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.

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 cujas saída e entrada são textos reais ou sintéticos.

1.1.1 A arquitetura *Transformer*

Introduzidos em 2017 no artigo entitulado "*Attention is All You Need*"[Vaswani *et al.*, 2017], os *Transformers* submetem a sequência de entrada a 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 anteriormente [Sutskever *et al.*, 2014] . 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*.

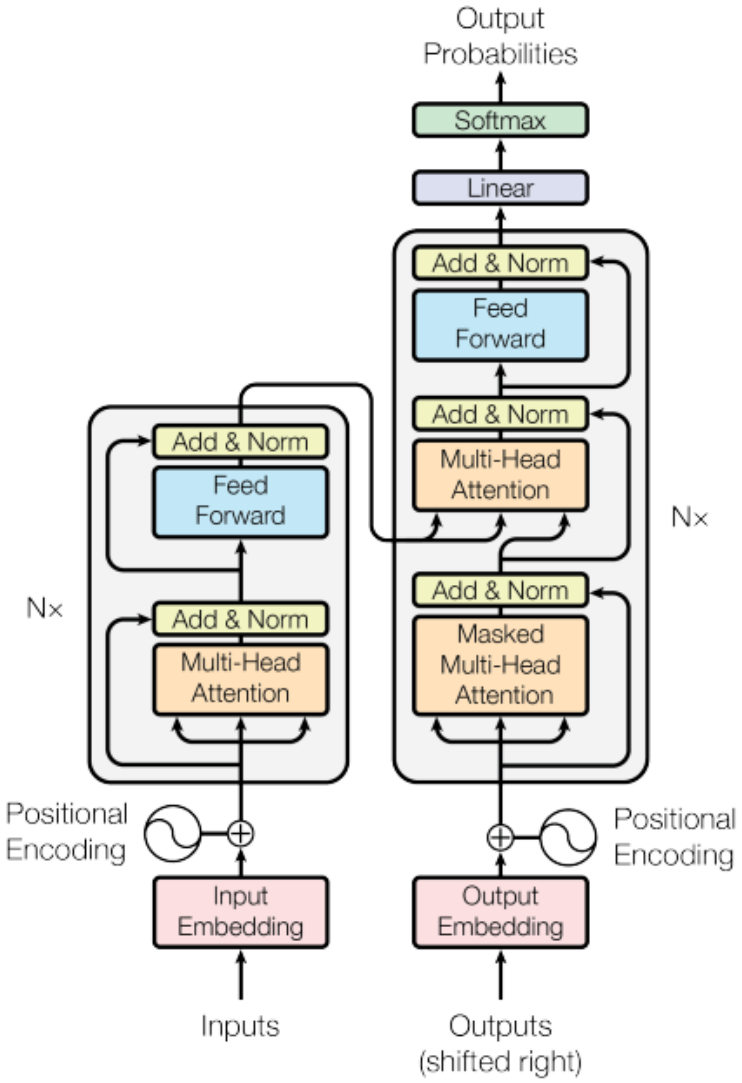

Figura 1.1: *A arquitetura Transformer*

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 da palavra ou sequência para a qual a atenção

$a_{q,k,v}$ está sendo calculada em seu contexto. Usando vetores q, o *Transformer* é capaz de comparar diferentes palavras e contextos em uma mesma sentença/entrada.

- **Key ($k \in \mathbb{R}^{d_{model}}$)**: é a representação 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 asm palavras de uma sentença/entrada.
- **Value ($v \in \mathbb{R}^{d_{model}}$)**: é a representação 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, o quaul é 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 2 como uma matriz em que cada linha representa a atenção de uma única *query* relativa a diversas *keys*.

Tanto em 1 quanto em 2, 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. Cada um

desses subespaços é denominado *head*. A atenção (2) é realizada paralelalemente para cada uma dessas projeções, gerando resultados com d_v dimensões cada. Tais resultados são concatenados e então projetados de volta para o espaço com d_{model} dimensões, gerando o resultado final do mecanismo de atenção *Multi-head*. (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 *heads* distintas gerando diferentes *queries*, *keys* e *values*, permite aos *Transformers* a representação de sentenças e textos segundo diversos aspectos e relações entre seus componentes. Sendo assim, o mecanismo de atenção *Multi Head* é fundamental para caracterizar essa arquitetura em sua notável capacidade de aprendizado de diferentes estruturas sintáticas e semânticas de um texto, justificando em parte o desempenho notável de modelos *Transformers* em tarefas de NLP. Os múltiplos atributos caracterizadores de linguagens escritas podem ser abstratamente representados pelas *heads*.

O *encoder*

O *encoder* mostrado à esquerda 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 idioma das entradas de modo que a *decoder* explorá-las 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 da saída produzida pelo *encoder*.

Os *tokens* que formam a saída do *decoder* são formados sequencialmente de modo que a cada instante, o *token* sendo produzido depende somente daqueles gerados nos instantes anteriores.Essa característica é chamada **autorregressão**.

Tal qual os *encoders*, os *decoders* são formados por 6 camadas conectadas. Cada camada do *decoder* aplica a atenção duas vezes, sendo a primeira a autoatenção cujas entradas são *queries, keys* e *values* produzidas na camada anterior do próprio *decoder*.

A **autorregressão** dos *Transformers* limita a autoatenção da camada de decodificação à exploração das relações entre uma posição do texto de entrada com somente o texto anterior a essa posição. Dessa forma, a autoatenção no *decoder* é realizada com *queries, keys* e *values* em que conexões de uma posição com posições posteriores são ocultadas e, por conseguinte, não aprendidas em tempo de treinamento.

