Universidade de São Paulo
Instituto de Matemática e Estatística
Bacharelado em Ciência da Computação

Modelos Transformer para Inferência de Linguagem Natural em Português

Giovani Tavares de Andrade

Monografia Final

mac 499 — Trabalho de Formatura Supervisionado

Supervisora: Profª. Drª Renata Wassermann
Cossupervisor: Me. Felipe Ribas Serras

São Paulo
2023

Esta seção é opcional e fica numa página separada;
ela pode ser usada para uma dedicatória ou epígrafe.

Agradecimentos

Do. Or do not. There is no try.
— Mestre Yoda

Texto texto. Texto opcional.

Resumo

Giovani Tavares de Andrade. **Modelos Transformer para Inferência de Linguagem Natural em Português**. Monografia (Bacharelado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2023.

Elemento obrigatório, constituído de uma sequência de frases concisas e objetivas, em forma de texto. Deve apresentar os objetivos, métodos empregados, resultados e conclusões. O resumo deve ser redigido em parágrafo único, conter no máximo 500 palavras e ser seguido dos termos representativos do conteúdo do trabalho (palavras-chave). Deve ser precedido da referência do documento. Texto texto. Texto texto.

Palavras-chave: Palavra-chave1. Palavra-chave2. Palavra-chave3.

Abstract

Giovani Tavares de Andrade. **Transformer Models for Natural Language Inference in Portuguese**. Capstone Project Report (Bachelor). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2023.

Elemento obrigatório, elaborado com as mesmas características do resumo em língua portuguesa. De acordo com o Regimento da Pós-Graduação da USP (Artigo 99), deve ser redigido em inglês para fins de divulgação. É uma boa ideia usar o sítio www.grammarly.com na preparação de textos em inglês. Text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text. Text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text.

Keywords: Keyword1. Keyword2. Keyword3.

Lista de Abreviaturas

NLP	*Natural Language Processing* (Processamento de Linguagem Natural)
RNN	*Recurrent Neural Networks* (Redes Neurais Recorrentes)
NLI	*Natural Language Inference* (Inferência de Linguagem Natural)
CFT	Transformada contínua de Fourier (*Continuous Fourier Transform*)
DFT	Transformada discreta de Fourier (*Discrete Fourier Transform*)
EIIP	Potencial de interação elétron-íon (*Electron-Ion Interaction Potentials*)
STFT	Transformada de Fourier de tempo reduzido (*Short-Time Fourier Transform*)
ABNT	Associação Brasileira de Normas Técnicas
URL	Localizador Uniforme de Recursos (*Uniform Resource Locator*)
IME	Instituto de Matemática e Estatística
USP	Universidade de São Paulo

Lista de Símbolos

ω	Frequência angular
ψ	Função de análise *wavelet*
Ψ	Transformada de Fourier de ψ

Lista de Figuras

Lista de Tabelas

Lista de Programas

Lista de Definições

Sumário

Capítulo 1

Revisão Bibliográfica

1.1 *Fine Tuned Models*

Foram utilizados modelos *Transformer* [Vaswani *et al.*, 2017] para a realização da tarefa de NLI. *Transformers* são redes neurais capazes de classificar pares de sentenças na forma (*premise*, *hypothesis*) como descrito em [INSERIR LINK - DEFINIÇÃO DE NLI] usando a abordagem de *machine learning*.

Esses modelos possuem a estrutura das redes neurais *sequence-to-sequence* com os melhores desempenhos atualmente [Vaswani *et al.*, 2017] denominada estrutura *encoder-decoder*. Modelos *sequence-to-sequence* (Sequência-para-sequência, em tradução livre ao português) produzem sequências de texto a partir de outras sequências. São, portanto, modelos generativos cuja saída e entrada são textos reais ou sintéticos.

1.1.1 A arquitetura *Transformer*

Introduzidos pela primeira vez em 2017 no artigo "*Attention is All You Need*"[Vaswani *et al.*, 2017], os *Transformers* submetem a sequência de entrada em uma camada de codificação denominada *encoder* seguida de uma de decodificação denominada *decoder*. A arquitetura *encoder-decoder* resultante desse processo não é exclusividade dos *Transformers*, tendo sido demonstrada em redes neurais de tradução [Sutskever *et al.*, 2014] anteriormente. O denominado "Mecanismo de Atenção"presente em ambas camadas citadas é, por sua vez, parte importante dos aspectos que diferencia os modelos apresentados em Vaswani *et al.*, 2017 de outros modelos *sequence-to-sequence*.

