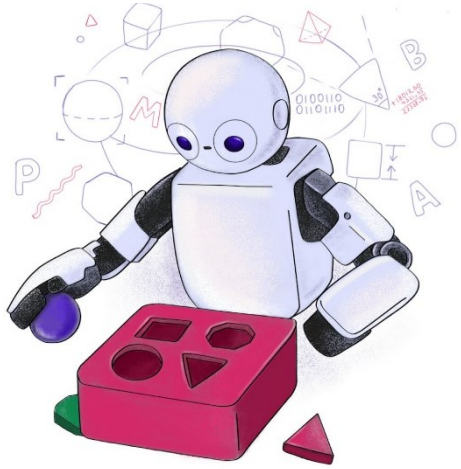
TP558 - Tópicos avançados em Machine Learning:

Automatic Speech Recognition with Transformer



Inatel

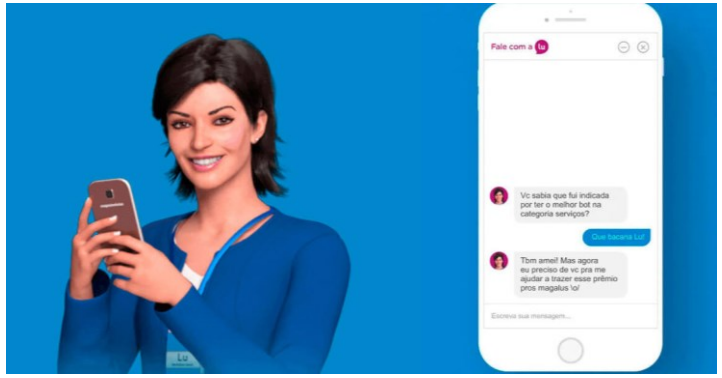
Georgino da Silva Baltazar
georgino.baltazar@dtel.inatel.br

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**.

Evolução dos Sistemas de Reconhecimento de Fala

- **Décadas de 1950-1980 :** Os primeiros sistemas de reconhecimento de fala baseavam-se em algoritmos de ***coincidência de padrões***, onde os sons eram comparados com um conjunto pré-definido de padrões armazenados.
- **Na década de 1970**, os ***Modelos de Markov Ocultos (HMM)*** foram introduzidos e se tornaram a base dos sistemas do ASR. Esses modelos probabilísticos são eficazes na modelagem de sequências temporais e foram amplamente utilizados devido à sua capacidade de lidar com variações na fala.

Evolução dos Sistemas de Reconhecimento de Fala

- **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.
- Isso permitiu que os sistemas de ASR levassem em consideração a probabilidade das sequências de palavras, melhorando ainda mais a precisão.

Evolução dos Sistemas de Reconhecimento de Fala

- **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.
- ***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.

Evolução dos Sistemas de Reconhecimento de Fala

- **2017-Presente -** Modelos Modernos e Transformers.
- A introdução do modelo ***Transformer***, descrito no artigo "*Attention Is All You Need*" por Vaswani et al. (2017), revolucionou o campo do processamento de linguagem natural (NLP) e, consequentemente, o ASR.
- O modelo ***Transformer foi originalmente desenvolvido para tarefas de processamento de linguagem natural***, como tradução automática.