A autoatenção no *decoder* é portanto denominada *Masked Multi-Head Attention* (Atenção *Multi-Head* Mascadada). Para garantir a ocultação de relações que violem a propriedade autoregressiva, posições em Q, K e V que as possibilitariam são transformadas em $-\infty$ durante o treinamento (Vaswani *et al.*, 2017).

A segunda aplicação da atenção no *decoder* utiliza as *keys* e *values* geradas pelo *encoder* e as *queries* produzidas pela última camada de decodificação. Nessa etapa, não há a ocultação de nenhuma parte de Q, K ou V. K resulta do *encoder* e, portanto, QK^T representa apenas relações entre a saída da última camada do *decoder* com a entrada do *encoder*. Sendo assim, nessa etapa a atenção explora apenas relações entre a saída do *encoder* e do *decoder* o que, em última instância, significa relacionar a entrada recebida pelo modelo como forma de *prompt* e codificada pelo *encoder* com o texto que ele produz como "resposta".

Os benefícios da atenção

As camadas de autoatenção no *encoder* e *decoder* conectam todas as posições de uma sentença entre si. Para isso, um número constante de operações é realizado para cada par de posições (Vaswani *et al.*, 2017). É válido verificar que a multiplicação QK^T em 3 requer

um número fixo de operações quaisquer sejam as linhas de Q e K multiplicadas. Consequentemente, relações entre a primeira e a segunda palavras de um texto são codificadas e decodificadas usando o mesmo número de operações que o mesmo processo realizado entre a primeira e palavras mais distantes.

Na arquitetura RNN (Jurafsky e Martin, 2021), utilizada em modelos de linguagem como Merity *et al.*, 2017, o mesmo processo é de complexidade $\mathcal{O}(n)$, em que n é distância no texto entre as palavras para as quais é calculada a relação ou dependência (Vaswani *et al.*, 2017). Essa complexidade revela dois aspectos característicos de modelos RNN. O primeiro refere-se ao tempo linearmente crescente para o cálculo das dependências internas de um texto. O segundo diz respeito ao número de operações a que a informação contida em uma palavra é submetida até que a sua dependência com outra seja calculada.

Em linhas gerais, no caso das RNNs, quanto maior o caminho citado, menor é a qualidade do aprendizado da dependência entre as palavras envolvidas. Isso se deve a problemas como o *vanishing* e *exploding gradients* (Hanin, 2018). As RNNs são projetadas para capturar dependências temporais entre as posições de um texto, as quais podem ser muito distantes. À medida que uma RNN é treinada com uma sequência longa de palavras, o gradiente pode desaparecer (tornar-se extremamente pequeno) ou "explodir"(tornar-se extremamente pequeno) devido às multiplicações consecutivas características do treinamento de redes neurais com *backpropagation*. Na prática, isso significa a perda de informações contidas nas dependências entre palavras distantes e, portanto, resulta na piora no aprendizadoque tais modelos adquirem sobre tais relações que são fundamentais para que atinjam um bom desempenho em tarefas de NLP.

Em suma, a complexidade $\mathcal{O}(1)$ do cálculo das dependências internas de um texto nos *Transformers* realizado pelas camadas de atenção sintetiza o benefício em utilizar tal arquitetura dos pontos de vista computacional e de desempenho, sendo este relacionado à sua alta capacidade em aprender relações entre partes distantes de uma sentença. Dessa forma, optou-se pela utilização de modelos baseados na arquitetura *Transformer* para a realização da tarefa de NLI nos conjuntos de dados descritos em [INSERIR LINK - DATASETS UTILIZADOS]. Esses modelos são descritos na seção seguinte.

1.1.2 XLM-RoBERTa

XLM-RoBERTa é o nome de um *Transformer* multilingual criado pelo *Facebook AI* introduzido em Conneau *et al.*, 2019. Trata-se de um modelo de linguagem mascarada pré-treinado em 100 idiomas que obtem resultados expressivos em tarefas de NLP como reconhecimento de entidades nomeadas [INSERIR LINK - DEFINIÇÃO DE NER], respostas a perguntas [INSERIR LINK - DEFINIÇÃO DE QA] e tarefas de classificação de textos

dentre as quais encontra-se o NLI.

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

[Conneau *et al.* 2019] Alexis Conneau *et al.* "Unsupervised cross-lingual representation learning at scale". Em: *CoRR* abs/1911.02116 (2019). arXiv: 1911.02116. URL: http://arxiv.org/abs/1911.02116 (citado na pg. 6).

[Hanin 2018] Boris Hanin. "Which neural net architectures give rise to exploding and vanishing gradients?" Em: 31 (2018). Ed. por S. Bengio *et al.* URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/13f9896df61279c928f19721878fac41-Paper.pdf (citado na pg. 6).

[Jurafsky e Martin 2021] Daniel Jurafsky e James H. Martin. *Speech and Language Processing*. https://web.stanford.edu/ jurafsky/slp3/, 2021 (citado na pg. 6).

[Merity *et al.* 2017] Stephen Merity, Nitish Shirish Keskar e Richard Socher. "Regularizing and optimizing lstm language models". Em: *arXiv preprint arXiv:1708.02182* (2017) (citado na pg. 6).

[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, 3–6).

Índice Remissivo