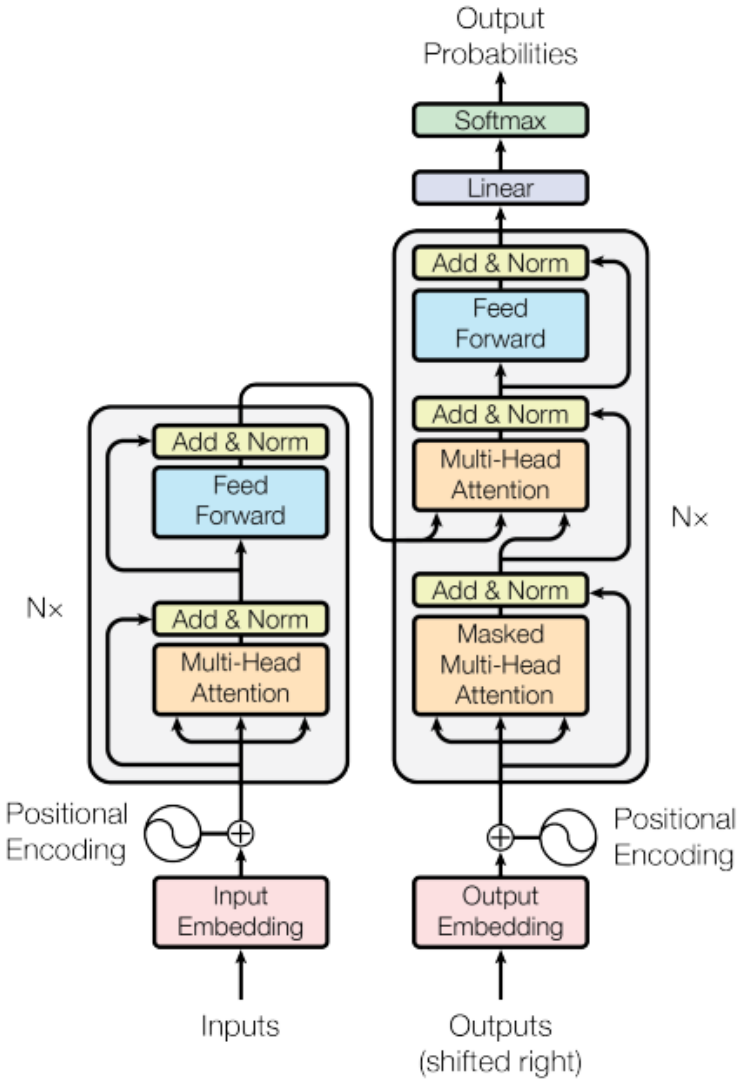

Figura 1.1: *A arquitetura Transformer*

O Mecanismo de Atenção

Definição 1. Atenção

Sejam $q \in \mathbb{R}^{d_{model}}$, $k \in \mathbb{R}^{d_{model}}$ e $v \in \mathbb{R}^{d_{model}}$, a *Função de Atenção* é definida abaixo:

$$Attention(q, k, v) = SoftMax(\frac{\langle q, k \rangle}{\sqrt{d_k}})v = a_{q,k,v}$$

onde d_{model} é o número de dimensões do modelo.

Da definição acima é possível inferir $a_{q,k,v} \in \mathbb{R}^{d_{model}}$. Abaixo encontram-se interpretações para q, k e v.

- **Query ($q \in \mathbb{R}^{d_{model}}$)**: é a representação vetorial da palavra ou sequência para a qual

a atenção $a_{q,k,v}$ está sendo calculada. Usando vetores q, o *Transformer* é capaz de comparar diferentes palavras em uma sentença.

- **Key ($k \in \mathbb{R}^{d_{model}}$)**: é a representação vetorial de outra palavra ou sequência no texto sendo processado pelo modelo. A realização do produto escalar entre ele e q permite ao *Transformer* mensurar a similaridade entre palavras de uma sentença.
- **Value ($v \in \mathbb{R}^{d_{model}}$)**: é a representação vetorial da informação contextual que cada palavra no texto possui. Dessa forma, ao multiplicar o resultado do produto escalar entre q e k por v, o que se obtem é a similaridade entre q e k ponderada pelo contexto que k carrega representado por v.

