UNIVERSIDADE ESTADUAL DE CAMPINAS

Instituto de Matemática Estatística e Computação Científica

Tradução automática usando atenção

Projeto Computacional 2 MS960

27/11/2024

Felipe Costa Amaral - 249239

Murilo Felício Nascimento dos Santos - 236783

Conteúdo

1 Introdução

O problema da **tradução automática** consiste na utilização de métodos computacionais para tradução de texto e fala entre diferentes idiomas. Ele é um tema bastante antigo na computação e que teve seus primeiros passos ainda em meados da década de 1950 [4]. Desde então, vem evoluindo continuamente em capacidade e em relevância junto aos avanços da tecnologia. De fato, com a crescente globalização e o acesso a computadores e *smartphones*, plataformas de tradução automática, como o *Google Translate*, tornaram-se cada vez mais essenciais na superação das barreiras linguísticas, aproximando as pessoas tanto a nível pessoal quanto profissional.

Sobretudo na última década, a evolução da tradução automática foi particularmente perceptível na transição de modelos antigos baseados em regras explícitas ou puramente estatísticos para abordagens envolvendo a utilização de **redes neurais profundas**. Nesse sentido, destaca-se o desenvolvimento das redes neurais recorrentes (RNNs) e suas variações, como a *Long Short-Term Memory* (LSTM) e a *Gated Recurrent Unit* (GRU), além de modelos baseados no mecanismo de atenção e os mais recentes *Transformers* e *Large Language Models* (LLMs). Esses avanços representaram um marco, permitindo traduções mais precisas, contextuais e fluentes [2].

O presente trabalho tem por objetivo explorar o mecanismo de atenção [1] junto à LSTM [3] para a realização de uma tarefa de tradução automática simplificada: a conversão de datas em linguagem natural para uma linguagem padronizada YYYY-MM-DD.

2 Modelo: arquitetura e intuição

Para este trabalho, utilizamos uma arquitetura de redes neurais recorrentes com dois elementos fundamentais: a LSTM e o modelo de atenção.

2.1 Long Short-Term Memory (LSTM)

A **LSTM** é uma arquitetura de rede neural recorrente que tem por objetivo amenizar o desaparecimento do gradiente, problema já conhecido de redes neurais profundas, mas que também está presente em RNNs clássicas. Para o problema da tradução automática,

isso quer dizer que a rede tem dificuldade em memorizar palavras distantes e, portanto, tem "memória curta".

Intuitivamente, a estratégia que a LSTM utiliza para ampliar a "memória"da rede é a introdução de mecanismos que atuam como portões, determinando, a cada tempo, quanto do fluxo de informações deve ser armazenado, esquecido e passado na saída. O quão abertos ou fechados esses portões devem estar é aprendido durante o treinamento, permitindo que a rede aprenda a reter informações úteis a longo prazo e a descartar aquelas menos relevantes.

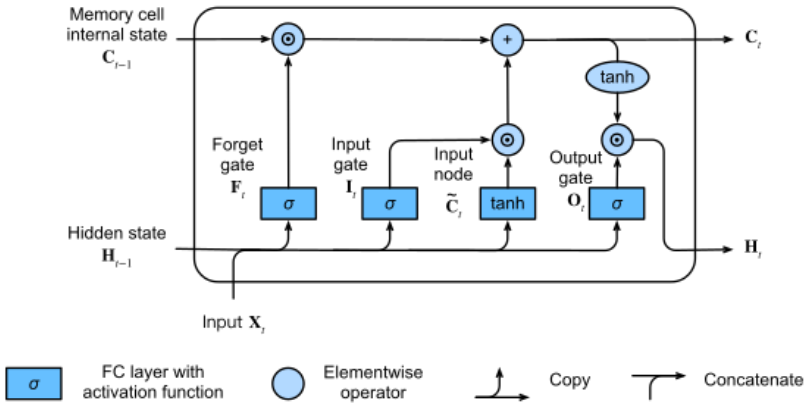

Figura 1: Esquema do funcionamento de uma célula LSTM [6].

2.2 Modelo de atenção

O **modelo de atenção** surgiu como uma evolução do modelo encoder/decoder. Este, por tentar sintetizar, durante o encoding, toda a informação do texto em somente uma ativação que seria então passada ao decoder, se tornava menos preciso com o aumento do tamanho da sequência de entrada.

