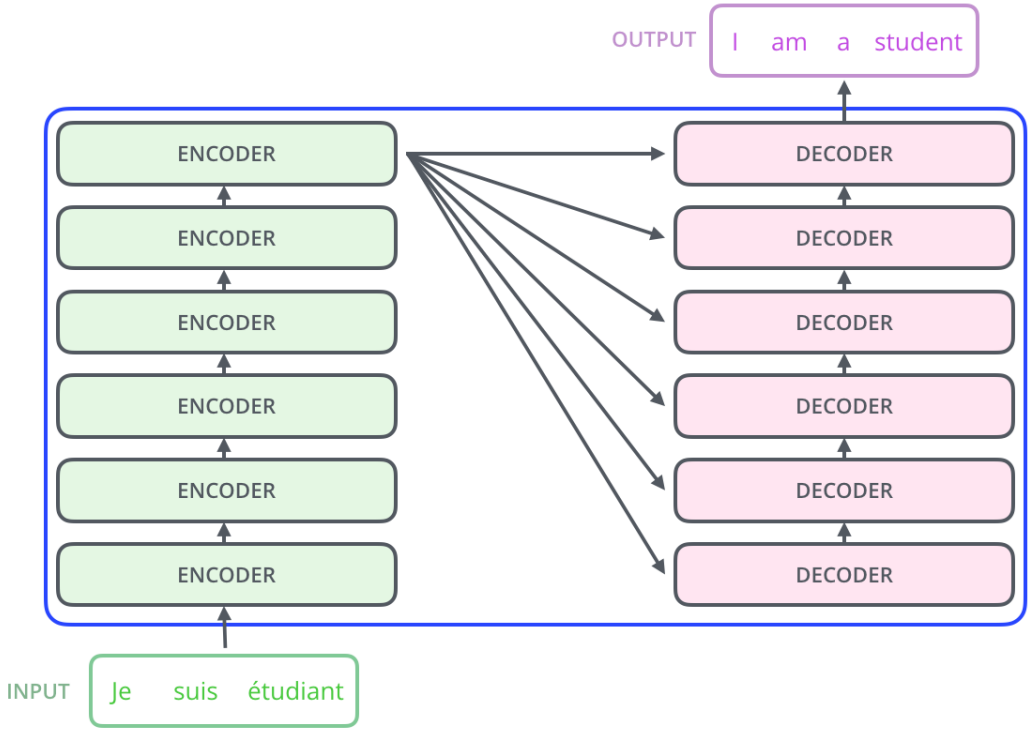
Os **Transformers** são uma arquitetura de modelo de aprendizado profundo usada principalmente para tarefas de processamento de linguagem natural (NLP), como tradução automática, resumo de texto e reconhecimento de fala.

Arquitetura e funcionamento



O ***Transformer*** é composto por dois componentes principais: o ***codificador(encoder)*** e o ***decodificador (decoder)***.

Arquitetura e funcionamento

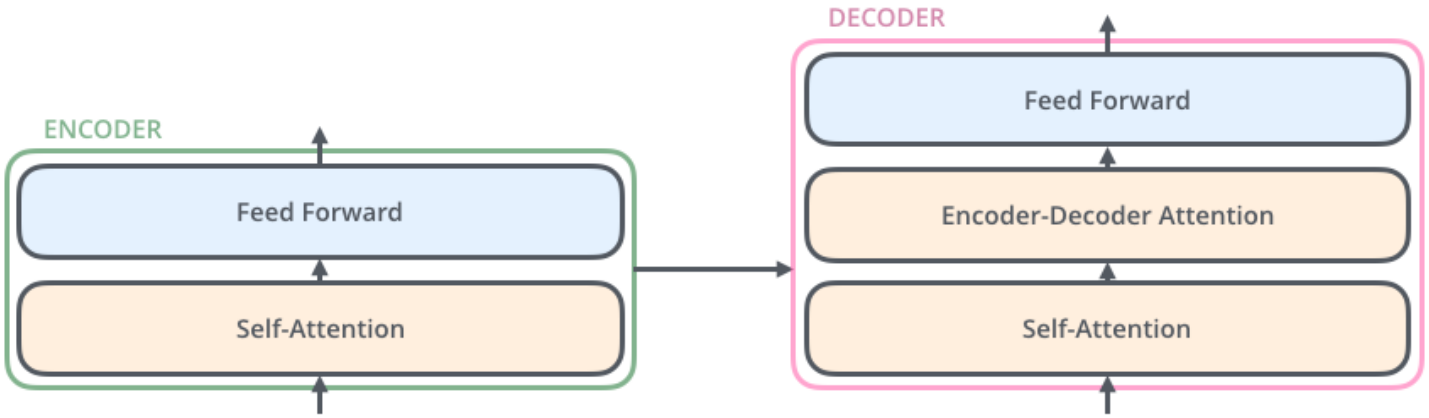


O componente de codificação é *uma pilha de codificadores*.

