UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E COMPUTAÇÃO CIENTÍFICA
DEPARTAMENTO DE MATEMÁTICA APLICADA

NICOLAS TOLEDO DE CAMARGO

Ajuste fino de um modelo de linguagem para análise de sentimentos financeiros.

Campinas
24/06/2024

NICOLAS TOLEDO DE CAMARGO

Ajuste fino de um modelo de linguagem para análise de sentimentos financeiros.

Monografia apresentada ao Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas como parte dos requisitos para obtenção de créditos na disciplina Projeto Supervisionado, sob a orientação do(a) Prof. João B. Florindo.

Resumo

Este relatório aborda a aplicação da análise de sentimentos no contexto financeiro, utilizando o modelo de linguagem LLaMa 2 ajustado para classificar notícias. A análise de sentimentos é crucial para entender o tom de mensagens, classificando-as como positivas, negativas ou neutras, possibilitando a aplicação em diversos setores, incluindo finanças. Ajustamos o modelo com o banco de dados *FinancialPhraseBank*, que contém classificações de notícias financeiras em inglês, anotadas por especialistas. Para otimizar o desempenho do modelo, utilizamos técnicas de *fine-tuning* em *Python* com as bibliotecas *transformers*, *bitsandbytes*, *trl* e *peft*. Testamos diferentes taxas de aprendizado, temperaturas e épocas de treinamento, identificando que a taxa de 2×10^{-4} e temperaturas mais baixas (0.1 e 1) proporcionaram os melhores resultados. Inicialmente, o modelo original apresentou baixa acurácia e *F1-score*, 0.389 e 0.333 respectivamente, mas após o ajuste fino, conseguimos melhorar significativamente as métricas, atingindo uma acurácia e *F1-score* de 0.867. Esses resultados validam a importância do *fine-tuning* em modelos de linguagem para tarefas específicas e destacam o potencial da análise de sentimentos como uma ferramenta auxiliar na tomada de decisões no mercado financeiro. Futuros trabalhos podem explorar outras configurações de hiperparâmetros, tratar diferentes idiomas, estudar outros modelos ou expandir a aplicação para outros mercados e tipos de dados.

Abstract

This report addresses the application of sentiment analysis in the financial context, utilizing the LLaMa 2 language model fine-tuned to classify news. Sentiment analysis is crucial for understanding the tone of messages, classifying them as positive, negative, or neutral, enabling applications across various sectors, including finance. We fine-tuned the model using the *FinancialPhraseBank* dataset, which contains classifications of financial news in English, annotated by experts. To optimize the model's performance, we employed *fine-tuning* techniques in *Python* with the *transformers*, *bitsandbytes*, *trl*, and *peft* libraries. We tested different learning rates, temperatures, and training epochs, identifying that a learning rate of 2×10^{-4} and lower temperatures (0.1 and 1) provided the best results. Initially, the original model exhibited low accuracy and *F1-score*, 0.389 and 0.333 respectively, but after fine-tuning, we significantly improved the metrics, achieving an accuracy and *F1-score* of 0.867. These results validate the importance of *fine-tuning* language models for specific tasks and highlight the potential of sentiment analysis as an auxiliary tool in financial market decision-making. Future works can explore other hyperparameter configurations, address different languages, study other models, or expand the application to other markets and data types.

Conteúdo

1 Introdução

A análise de sentimentos é um processo que envolve a classificação do tom de uma mensagem, indicando se ela possui uma conotação positiva, negativa ou neutra. Esta forma de avaliação representa uma valiosa fonte de informação, com aplicações em uma ampla variedade de setores, auxiliando na tomada de decisões estratégicas.

Neste relatório, exploraremos uma aplicação específica no âmbito financeiro, focando na classificação de notícias. A motivação por trás dessa abordagem reside no fato de que, entre diversos fatores, o sentimento presente no mercado é uma consideração crucial para os acionistas ao investirem seus recursos.