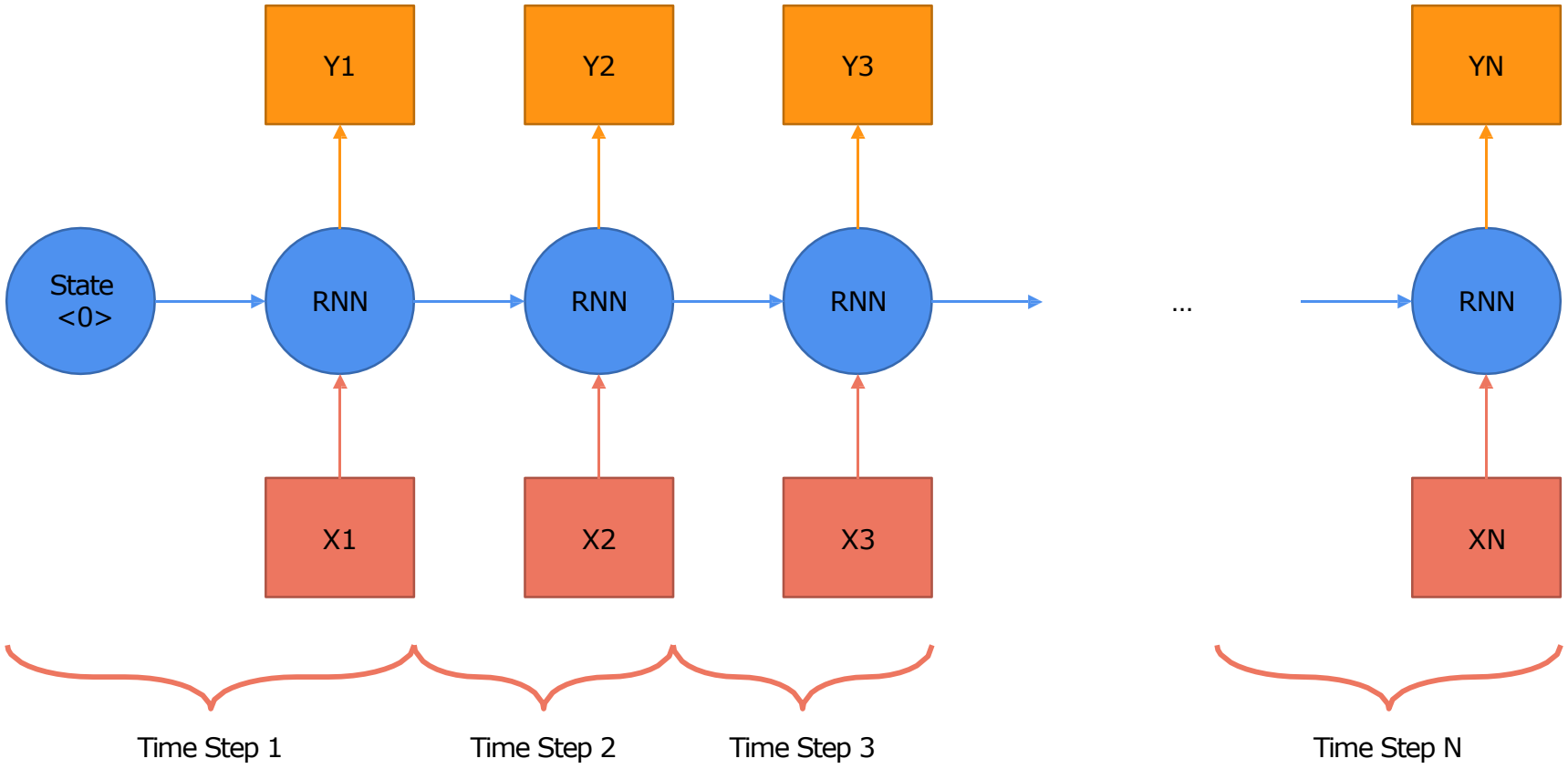
Evolução dos Sistemas de Reconhecimento de Fala

- **2017-Presente -** Modelos Modernos e Transformers.
- Os ***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.*
- Modelos Pré-treinados como ***BERT***, ***GPT*** e ***Whisper aplicam a arquitetura Transformer*** para alcançar resultados de ponta em várias tarefas de NLP e ASR. O ***Whisper***, por exemplo, utiliza Transformars para fornecer um desempenho superior em reconhecimento de fala.

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

Redes Neurais Recorrentes



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

Problemas com RNN (entre outros)

- Computação lenta para sequências longas
- Gradientes desaparecendo ou explodindo
- 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

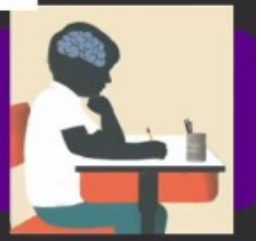
Llion Jones* Google Research llion@google.com