O componente de decodificação é uma pilha de decodificadores do mesmo número do codificador.

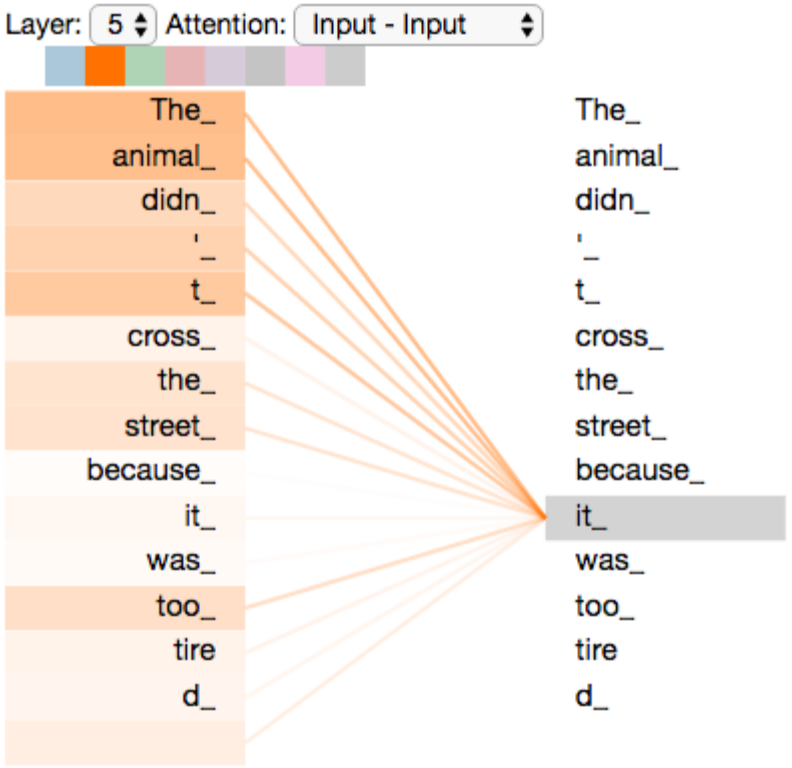
Arquitetura e funcionamento

- Os *codificadores são todos idênticos em estrutura* (embora não compartilhem pesos). Cada uma é dividida em duas subcamadas principais. *O decodificador possui ambas as camadas*, mas entre elas há uma camada de atenção que ajuda o decodificador a se concentrar nas partes relevantes da frase de entrada.



Arquitetura e funcionamento

Intuição por Trás do Self-Attention



- ***Permite que cada palavra em uma sequência de entrada considere todas as outras palavras na sequência ao calcular suas representações.***
- Este processo é crucial para capturar dependências contextuais e entender melhor o significado das palavras no contexto da frase completa.

Arquitetura e funcionamento

Funcionamento do Self-Attention

1. Cálculo de Query, Key e Value:

- *Para cada palavra na sequência de entrada, são gerados três vetores:* ***Query (Q),*** ***Key (K)*** *e* ***Value (V),*** *através de matrizes de projeção aprendidas.*

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V$$

- X é a matriz de embeddings da sequência de entrada.

- W^Q , W^K e W^V são matrizes de pesos aprendidas

Arquitetura e funcionamento

Funcionamento do Self-Attention

2. Cálculo das Similaridades:

- *Calcula-se a similaridade entre cada par de palavras multiplicando os vetores de **Query (Q)** pelos vetores de **Key (K).***

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

- Escalonamento: Divide-se pela raiz quadrada da dimensão dos vetores de Key para estabilizar os gradientes.

- Softmax: Normaliza as pontuações para que somem 1, convertendo-as em probabilidades

Arquitetura e funcionamento

Funcionamento do Self-Attention

3. Ponderação dos Valores:

- *Usam-se as pontuações de atenção para ponderar os vetores de **Value (V)**.*
- *Isso significa que as palavras mais relevantes têm maior influência na representação final da palavra atual.*

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

Arquitetura e funcionamento

- Exemplo Simples

1. Cálculo dos vetores

- *Vamos considerar a frase "**O gato está no tapete**":*

- Query (Q), Key (K) e Value (V) são gerados para cada palavra.