No sentido de contornar esse problema, o modelo de atenção adota uma abordagem mais próxima da tradução humana, dando atenção a diferentes partes do texto de entrada enquanto gera cada palavra na saída. Assim, ele é capaz de apreender melhor informações contextuais específicas, ao invés de ter que analisar todo o texto de uma vez. De fato, utilizando o BLEU Score [5] como medida de precisão para os dois modelos, a curva que representa a RNN com atenção é capaz de permanecer alta enquanto a curva do encoder/decoder cai à medida em que o comprimento da seqência aumenta:

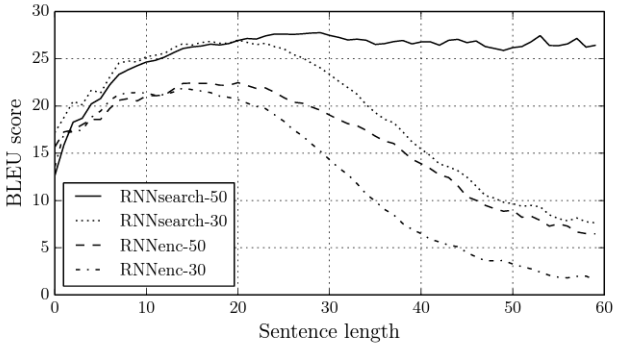

Figura 2: Gráfico BLEU-score × tamanho da sequência. RNNsearch são modelos com atenção e RNNenc são modelos encoder/decoder. Disponível em [1].

2.3 Panorma geral

Introduzidos os elementos fundamentais do modelo, passemos a uma análise da arquitetura utilizada no projeto.

Ela consiste, em primeiro lugar, numa rede bidirecional LSTM de encoding que tem a função de extrair features para cada palavra e suas proximidades. Essas features estão codificadas nos vetores $a^{<t>}$ que são a concatenação da ativação no tempo t da rede foward e da rede backward. Além disso, temos uma segunda LSTM, dessa vez unidirecional, de decoding responsável por receber os contextos gerados pela atenção e gerar a saída, isto é, a tradução do texto. Vamos denotar $s^{<t>}$ para a ativação dessa rede no tempo t. Como a atenção em cada tempo t deve depender também do que já foi traduzido anteriormente, utiliza-se $s^{<t-1>}$ no cálculo dos pesos de atenção. Veja a Figura 3.