Evidentemente, os textos submetidos como *input* aos *Transformers* são formados por diversas palavras e, portanto, o mecanismo de atenção deve ser aplicado a múltiplas *queries*, *keys* e *values*. Em Vaswani *et al.*, 2017 é introduzida o *Scaled Dot-Product Attention*. Tal definição sugere como aplicar o mecanismo de atenção de 1 em múltiplos q, k e v simultaneamente. Para isso, todas as *queries* q, *keys* k e *values* v são empilhadas nas matrizes Q, K e V respectivamente. Abaixo se define o *Scaled Dot-Product Attention* como em Vaswani *et al.*, 2017.

Definição 2. ***Scaled Dot-Product Attention***

Sejam $Q \in \mathbb{R}^{m_q \times d_{model}}$, $K \in \mathbb{R}^{m_k \times d_{model}}$ e $V \in \mathbb{R}^{m_k \times d_{model}}$, a função *Scaled Dot-Product Attention* é definida abaixo:

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

m_q é o número de *queries* cujas atenções estão sendo cauculadas e m_k é o número de *keys* com as quais as *queries* serão comparadas.

Analogamente ao que foi feito em 1, é possível inferir $A_{Q,K,V} \in \mathbb{R}^{m_q \times d_{model}}$. Assim, pode-se interpretar a atenção resultante de 3 como uma matriz em que cada linha representa a atenção de uma única *query* relativa a diversas *keys*.

Tanto em 1 quanto em 3, todas as *queries*, *keys* e *values* têm d_{model} dimensões. Ou seja, toda a informação é representada em subespaços vetoriais de dimensionalidade igual à do modelo.

Atenção *Multi-Head*

Vaswani *et al.*, 2017 alega ser benéfico projetar linearmente as *queries*. *keys* e *values* h vezes em diferentes subespaços de dimensões d_k, d_k e d_v, respectivamente. Usando cada

uma dessas projeções a atenção (3 é realizada de maneira paralela gerando resultados com d_v dimensões. Tais resultados são concatenados e então projetados de volta para o espaço com d_{model} dimensões, gerando o resultado final do mecanismo de atenção (Vaswani *et al.*, 2017).

Definição 3. Atenção *Multi-Head*

Sejam $Q \in \mathbb{R}^{m_q \times d_{model}}$, $K \in \mathbb{R}^{m_k \times d_{model}}$ e $V \in \mathbb{R}^{m_k \times d_{model}}$. A atenção *Multi-Head* é definida abaixo:

$$MultiHead(Q, K, V) = Concat(head_1, head_2, ..., head_h)W^0$$
$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

where

$$W_i^Q \in \mathbb{R}^{d_{model}} \times d_k$$
$$W_i^k \in \mathbb{R}^{d_{model} \times d_k}$$
$$W_i^V \in \mathbb{R}^{d_{model}} \times d_v$$
$$W^0 \in \mathbb{R}^{hd_v} \times d_{model}$$

As projeções referidas são realizadas com as matrizes W acima. O uso da atenção com h representações distintas das *queries*, *keys* e *values* permite aos *Transformers* a representação de sentenças e textos segundo diversos aspectos e relações entre seus componentes de maneira paralelizável. Esse é um aspecto fundamental desse tipo de arquitetura presente em diversos modelos de reconhecido desempenho em diferentes tarefas de NLP.

O *encoder*

O *encoder* representado à esqueda na Figura 1.1 é uma função cuja saída é um vetor de contexto e a entrada uma sentença de texto representada por *word embeddings* [INSERIR LINK- DEFINIÇÃO DE WORD EMBEDDING]. Em linhas gerais, o *encoder* é quem codifica as informações e relações existentes no texto e linguagem das entradas de modo que o *decoder* as explore a fim de gerar novos textos.

Nos *Transformers*, há um total de 6 camadas idênticas de *encoders* empilhadas, o que significa que a saída de uma camada é entrada para a próxima.

Esse processo de codificação realizado pela camada do *encoder* tem uma etapa de atenção *Multi-Head* como se observa em 1.1. Nesse caso, as *queries*, *keys* e *values* são recuperadas da última camada de codificação, o que se denomina **autoatenção**.