-"O" → Q1, K1, V1

-"gato" → Q2, K2, V1

-"está" → Q3, K3, V3

-"no" → Q4, K4, V4

-"tapete" → Q5, K5, V5

2. Cálculo das Similaridades e Ponderação:

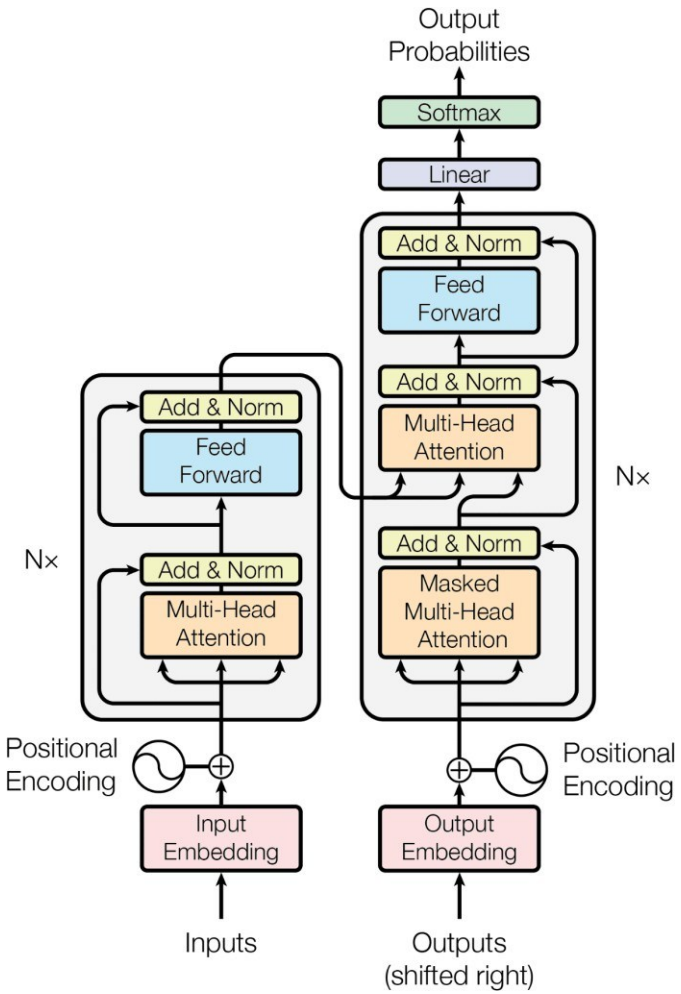
- Multiplica-se Q1 por todos os K para calcular a atenção de "O" em relação a todas as palavras.

- Aplica-se softmax para normalizar as pontuações.

- Pondera-se os valores V com as pontuações de atenção.