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

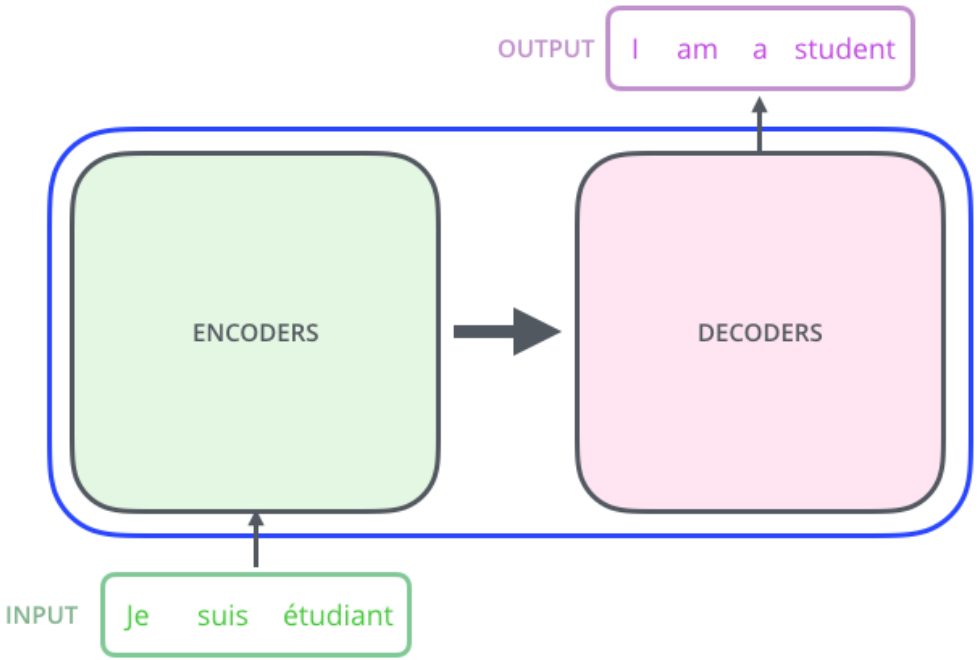


Originalmente sequence-to-sequence

+ 122 mil citações

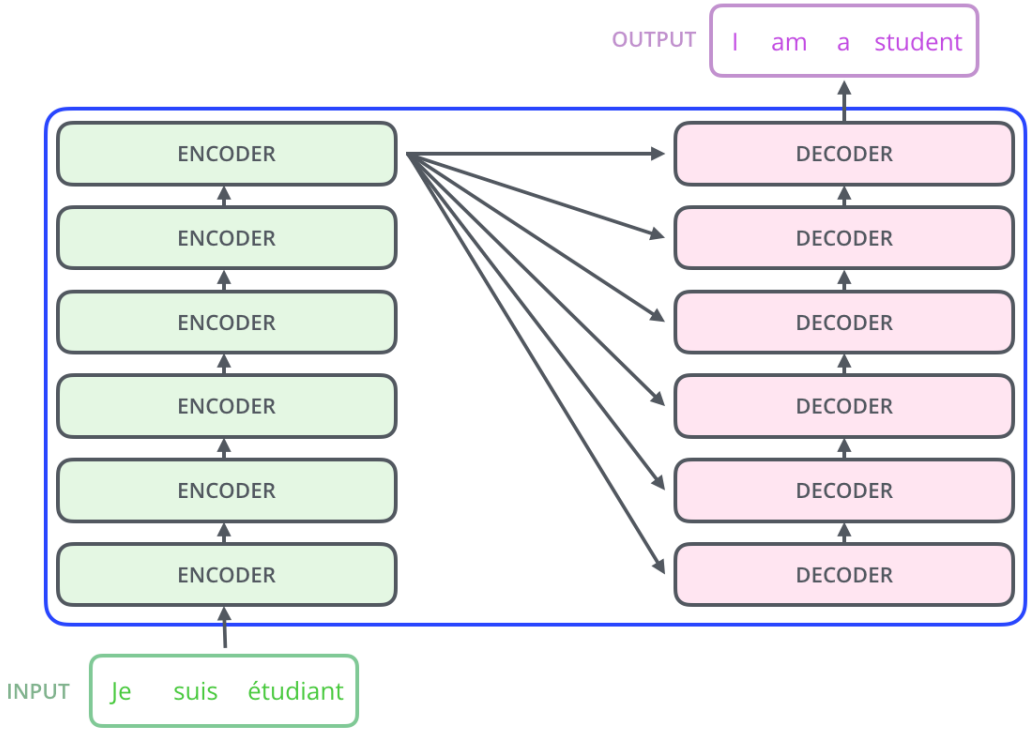
Transformers

Transformers - Arquitetura



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

Transformers – Arquitetura

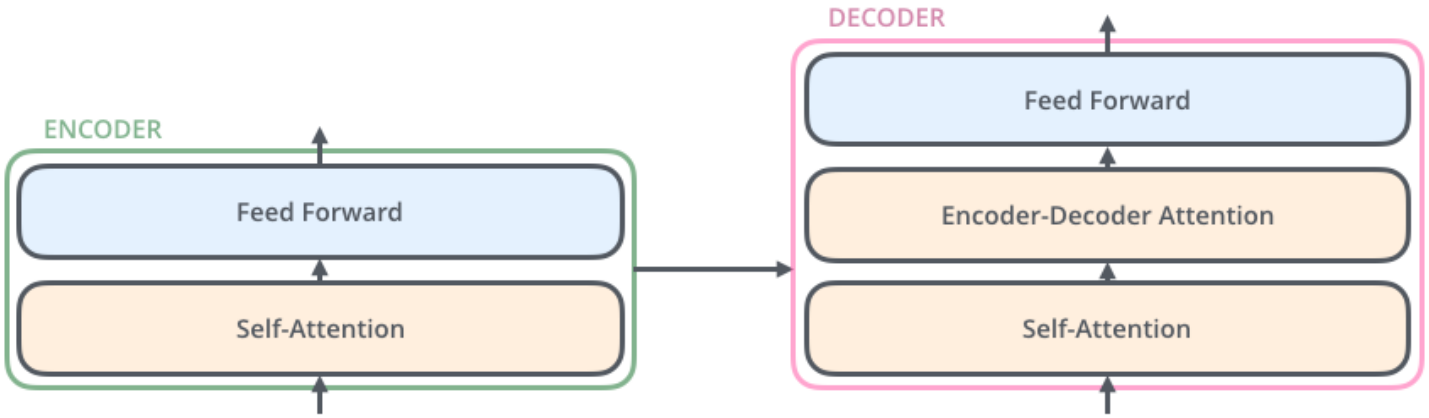


O componente de codificação é *uma pilha de codificadores*.

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