Adotaremos uma metodologia de *deep learning*, realizando um ajuste fino (*fine-tuning*) do modelo de linguagem LLaMa 2, desenvolvido pela META.

2 Trabalhos relacionados

Bhumika Gupta et al. (2017) aplicam a análise de opiniões de usuários da rede social X (Twitter) por meio de modelos de aprendizado de máquinas. O texto discorre passo a passo desde o processo de extração e tratamento de dados até o treinamento de modelos SVM, redes neurais, *ensemble*, entre outros. Conclui com o desempenho de cada um deles neste contexto.

Kostadin Mishev et al. (2020) realizam experimentos para classificar sentimentos em notícias financeiras, utilizando desde abordagens lexicográficas até alguns modelos de linguagem, avaliando o desempenho e eficiência de cada um.

Mayur Wankhade; Annavarapu Chandra Sekhara Rao e Chaitanya Kulkarni (2022) discutem o aumento da relevância do estudo de análise de sentimentos, revisando diversas metodologias, incluindo abordagens lexicográficas, estatísticas, por aprendizado de máquinas, entre outras, além de apresentar seus resultados e desafios. Também destacam diversos contextos de aplicação, como satisfação de consumidores, pesquisas de *marketing*, setor financeiro e de saúde.

Biodoumoye George Bokolo e Qingzhong Liu. (2023) realizam um estudo sobre análise de sentimentos de posts com foco em monitorar comportamentos depressivos. Além de treinar modelos usuais de aprendizado de máquinas, também implementam *deep*

learning, com variações do modelo de linguagem em larga escala BERT. Concluem que o modelo de linguagem obteve resultados melhores para a tarefa.

Michele Costola et al. (2023) apresentam um trabalho que examina a relação entre notícias durante a pandemia de COVID-19 e seus impactos nas expectativas do mercado. Foram analisadas notícias sobre o COVID-19 durante um período de 2020 utilizando um modelo modificado do BERT para análise de sentimentos financeiros. Seus resultados indicam uma relação significativa entre a classificação dos sentimentos e os movimentos do mercado.

Boyu Zhang et al. (2023) apontam que o objetivo principal de LLMs pré-treinados não é realizar análise de sentimentos, especialmente no contexto financeiro, o que leva a resultados precários. Eles propõem um *framework* com um módulo contendo um LLM instruído para atuar como um preditor de sentimentos e outro módulo de geração aumentada de recuperação (RAG), que fornece contexto adicional de fontes confiáveis para o LLM utilizar em suas predições. Comparado a modelos tradicionais e alguns LLMs, este método apresentou melhorias de desempenho.

Kelvin Du et al. (2024) reconhecem a proficiência de LLMs em raciocinar sobre diversas tarefas complexas, mas levantam uma questão: quão bem esses modelos conseguem raciocinar no contexto da análise de sentimentos financeiros? O estudo avalia o raciocínio do modelo em relação a atributos financeiros, fazendo o modelo explicar o sentimento predito relacionando-o a atributos semânticos, numéricos, temporais, comparativos e fatores de risco presentes nos textos. Os experimentos destacam especialmente a dificuldade dos modelos em explicar atributos numéricos e comparativos.

3 Revisão teórica

3.1 Transformer

3.1.1 Introdução

Por algum tempo, os modelos mais populares para processamento de séries temporais e de linguagem natural baseavam-se em redes neurais recorrentes e convolucionais, conectadas por mecanismos de atenção (Bahdanau et al., 2014); até que, em 2017, pesquisadores da Google apresentaram uma arquitetura inovadora no artigo *Attention Is All You Need* (Ashish Vaswani et al., 2017.), chamada Transformer.

Este novo modelo elimina a necessidade de redes recorrentes ou convolucionais, focando exclusivamente no mecanismo de atenção. A motivação por trás desse método era resolver problemas encontrados em modelos anteriores, como alto custo computacional, desaparecimento ou explosão do gradiente e perda de informações de passos anteriores.