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

Arquitetura e funcionamento

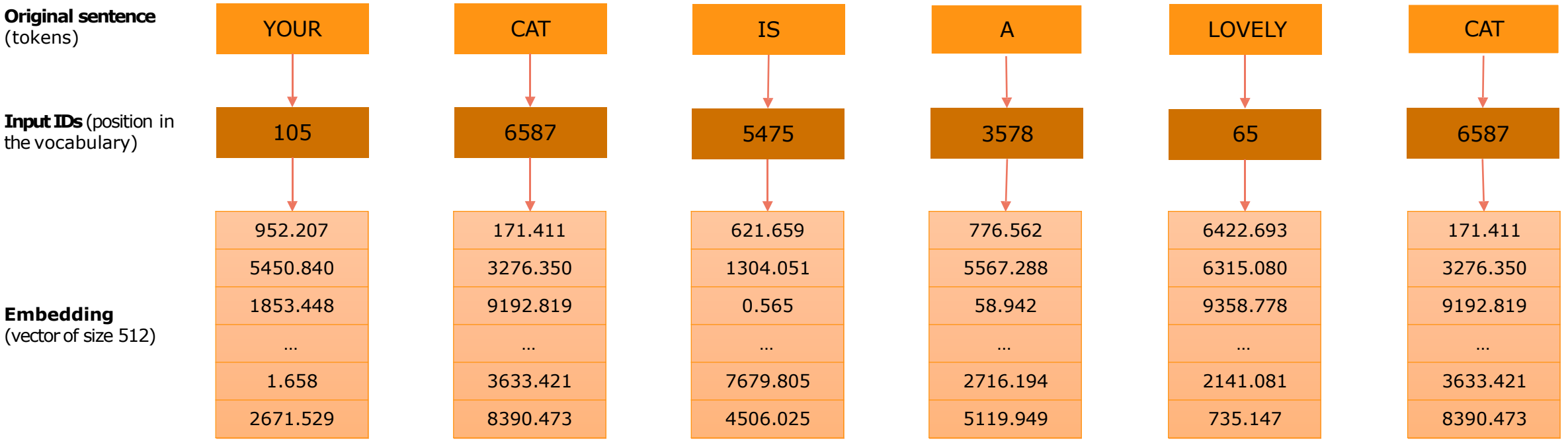


Codificador

- As palavras da sequência de entrada são convertidas em vetores de alta dimensão chamados ***embeddings***.
- **Positional Encoding**: Adiciona informações de posição aos embeddings das palavras.
- **Multi-Head Attention**: Captura dependências contextuais de longo alcance.

Arquitetura e funcionamento

Input embedding



We define $d_{model} = 512$, which represents the size of the embedding vector of each word

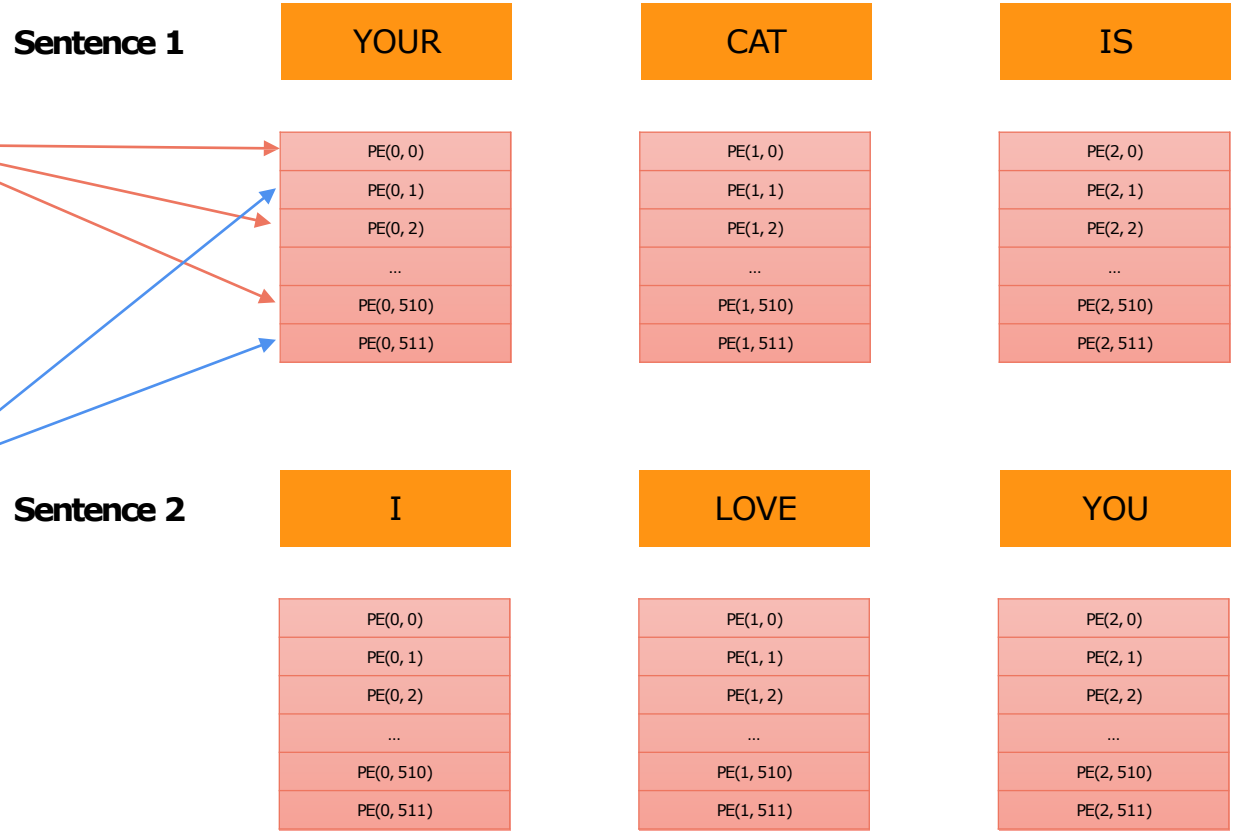
Umar Jamil – https://github.com/hkproj/transformer-from-scratch-notes