Transformers – Arquitetura

- 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.

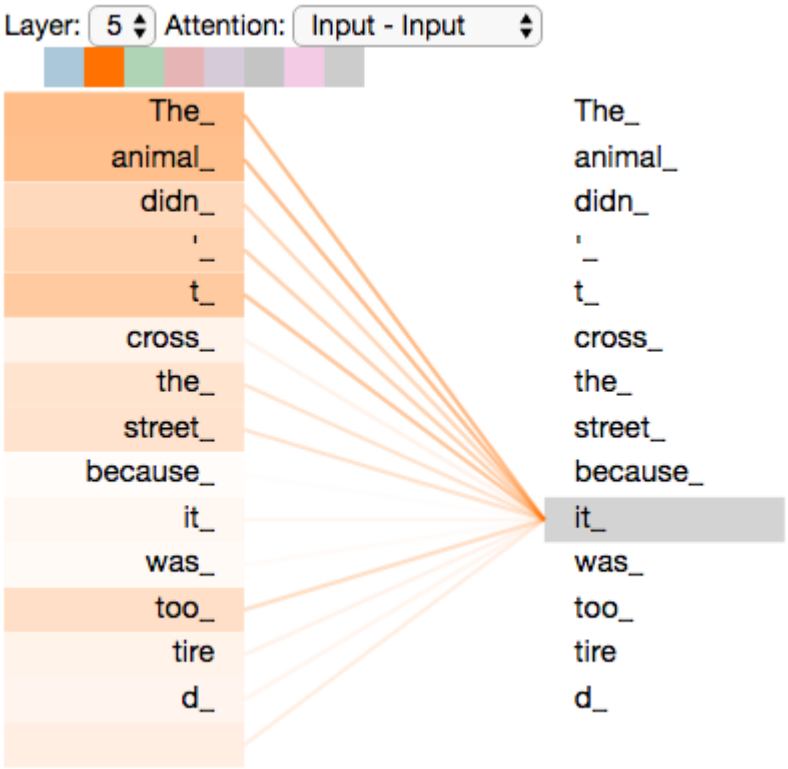


O que é Self-Attention?

O que é Self-Attention ?

- *Self-Attention* é um *mecanismo utilizado em modelos de aprendizado profundo, especialmente em Transformers*, ***para permitir que cada elemento de uma sequência considere todos os outros elementos da mesma sequência ao calcular sua representação.***

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.