O *decoder*

O *decoder* representado à direita na Figura 1.1 tem como função produzir textos a partir da saída da camada de codificação, ao que se denonima decodificação. Dessa forma, a qualidade do texto gerado pelo *decoder* depende diretamente da qualidade dos vetores de contexto produzidos pelo *encoder*.

Tal qual nos *encoders*, os *decoders* são formados por 6 camadas conectadas. Por outro lado, cada camada do *decoder* aplica a atenção duas vezes, sendo a primeira a autoatenção. A autoatenção permite ao *decoder* considerar todas as posições de texto produzidas até a posição sendo gerada naquele momento, mas não após ela. Essa característica é crucial, porque durante o treinamento o modelo tem acesso à sentença completa a ser gerada pelo *decoder*, mas em tempo de inferência a sentença é gerada com um *token* de cada vez. Por conseguinte, cada posição em uma sentença gerada por um *Transformer* depende apenas das posições anteriores a ela, ao que se chama de **propriedade de autoregressão**.

Na **autoatenção mascarada**, denominada *Masked Multi-Head Attention* em 1.1, as posições em QK^T referentes a posteridades da posição para a qual se calcula a atenção são ocultadas Para isso, elas transformadas em $-\infty$ durante o treinamento (Vaswani *et al.*, 2017).

A segunda aplicação da atenção no *decoder* utilizanda as *keys* e *values* geradas pelo *encoder* e *queries* pela última camada de decodificação. Nessa etapa, não há a ocultação de nenhuma parte de QK^T, já que K resulta do *encoder* e, portanto, QK^T representa apenas relações entre a saída da últma camada do *decoder* com a entrada do *encoder*. Ou seja, K, nesse caso, possui codificadas pelo *encoder* presentes em tempo de treinamento de inferência.

Os benefícios da atenção

As camadas de autoatenção no *encoder* e *decoder* conectam todas as posições de uma sentença entre si com um número constante de operações para cada par de posições (Vaswani *et al.*, 2017). Isso significa que em um texto as relações entre a primeira e a segunda palavras são codificadas e decodificadas usando o mesmo número de operações que se faria para o mesmo processo entre a primeira e a última palavras. Essa operação é, portanto, $\mathcal{O}(1)$ do ponto de vista de complexidade computacional.

A arquitetura de uma RNN como descrita no capítulo 9 de Jurafsky e Martin, 2021 possui complexidade $\mathcal{O}(n)$ para a mesma operação, em que n é a posicão para a qual são calculadas as relações (Vaswani *et al.*, 2017).

Capítulo 2

Testes Cruzados

- O que são os testes cruzados
- Por que foram realizados
- As transformações de realizadas para realizá-los
- Que tipo de comparações eles permitem que sejam feitas entre os modelos

2.1 Resultados

2.1.1 *Fine Tuned* XLM-RoBERTa (base sized model)

Train Dataset	Test Dataset	Classe	Accuracy	F1 Score	Precision	Recall
$ASSIN_{train}$	$ASSIN_{test}$	Entailment	0.75	0.28	0.33	0.25
		Paraphrase	0.56	0.24	0.33	0.18
		None	0.96	0.32	0.33	0.32
		General	0.89	0.89	0.89	0.89
$ASSIN_{train}$	$ASSIN2_{test}$	Entailment	0.86	0.46	0.5	0.43
		None	0.54	0.35	0.5	0.27
		General	0.70	0.69	0.73	0.70

Fonte: O autor.

Tabela 2.1: *Tabela 1 - Sua legenda aqui*

Referências

[Jurafsky e Martin 2021] Daniel Jurafsky e James H. Martin. *Speech and Language Processing*. https://web.stanford.edu/ jurafsky/slp3/, 2021 (citado na pg. 5).

[Sutskever *et al.* 2014] Ilya Sutskever, Oriol Vinyals e Quoc V. Le. *Sequence to Sequence Learning with Neural Networks*. 2014. arXiv: 1409.3215 [cs.CL] (citado na pg. 1).

[Vaswani *et al.* 2017] Ashish Vaswani *et al.* "Attention is all you need". Em: *Advances in neural information processing systems* 30 (2017) (citado nas pgs. 1, 3–5).

Índice Remissivo