Arquitetura e funcionamento

Positional encoding

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

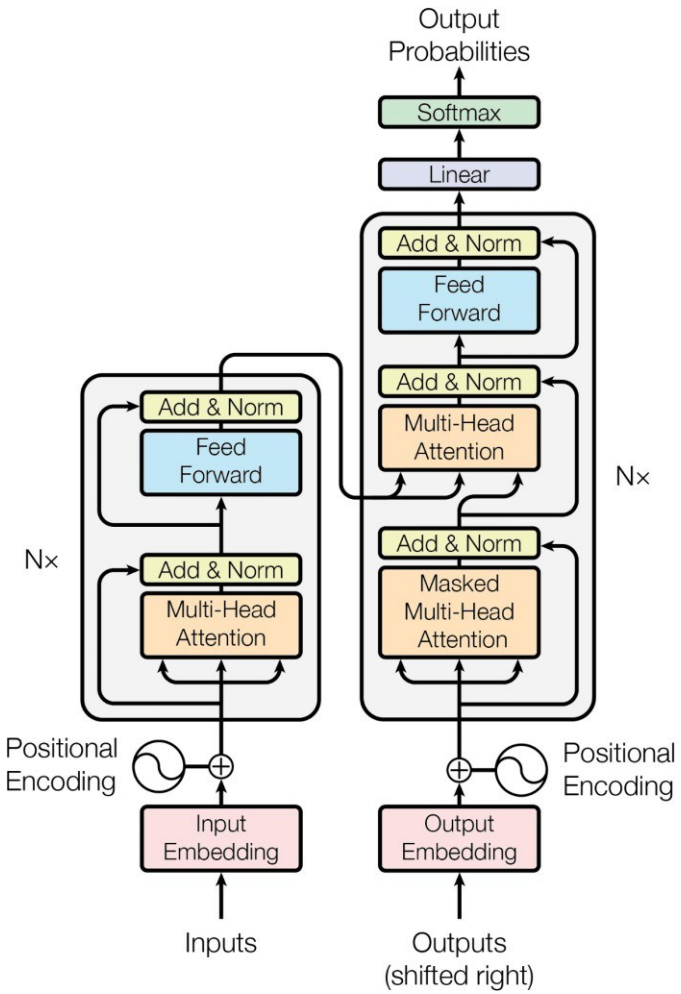
$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$



- ***pos*** é a posição na sequência.
- ***i*** é a dimensão do embedding.
- *dmodel* é a dimensão total do embedding

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

Arquitetura e funcionamento

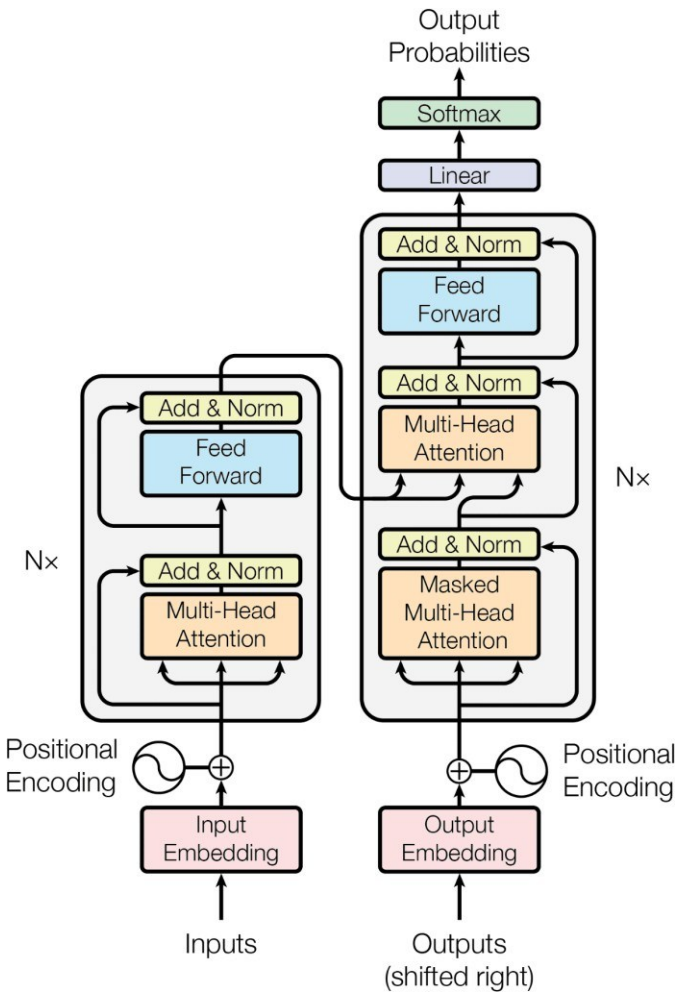


Codificador

- **Add & Norm**: Mantém a estabilidade do treinamento e facilita a propagação de gradientes.
- **Feedforward**: Rede neural composta por duas camadas lineares com uma função de ativação não linear (ReLU) no meio. *Processa cada posição de forma independente*.