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

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

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.*

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

Funcionamento do Self-Attention

- 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

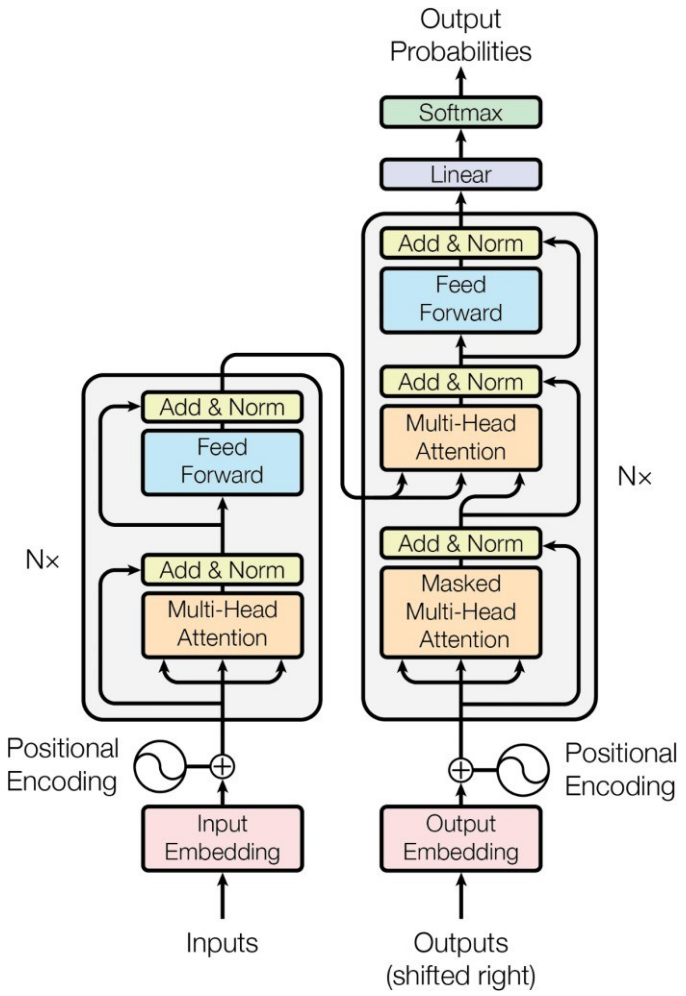
2. Cálculo das Similaridades e Ponderação:

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

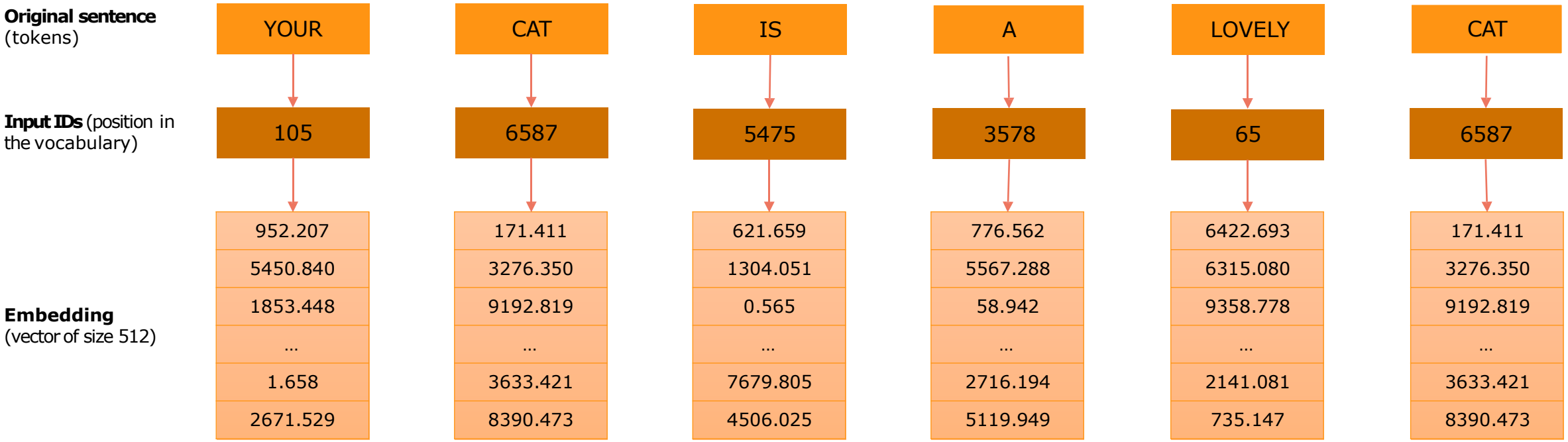
- Aplica-se softmax para normalizar as pontuações.