O artigo demonstrou que o Transformer alcançou resultados superiores e com menor custo computacional em comparação com modelos anteriores, marcando um avanço significativo no processamento de linguagem natural.

3.1.2 Encoder-Decoder

O *encoder* (parte esquerda da Figura 3) extrai as características do *input* original, capturando o significado das palavras, suas posições na frase e as relações entre elas. O *output* do *encoder* alimenta a atenção cruzada no *decoder* (parte direita da Figura 3).

O papel do *decoder* é gerar a frase final com base no que foi previamente gerado e nas características do *input* original. Começando com o *token* [SOS] (*Start Of Sentence*), o *decoder* gera os *tokens* sucessivamente com base em suas probabilidades de serem as próximas palavras, continuando até gerar o *token* [EOS] (*End Of Sentence*).

3.1.3 Arquitetura

Embedding

Para processar computacionalmente uma frase, é utilizado o conceito de *embedding*. Cada palavra da frase é mapeada para um *token* em um dicionário, e cada *token* é representado por um vetor específico criado por pesos, processo exemplificado na Figura 4.

Encoding posicional

Ao realizar o *embedding*, apenas as palavras são representadas, sem informações sobre suas posições na frase, o que é essencial para o processamento de linguagem natural.

Então, antes de alimentar o modelo com os dados, é necessário adicionar às representações de *embedding* a posição de cada palavra. Isso é feito adicionando-se um vetor pré-calculado para cada posição.

A fórmula para calcular esses valores precisa levar em consideração valores únicos para cada posição e ser eficaz para frases longas. No artigo original a formula para posição é baseada em senos e cossenos:

$$PE_{pos,2i} = sin(\frac{pos}{10000^{\frac{2i}{d}}})$$
$$PE_{pos,2i+1} = cos(\frac{pos}{10000^{\frac{2i}{d}}})$$

Auto-Atenção

O mecanismo de auto-atenção captura as relações entre as palavras. No Transformer, cada palavra possui vetores de consulta (*query*), chave (*key*) e valor (*value*), obtidos por meio do produto entre pesos treinados e a representação de *embedding*. A atenção é calculada usando esses vetores, resultando em uma medida da relação entre cada palavra:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d}})V,$$

em que Q, K e V são as matrizes com os *querries*, *keys* e *values* de todas as palavras da frase. O resultado dessa operação é uma medida de relação entre cada palavra, exemplificado na Figura 5.

Atenção Multi-Cabeça

A atenção multi-cabeça (Figura 6) é um processo paralelizável que calcula várias atenções com diferentes pesos entre as cabeças para Q, K e V. Cada cabeça tenta extrair diferentes significados das palavras, e os resultados são concatenados e multiplicados por outra matriz de pesos, antes de serem passados para a próxima camada.

Atenção Multi-Cabeça com Mask

O Transformer é um modelo causal, ou seja, ao prever a próxima palavra, apenas as informações das palavras anteriores são consideradas.

Portanto, uma *mask* é aplicada na atenção após o *embedding* do *decoder*, zerando os valores de atenção acima da diagonal principal. Isso impede que o modelo veja palavras futuras durante o treinamento.

Add & Norm

Add se refere à conexão residual entre camadas, que é realizada somando-se ponto a ponto os tensores (Kaiming He; Xiangyu Zhang; Shaoqing Ren; and Jian Sun., 2016.). Após essa conexão residual, é aplicada uma *layer normalization* (Jimmy Lei Ba; Jamie Ryan Kiros; and Geoffrey E Hinton., 2016.), que é a normalização com relação a cada item do *batch* (Figura 7).

Feed Forward

Esta parte consiste em uma rede densa totalmente conectada, neste caso, com duas camadas lineares e uma função de ativação ReLU entre elas:

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2.$$

3.2 Llama 2

3.2.1 Introdução

LLaMa 2 é o modelo LLM desenvolvido por pesquisadores da empresa META (Hugo Touvron et al., 2023 .). Este é um projeto *open-source*, que permite *fine-tuning* e é focado na função de dialogar.