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

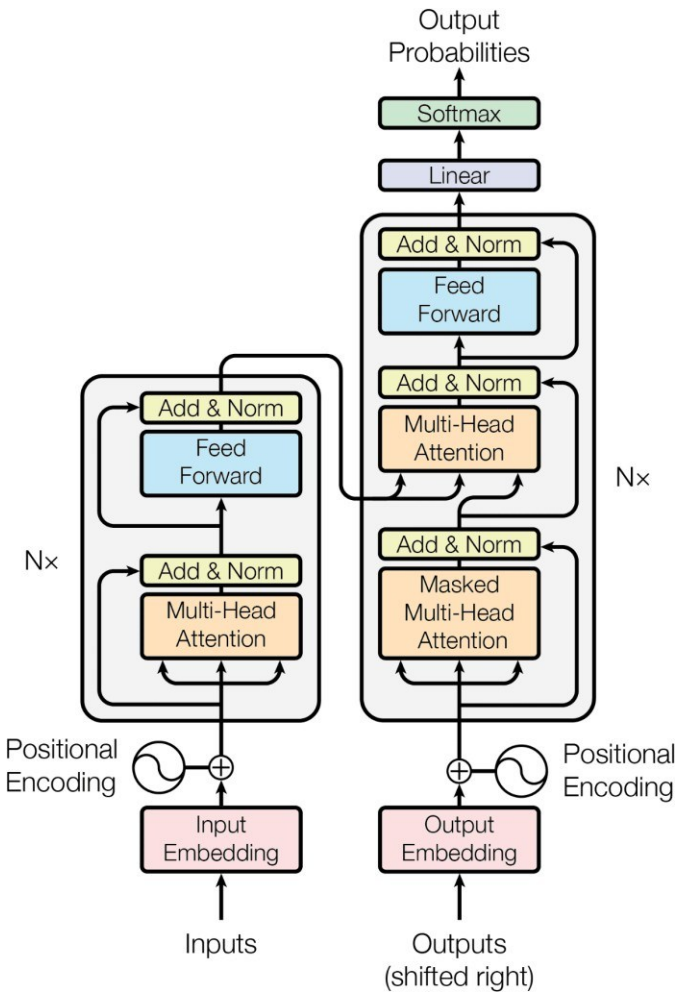
Arquitetura e funcionamento



Decodificador

- As palavras da sequência de saída (deslocadas para a direita) são convertidas em ***embeddings.***
- **Positional Encoding**: Adiciona informações de posição aos embeddings das palavras de saída.
- **Masked Multi-Head Attention** : Garante que a geração de saída seja auto-regressiva. Impede que a posição atual veja as futuras durante o treinamento.