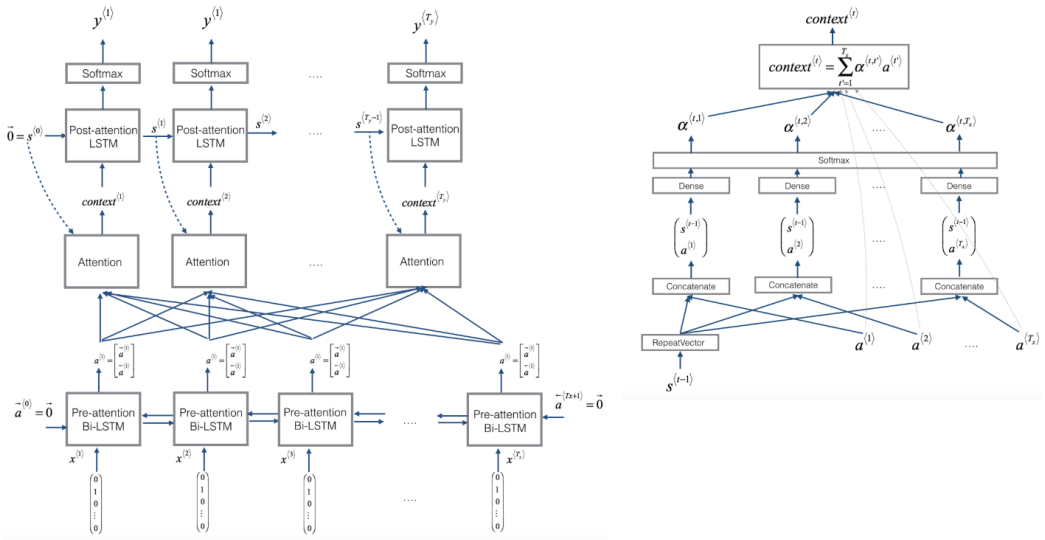

Figura 3: Arquitetura

Para o cálculo dos contextos tem cada tempo t, concatena-se cada um dos $a^{<t'>}$ com a ativação da rede de decoding $s^{<t-1>}$. Esses vetores são fornecidos a uma rede neural densa que deve aprender a relação entre o estado anterior da tradução e para onde deve ir a atenção da rede em seguida. Os resultados dessa rede densa são chamados de energias $e^{<t,t'>}$ e são fornecidas a uma softmax produzindo os pesos de atenção $\alpha^{<t,t'>}$. Por fim, calcula-se o contexto no tempo t via o somatório $\sum_{t'=1}^{T_x} \alpha^{<t,t'>} a^{<t'>}$, onde $\sum_{t'=1}^{T_x} \alpha^{<t,t'>} = 1$. Esse contexto é então fornecido como entrada para a rede de decoding.

2.4 Aspectos teóricos e práticos

Em síntese, do ponto de vista teórico, a utilização da LSTM permite que o modelo memorize informações a longo prazo ao passo que a atenção permite que o modelo compreenda informações contextuais mais específicas em cada tempo, se aproximando mais da forma de tradução humana. Do ponto de vista prático, a combinação desses fatores na arquitetura utilizada fornece uma rede que é mais precisa do que uma encoder-decoder clássica, mas que é mais computacionalmente custosa.

3 Resultados

A implementação da rede foi realizada em um notebook Python utilizando o framework Keras, conforme ilustrado no diagrama da Figura 4. Além dos hiperparâmetros de treinamento, o modelo possui outros três hiperparâmetros principais: a dimensão dos estados do encoder (u_E), a dimensão dos estados do decoder (u_D) e o número de neurônios (u_A) na camada densa do mecanismo de atenção, que utiliza uma função de ativação tanh.

Os treinamentos foram realizados em um conjunto de dados com 8.000 amostras, utilizando o otimizador Adam configurado com uma taxa de aprendizado de 0,001, $\beta_1 = 0,9$, $\beta_2 = 0,999$ e $\varepsilon = 10^{-7}$. A função de perda adotada foi a cross-entropy, e a performance do modelo foi avaliada em um conjunto de validação com 2.000 amostras.

Um total de cinco combinações de hiperparâmetros foram testadas:

- ($u_E = 32$, $u_D = 64$, $u_A = 10$, batch = 80),
- ($u_E = 64$, $u_D = 32$, $u_A = 10$, batch = 80),
- ($u_E = 16$, $u_D = 23$, $u_A = 10$, batch = 80),

- $(u_E = 32,\ u_D = 64,\ u_A = 10,\ \text{batch} = 800)$,
- $(u_E = 32,\ u_D = 64,\ u_A = 32,\ \text{batch} = 80)$.