O modelo foi treinado em diferentes métricas, como programação, lógica, conhecimentos gerais e matemática, e apresentou melhor performance em relação a outros projetos *open-source* até o momento.

3.2.2 Decoder-only

Diferente do Transformer, o LLaMa 2 se desfaz do *encoder* e possui apenas uma parte de *decoder*, sendo chamado de um modelo *decoder-only* (Figura 8).

Modelos *decoder-only* são eficientes na função específica de gerar próximas palavras baseadas no contexto anterior. Outros exemplos de *decoder-only* são o Gemini da Google e a família GPT da OpenAI.

3.2.3 Modificações de arquitetura

Normalização RMS

Esta técnica foi pensada por Biao Zhang e Rico Sennrich (2019.). A hipótese é que na verdade o mais importante na normalização é o dimensionamento. Então eles propuseram a RMSNorm, uma normalização que não realiza a centralização dos dados, porém dimensiona-os com base na estatística RMS (Figura 9).

Este método apresentou bons resultados e é mais eficiente que a normalização convencional, já que apenas calcula uma estatística ao invés de duas.

Encoding posicional rotativo

Este método de *encoding*, proposto por Jianlin Su et al. (2021.), é fundamentado em matrizes de rotação e oferece contribuições significativas para a representação

posicional. Ele abrange tanto a posição absoluta na frase quanto a posição relativa entre as palavras. Além disso, apresenta a propriedade de decaimento do produto interno à medida que as posições relativas aumentam, uma característica desejável no cálculo da auto-atenção, já que palavras mais próximas tendem a ter relações mais fortes do que palavras muito distantes.

Grouped Multi-Query Attention com KV Cache

Durante a inferência, muitos cálculos redundantes ou desnecessários surgem. Por exemplo, ocorre o cálculo das relações entre uma palavra previamente gerada e as palavras subsequentes, o que não é desejado no contexto do modelo causal. Além disso, há produtos internos entre os *tokens* já gerados que podem ser armazenados após o primeiro cálculo.

Uma técnica para contornar esses problemas é o uso do KV *Cache*, um método que armazena os valores previamente calculados e opera apenas com o último *token* gerado. Isso resulta em uma redução significativa no custo computacional.

O *Grouped Multi-Query Attention* (Joshua Ainslie; James Lee-Thorp; Michiel de Jong et al, 2023.) é uma variação na computação da atenção multi-cabeças. No método original, cada *query* tinha seus próprios *keys* e *values*. Na técnica proposta, as *queries* são divididas em grupos, e cada grupo compartilha uma única *key* e um único *value* entre seus elementos (Figura 11). Essa abordagem visa reduzir o custo computacional sem comprometer drasticamente a qualidade do modelo.

Feed Foward SwiGLU

Agora, a rede densa adota a função de ativação SwigLU (Figura 10) em vez da ReLU. O artigo (Noam Shazeer, 2020.) apresenta empiricamente que esta função produziu resultados superiores quando testada no modelo Transformer.

$$SwigLU(x) = Swish(xW) \cdot xV, \text{ W e V matrizes de pesos;}$$

$$Swish(x) = x \cdot Sigmoid(\beta x), \ \beta \text{ escalar;}$$

4 Metodologia

4.1 Fonte de dados

Os dados utilizados para treino e teste são estruturados em uma tabela que inclui as notícias e seus respectivos sentimentos (positivo, neutro ou negativo). Esse conjunto de dados, denominado *FinancialPhraseBank*, consiste em 4837 classificações de notícias financeiras em inglês. Essas classificações foram realizadas por especialistas em finanças, incluindo pesquisadores e estudantes de mestrado de *Aalto University School of Business*.