Arquitetura e funcionamento

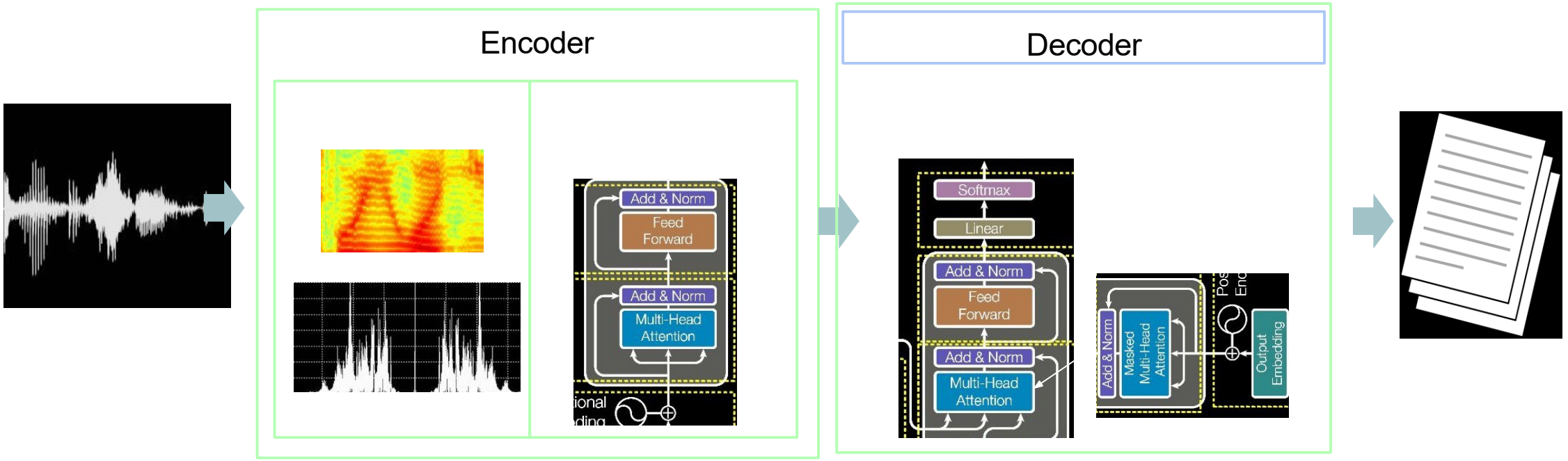


Decodificador

- **Encoder-Decoder Attention**: Integra a informação da sequência de entrada ao processo de geração da sequência de saída.
- **Camada Linear e Softmax**: Transforma a saída final do decodificador em probabilidades de palavras.
  - **Linear**: Projeta a saída do decodificador para o tamanho do vocabulário.
  - **Softmax**: Converte os valores projetados em probabilidades.

Arquitetura e funcionamento

Funcionamento do ASR com Transformer



Treinamento e otimização

Preparação dos Dados

Coleta e Pré-processamento

- **Coleta de Dados de Áudio:** Gravações de fala com transcrições.
- **Pré-processamento:** Extrair características como espectrogramas ou MFCCs.

Tokenização e Embeddings

- **Tokenização:** Dividir as transcrições em unidades menores (palavras ou sub-palavras).
- **Embeddings de Fala:** Converter tokens de áudio em vetores de alta dimensão.

Treinamento e otimização

Treinamento do Modelo

Estrutura do Transformer

- **Codificador (Encoder):** Converte áudio em representações internas.
- **Decodificador (Decoder):** Gera transcrições a partir das representações internas.

Treinamento e otimização

Função de Perda

Entropia Cruzada: Compara a sequência prevista com a real.

$$\text{Loss} = -\sum_{t=1}^{T} y_t \log(\hat{y}_t)$$

- *t* representa cada passo de tempo na sequência de palavras.
- *T* é o comprimento total da sequência (número de palavras na transcrição).
- *Yt* é o valor real ou verdadeiro rótulo para a palavra na posição.
- Y^t é a probabilidade prevista pelo modelo para a palavra na posição.

Otimização:

- **Algoritmo de Otimização:** Utilização do *Adam* para ajustes eficientes.

Vantagens e desvantagens

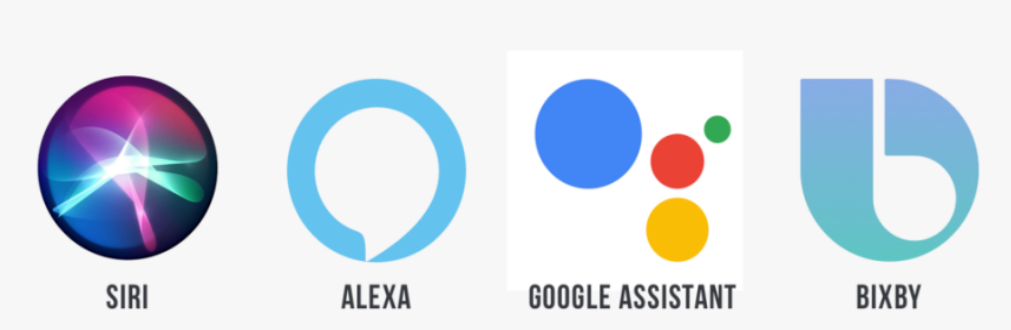
- Os Transformers revolucionaram o campo do reconhecimento de fala devido ao seu mecanismo de atenção multi-cabeças.
- **Vantagens:**
  - **Maior Precisão**: Por capturar relações contextuais longas e padrões complexos, são mais precisos e robustos.
  - **Contexto Global**: O mecanismo de atenção permite que os modelos entendam o contexto global da fala, beneficiando-se da análise simultânea de toda a sequência.
  - **Paralelização**: Processam a sequência de entrada de forma paralela, resultando em tempos de inferência mais rápidos.

Vantagens e desvantagens

- **Desvantagens**:
  - **Tempo de Treinamento**: Por terem muitas camadas e cabeças de atenção, os modelos Transformer requerem mais tempo de treinamento, especialmente se comparados aos métodos tradicionais.
  - **Dados Necessários**: Para alcançar a precisão desejada, requerem grandes volumes de dados rotulados, o que pode ser difícil de obter para domínios específicos.
  - **Recursos Computacionais**: O treinamento exige hardware potente (GPUs/TPUs) devido à complexidade computacional.