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

Arquitetura do Transformer



O que é um 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

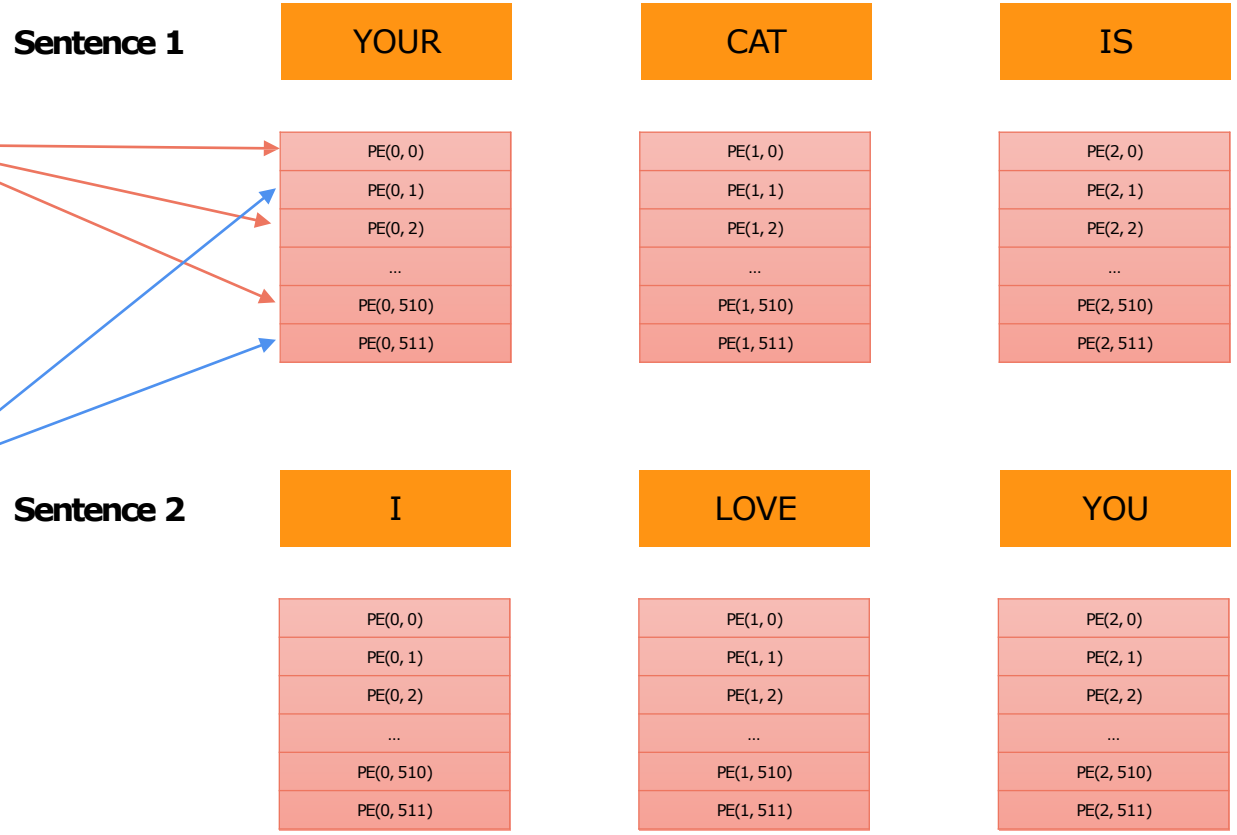
O que é positional encoding?