4.2 Fine tuning

A versão aplicada do LLaMa 2 neste projeto possui 7 bilhões de parâmetros. Utilizamos a biblioteca *transformers* para carregar o modelo e realizar inferências.

Para otimizar a eficiência computacional, empregamos o *framework bitsandbytes*, que implementa técnicas de quantização para otimizar operações e reduzir o uso de memória.

Para o treino, utilizamos as bibliotecas *trl* e *peft*, que fornecem um treinador supervisionado para *fine-tuning* e implementam uma abordagem que atua apenas sobre um número limitado de parâmetros do modelo, reduzindo assim o custo computacional e evitando alterações drásticas no modelo pré-treinado.

Com esses *frameworks* em mãos, o treinamento supervisionado foi realizado com base em *prompts* para o LLaMa 2 interpretar. O *prompt* de treinamento contém a notícia com sua classificação, enquanto o *prompt* de teste contém apenas a notícia, esperando que o modelo gere um *token* de classificação. Arbitrariamente também foi decidido que se o modelo gerar um *token* inesperado, a classificação se dará como 'neutro'.

Padrão de prompt de treinamento

Analyze the sentiment of the news headline enclosed in square brackets, determine if it is positive, neutral, or negative, and return the answer as the corresponding sentiment label "positive"or "neutral"or "negative".

$[HEADLINE] = SENTIMENT$

Padrão de prompt de teste

Analyze the sentiment of the news headline enclosed in square brackets, determine if it is positive, neutral, or negative, and return the answer as the corresponding sentiment label "positive"or "neutral"or "negative".

$[HEADLINE] =$

Dentre os hiperparâmetros modificáveis, realizamos experimentos alterando a temperatura do modelo, o número de épocas e a taxa de aprendizado. A temperatura regula o quão 'criativo' será o *output*, sendo que temperaturas menores implicam em respostas menos variadas e temperaturas maiores resultam em respostas mais diversificadas. O método consiste em modificar o peso da saída *softmax* ao gerar o *token* (Figura 12). O número de épocas se refere ao número máximo de iterações para o treinamento, enquanto a taxa de aprendizado afeta a escala de modificação dos parâmetros em cada iteração.

Para todas as configurações, foram testadas as temperaturas 0.1, 1 e 10, para cobrir temperatura baixa, padrão e alta. Começamos a testar a taxa de aprendizado com o mesmo número de épocas e, para a taxa que obteve resultados melhores, variamos o número de épocas de treino. Projeto disponível em https://www.kaggle.com/code/nicolastdc/ms777-llama2-fine-tune.

5 Resultados

5.1 Modelo original

Temperatura	Acurácia	F1-Score
0.1	0.378	0.273
1	**0.389**	**0.333**
10	0.321	0.197

Tabela 1: Resultados de teste para modelo original.

5.2 Modelos aperfeiçoados

Taxa de aprendizado	Épocas	Acurácia	F1-Score
2e-6	1	0.386	0.28
2e-4	**1**	**0.809**	**0.807**
2e-2	1	0.333	0.167
2e-6	3	0.436	0.38
2e-4	**3**	**0.842**	**0.843**
2e-2	3	0.333	0.167

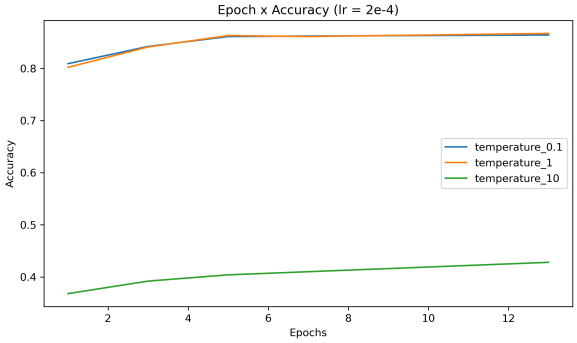
Tabela 2: Resultados de teste para temperatura = 0.1.