Exemplo(s) de aplicação

- **Sistemas de Assistentes Virtuais:**
  - **Exemplo:** Google Assistant, Siri, Alexa.
  - **Descrição:** Assistentes virtuais utilizam modelos Transformer para reconhecer comandos de voz e interpretar o contexto de conversas. A capacidade do mecanismo de atenção multi-cabeças ajuda a identificar a intenção do usuário e responder adequadamente, como ativar uma função do smartphone ou fornecer informações úteis.



Exemplo(s) de aplicação

- **Transcrição de Áudio:**
  - **Exemplo:** Otter.ai, Rev.com, Google Live Transcribe.
  - **Descrição:** Serviços de transcrição de áudio para texto utilizam modelos Transformer pré-treinados para identificar palavras e frases em tempo real. A atenção global do Transformer possibilita a compreensão de longos trechos de áudio, mesmo com variações de pronúncia, sotaque e ruído de fundo.

Exemplo(s) de aplicação

- **Reconhecimento de Fala para Atendimento ao Cliente:**
  - **Exemplo:** Chatbots, Call Centers.
  - **Descrição:** Os call centers usam modelos Transformer para automatizar partes do atendimento ao cliente. Os chatbots podem entender as perguntas do usuário e fornecer respostas adequadas, direcionando chamadas para setores específicos com mais precisão.
- **Análise de Sentimentos em Mídias Sociais:**
  - **Exemplo:** Ferramentas de análise de dados de redes sociais.
  - **Descrição:** Ferramentas de análise de áudio em redes sociais usam reconhecimento de voz para transcrever e analisar os sentimentos expressos em vídeos e mensagens de áudio, fornecendo insights sobre tendências e preferências do público.

Comparação com outros algoritmos

Critério	Transformer para ASR	HMM-GMM (Hidden Markov Models-Gaussian Mixture Models)	DNN-HMM (Deep Neural Networks-Hidden Markov Models)	RNN (Recurrent Neural Networks) / LSTM (Long Short-Term Memory)
Estrutura	Codificador-Decodificador com atenção	Modelos baseados em estados ocultos	Redes Neurais Profundas acopladas com HMMs	Redes Neurais Recorrentes
Treinamento	Computacionalmente intensivo, mas eficiente	Menos intensivo, mas mais lento devido à falta de paralelização	Intensivo devido às DNNs, mas ainda limitado	Intensivo e lento devido à dificuldade de paralelização
Paralelização	Altamente paralelizável	Processamento sequencial, pouca paralelização	Melhoria na paralelização com DNN	Difícil de paralelizar devido ao processamento sequencial

Comparação com outros algoritmos

Critério	Transformer para ASR	HMM-GMM (Hidden Markov Models-Gaussian Mixture Models)	DNN-HMM (Deep Neural Networks-Hidden Markov Models)	RNN (Recurrent Neural Networks) / LSTM (Long Short-Term Memory)
Precisão	Alta precisão, especialmente em grandes dados	Precisão moderada, dependente da qualidade dos dados	Boa precisão, mas depende da segmentação HMM	Alta precisão, especialmente com grandes conjuntos de dados
Robustez a Variabilidade	Alta, pode capturar variabilidade complexa	Baixa, depende de muitas suposições	Moderada, melhor que HMM-GMM	Alta, pode lidar com variabilidade temporal
Implementação	Complexa	Simples, mas manualmente intensiva	Moderada, necessita de integração de HMM e DNN	Complexa, especialmente com LSTMs

Exemplo(s) de aplicação

https://colab.research.google.com/drive/1YSE3X8oy_owbAT-jH9siq-MLcGF6nESW?usp=sharing

Perguntas?

Referências

[1]https://papers.nips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[2] https://keras.io/examples/audio/transformer_asr/

[3] https://jalammar.github.io/illustrated-transformer/

[4] Kevin Clark, Urvashi Khandelwal, Omer Levy e Christopher D. Manning, O que o BERT analisa? Uma análise da atenção do BERT (2019).

[5] Biblioteca de transformadores do Huggingface (https://github.com/huggingface/transformers)

[6] Google AI Blog: [Using Deep Learning to Automatically Caption YouTube Videos](https://ai.googleblog.com/2020/03/using-deep-learning-to-automatically.html)

Quiz

- Quiz - Automatic Speech Recognition with Transformer

- Anexe o *print screen* da pontuação obtida à tarefa específica do MS Teams.

Obrigado!