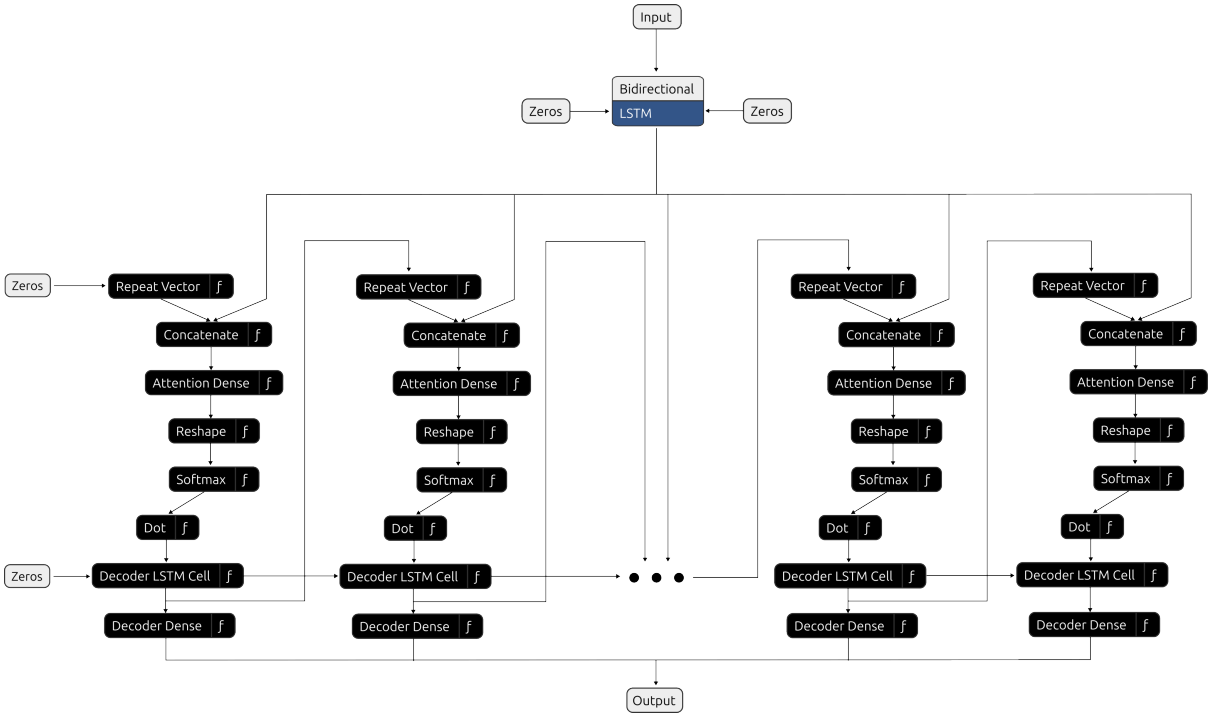

Figura 4: Diagrama do modelo implementado.

4 Discussões

Todas as combinações de hiperparâmetros testadas produziram bons resultados no conjunto de teste, como demonstram os gráficos nas Figuras 9 a 11. Observa-se, conforme o esperado, uma diferença significativa no desempenho entre os treinamentos realizados com batches de tamanho 800 e aqueles com batches de tamanho 80. No caso dos batches de tamanho 800, o desempenho de 60% foi alcançado apenas na época 100, enquanto os treinamentos com batches de tamanho 80 atingiram esse desempenho antes da vigésima época.

Além disso, alterações nos hiperparâmetros u_E, u_D e u_A não apresentaram impactos significativos no desempenho do modelo. Complementarmente, realizamos testes manuais utilizando dados mais diversos, incluindo datas futuras (após 2023) e muito antigas (antes de 1970). Nessas situações, o modelo não conseguiu gerar respostas corretas. Para

investigar esse comportamento, construímos um histograma das frequências em que cada ano aparece no dataset, apresentado na Figura 5.

O histograma revela que todas as datas do dataset possuem anos no intervalo entre 1970 e 2023. Consequentemente, o modelo não conseguiu generalizar para datas fora desse intervalo, explicando os resultados obtidos nos testes manuais. Tanto essa limitação quanto a ausência de impacto significativo das alterações nos hiperparâmetros sugerem um possível grau de *overfitting* no modelo.

Ademais, com base nos mapas de atenção apresentados nas Figuras 6, 7 e 8, foi possível verificar o funcionamento adequado do mecanismo de atenção.