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

Oque é 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

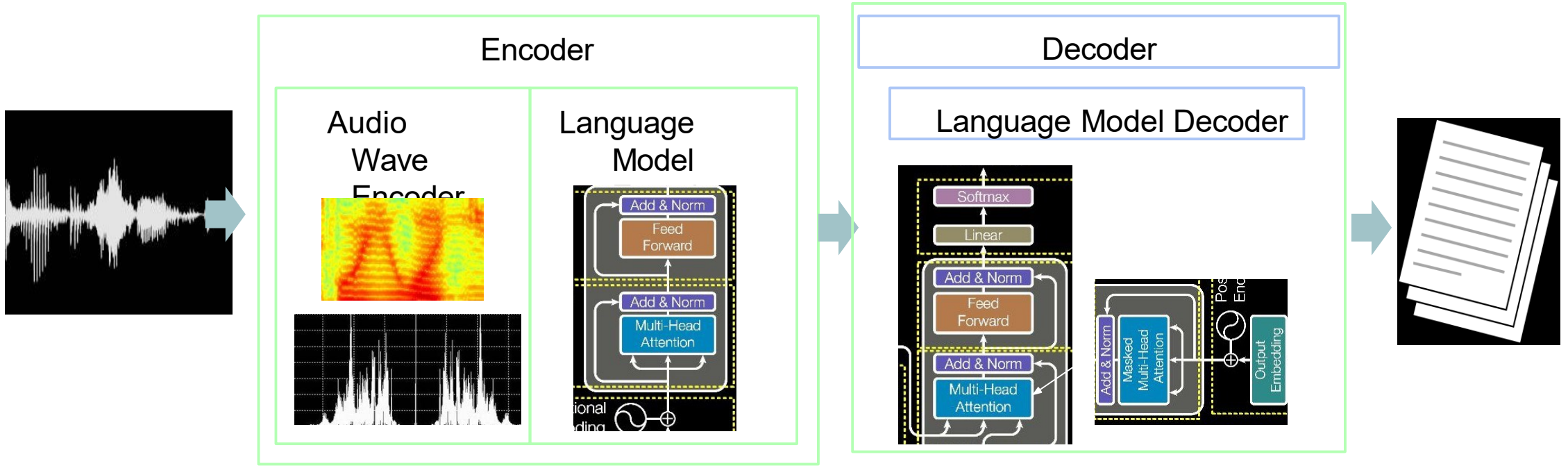
Funcionamento do ASR com Transformer

- **Pré-processamento de Áudio (Input Layer):** O áudio é convertido em uma representação numérica, como um espectrograma ou características extraídas usando técnicas como Mel-Frequency Cepstral Coefficients (MFCCs) ou *wav2vec*. Estas características são a entrada inicial do Transforme
- **Codificador (Encoder Layers): Atenção Multi-Cabeças (Self-Attention):** Cada camada do codificador contém um mecanismo de auto-atenção multi-cabeças. Isso permite que o modelo atenda a diferentes partes da sequência de entrada simultaneamente, identificando relações entre diferentes características de áudio.

Funcionamento do ASR com Transformer

- **Camada de Saída (Output Layer):**
  - A saída final do decodificador é processada para gerar a sequência textual correspondente à transcrição da fala.
- **Rede Feed-Forward:** Após a atenção, cada camada possui uma rede feed-forward que aplica uma transformação não-linear aos vetores de atenção, enriquecendo a representação das características.

Funcionamento do ASR com Transformer



Treinamento e otimização

- Para treinar modelos Transformer no reconhecimento automático de fala (ASR), são utilizadas várias técnicas e estratégias, adaptando-se a diferentes conjuntos de dados, domínios e requisitos de precisão.
- **Aprendizado Supervisionado:**
  - **Definição:** No aprendizado supervisionado, o modelo é treinado com pares de entrada e saída rotulados. No caso do ASR, as entradas são características de áudio e as saídas são as transcrições correspondentes.

Treinamento e otimização

- **Estratégia:**
  - Uma grande quantidade de dados de áudio transcritos é necessária para capturar adequadamente as variações da fala, sotaques e estilos de linguagem.
  - O conjunto de dados é dividido em subconjuntos de treinamento, validação e teste.
  - O erro entre as previsões do modelo e as transcrições reais é usado para calcular a função de perda, como a *Connectionist Temporal Classification* (CTC), e ajustar os pesos das camadas da rede.

Treinamento e otimização

- **Preparação de Dados**
- **Coleta e Pré-processamento**:
  - **Coleta**: Obtenha um conjunto de dados de áudio transcrito. Os exemplos podem incluir gravações de diferentes durações, idiomas e sotaques.
  - **Pré-processamento**: Converta os arquivos de áudio em representações numéricas, como espectrogramas, Mel-Frequency Cepstral Coefficients (MFCCs) ou representações extraídas por modelos pré-treinados (por exemplo, *wav2vec*).
- **Divisão dos Dados**:
  - **Treinamento**: A maioria dos dados (geralmente 70-80%) é usada para treinar o modelo.
  - **Validação**: Uma parte menor (10-20%) é reservada para validação, monitorando o desempenho durante o treinamento.
  - **Teste**: O restante (10-15%) é usado para testar a precisão final.

Treinamento e otimização

- **Treinamento Supervisionado**
- **Objetivo**:
  - Usar pares de entrada e saída rotulados (áudio e transcrições) para calcular a perda e ajustar os pesos do modelo.
- **Função de Perda**:
  - **Connectionist Temporal Classification (CTC)**: Uma função de perda usada para alinhar seqüências de áudio com transcrições, corrigindo diferenças de duração.
  - **Cross-Entropy Loss**: Utilizada nos modelos que geram texto diretamente, como na tradução.
- **Otimização**:
  - **Backpropagation**: Calcula-se o gradiente da função de perda em relação aos parâmetros e ajusta-se os pesos das camadas por meio do método de otimização como Adam.
  - **Batch Training**: O treinamento é feito em lotes, processando múltiplos exemplos em paralelo para eficiência.

  - O objetivo geral é treinar o modelo para capturar as nuances das características de áudio, e gerar transcrições precisas ao compreender os padrões de dependência entre as diferentes partes da fala.

Vantagens

- 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.

Desvantagens

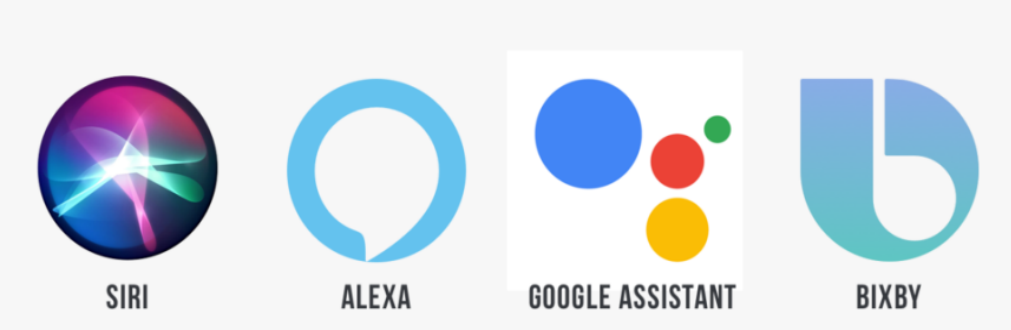
- **Desvantagens**:
  - **Tempo de Treinamento**: Por terem muitas camadas e cabeças de atenção, os modelos Transformer requerem mais tempo de treinamento, especialmente se comparados aos métodos tradicionais.
  - **Dados Necessários**: Para alcançar a precisão desejada, requerem grandes volumes de dados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

Vantagens e desvantagens

- **Resumo**
- Os Transformers se destacam em tarefas onde a precisão e o contexto são críticos, enquanto os métodos tradicionais ainda têm sua utilidade onde simplicidade e eficiência computacional são mais importantes. A escolha entre ambos deve ser feita considerando as demandas específicas da aplicação e os recursos disponíveis.

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

- **Tradução Automática:**
  - **Exemplo:** Google Translate, DeepL.
  - **Descrição:** Modelos Transformer podem ser treinados para converter a fala de um idioma para texto, e depois traduzir esse texto para outra língua. A arquitetura codificador-decodificador consegue traduzir falas complexas com precisão, identificando nuances e expressões idiomáticas.
- **Ferramentas de Acessibilidade:**
  - **Exemplo:** Closed Captioning, Audiodescrição.
  - **Descrição:** Serviços de legendagem automática em tempo real e descrição de áudio para deficientes auditivos e visuais são outra aplicação dos Transformers. Eles capturam e processam diálogos, fornecendo legendas precisas ou descrição do conteúdo visual.

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

Tabela de Comparação: Transformer para ASR vs. Outros Algoritmos de ASR

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
Captura de Dependências	Dependências de longo alcance via atenção	Dependências de curto alcance	Dependências de curto alcance	Dependências de longo alcance com estados ocultos
Paralelização	Altamente paralelizável	Processamento sequencial, pouca paralelização	Melhoria na paralelização com DNN	Difícil de paralelizar devido ao processamento sequencial

Tabela de Comparação: Transformer para ASR vs. Outros Algoritmos de ASR

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)
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
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
Escalabilidade	Altamente escalável	Difícil de escalar devido à complexidade	Moderadamente escalável	Moderadamente escalável, mas limitado pelo tempo de treinamento
Robustez a Variabilidade	Alta, pode capturar variabilidade complexa	Baixa, depende de muitas suposições	Moderada, melhor que HMM-GMM	Alta, pode lidar com variabilidade temporal

Tabela de Comparação: Transformer para ASR vs. Outros Algoritmos de ASR

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)
Robustez a Variabilidade	Alta, pode capturar variabilidade complexa	Baixa, depende de muitas suposições	Moderada, melhor que HMM-GMM	Alta, pode lidar com variabilidade temporal
Requisitos de Dados	Grandes volumes de dados necessários	Menos dados necessários, mas de alta qualidade	Necessita de grandes volumes de dados	Necessita de grandes volumes de dados
Implementação	Complexa	Simples, mas manualmente intensiva	Moderada, necessita de integração de HMM e DNN	Complexa, especialmente com LSTMs
Exemplo de Aplicação	Tradução automática de fala, sistemas de assistente virtual	Sistemas de IVR, reconhecimento de fala telefônica	Aplicações em empresas, sistemas de IVR	Aplicações em assistentes virtuais, tradução simultânea

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)

Obrigado!

Quiz

- Quiz - Automatic Speech Recognition with Transformer

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