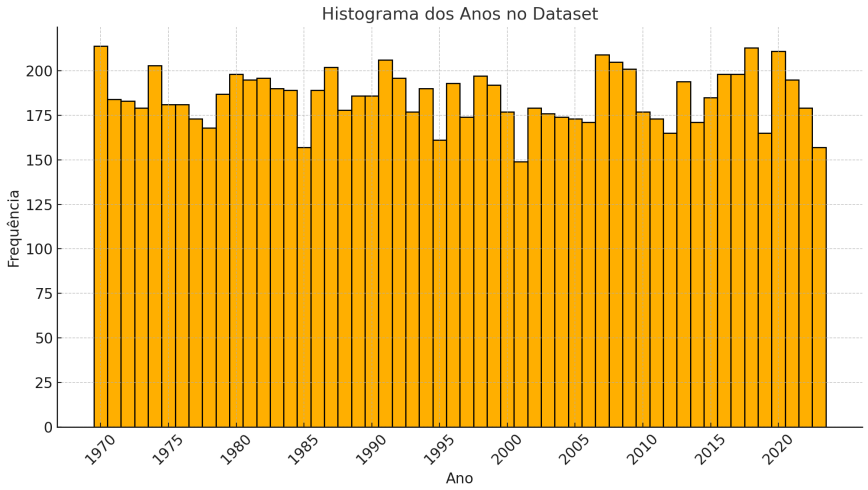

Figura 5: Histograma que mostra a frequência em que cada ano aparece no dataset

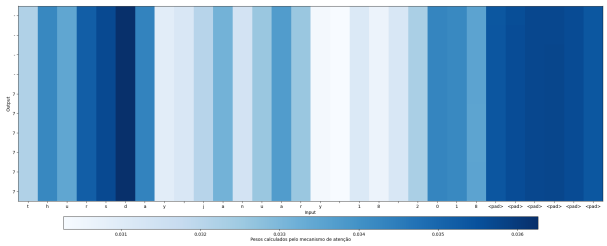

Figura 6: Mapa de atenção da sentença "thursday january 18 2018"na época 0 modelo ($u_E = 32$, $u_D = 64$, $u_A = 10$, batch= 80)

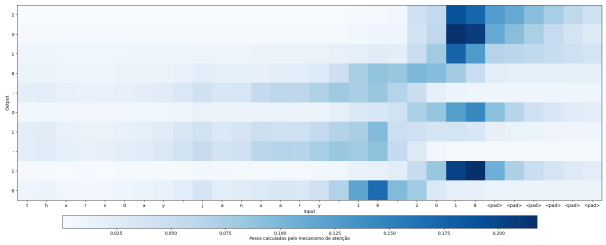

Figura 7: Mapa de atenção da sentença "thursday january 18 2018"na época 10 modelo ($u_E = 32$, $u_D = 64$, $u_A = 10$, batch= 80)

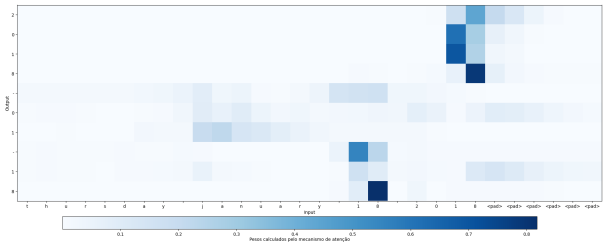

Figura 8: Mapa de atenção da sentença "thursday january 18 2018"na época 100 modelo ($u_E = 32$, $u_D = 64$, $u_A = 10$, batch= 80)

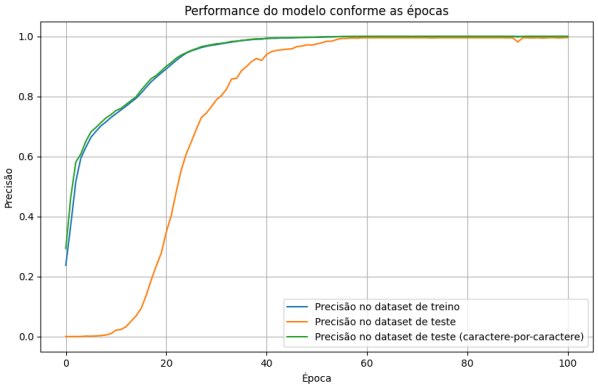

Figura 9: $u_E = 16$, $u_D = 23$, $u_A = 10$, batch= 80

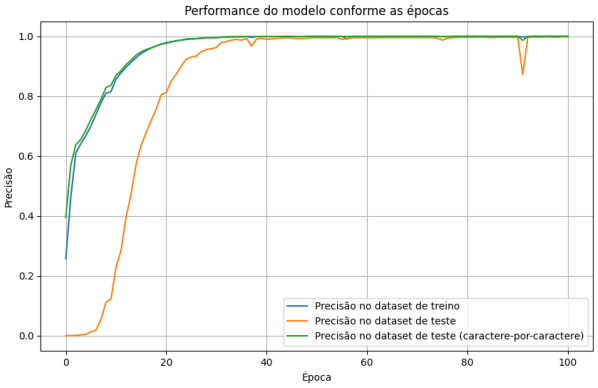

Figura 10: $u_E = 32$, $u_D = 64$, $u_A = 10$, batch= 80

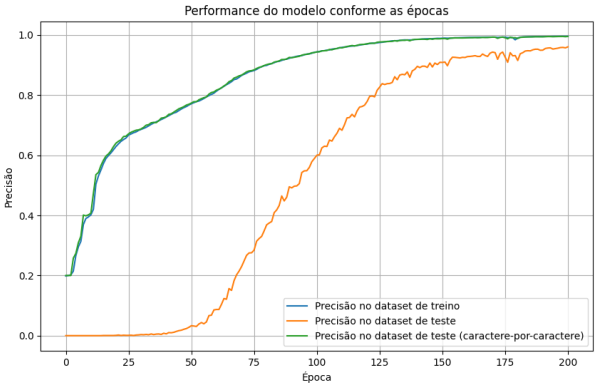

Figura 11: $u_E = 32$, $u_D = 64$, $u_A = 10$, batch= 800

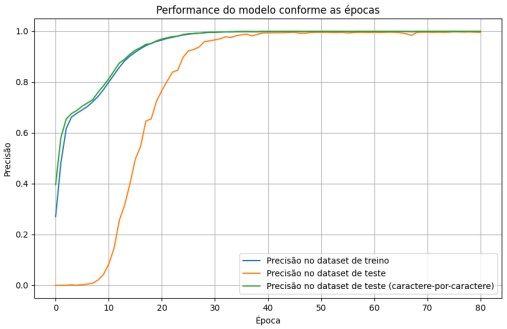

Figura 12: $u_E = 64$, $u_D = 32$, $u_A = 32$, batch= 80

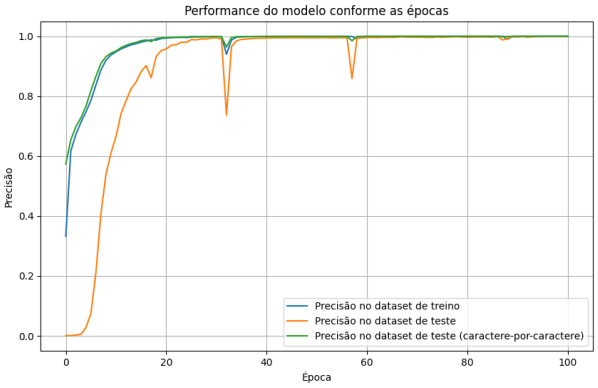

Figura 13: $u_E = 64$, $u_D = 32$, $u_A = 10$, batch= 80

4.1 Sugestões

A primeira sugestão para melhorar o modelo seria a expansão do dataset para um mais representativo no que diz respeito aos anos. Isso é importante dado que o modelo, em 2024, já está obsoleto nesse sentido.

Ademais, com o objetivo de aumentar a capacidade de generalização do modelo e reduzir o overfitting, poderiam ser aplicadas técnicas de regularização como L2 e dropout. Outra alternativa seria o uso de data augmentation.

Como a sequência de saída possui tamanho 10, relativamente pequeno, a utilização de uma LSTM no encoder pode ser desnecessária. Substituí-la por uma GRU pode ser benéfico, pois reduziria o número de parâmetros do modelo, o que possivelmente contribuiria para mitigar o overfitting.

Referências

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. **Neural Machine Translation by Jointly Learning to Align and Translate**. *arXiv preprint arXiv:1409.0473*, 2014.

[2] Isaac Caswell and Bowen Liang. **Recent Advances in Google Translate**. *Google Research Blog*, June 8 2020.

[3] Sepp Hochreiter and Jürgen Schmidhuber. **Long short-term memory**. *Neural Computation*, 9(8):1735–1780, 1997.

[4] John Hutchins. **The first public demonstration of machine translation: the Georgetown-IBM system, 7th January 1954**. *Hutchins Web*, March 2006. Archived from the original on 2007-10-21.

[5] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. **BLEU: a method for automatic evaluation of machine translation**. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[6] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. **Long Short-Term Memory (LSTM)**. `https://d2l.ai/chapter_recurrent-modern/lstm.html`. Accesso em: 2024-11-27.