Taxa de aprendizado	Épocas	Acurácia	F1-Score
2e-6	1	0.393	0.34
2e-4	**1**	**0.802**	**0.8**
2e-2	1	0.333	0.167
2e-6	3	0.401	0.353
2e-4	**3**	**0.841**	**0.843**
2e-2	3	0.333	0.167

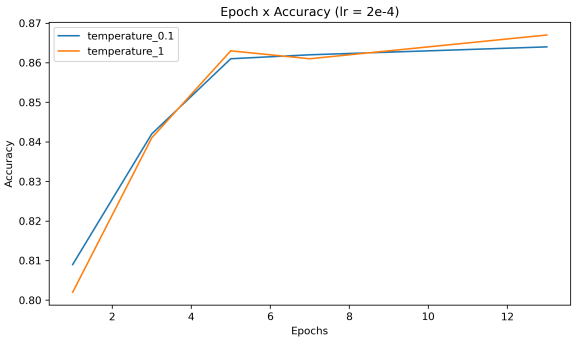
Tabela 3: Resultados de teste para temperatura = 1.

Taxa de aprendizado	Épocas	Acurácia	F1-Score
2e-6	1	0.336	0.223
2e-4	**1**	**0.368**	**0.287**
2e-2	1	0.333	0.167
2e-6	3	0.344	0.23
2e-4	**3**	**0.392**	**0.323**
2e-2	3	0.333	0.167

Tabela 4: Resultados de teste para temperatura = 10.

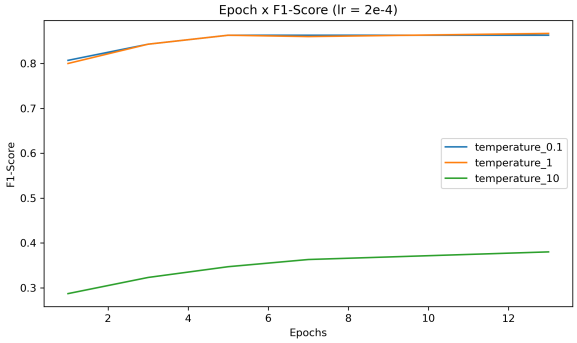


(a) Acurácia por época, todas temperaturas.

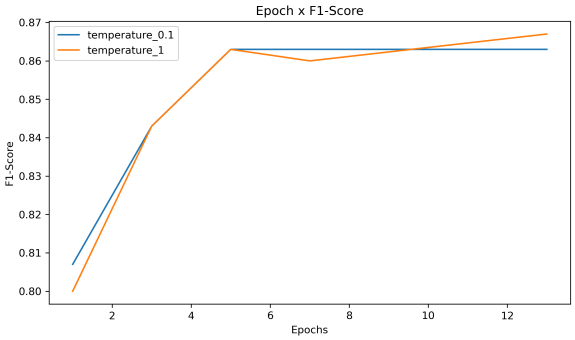


(b) Acurácia por época, temperaturas 0.1 e 1.

Figura 1: Gráficos de acurácia por época para taxa de aprendizado $2 \cdot 10^{-4}$.



(a) F1-Score por época, todas temperaturas.



(b) F1-Score por época, temperaturas 0.1 e 1.

Figura 2: Gráficos de *F1-score* por época para taxa de aprendizado $2 \cdot 10^{-4}$.

6 Discussão

Como demonstrado na Tabela 1, o modelo original mostra-se inadequado para esta tarefa, apresentando baixa acurácia e *F1-score*, independentemente da temperatura. Isso ocorre porque o LLaMa 2 não foi treinado com foco nesse objetivo específico, resultando na incapacidade de capturar as nuances da análise de sentimentos, especialmente no contexto de notícias financeiras, que demandam uma análise ainda mais específica.

Ao testarmos diferentes taxas de aprendizado para o mesmo número de épocas (Tabelas 2, 3 e 4), fica evidente que a taxa de 2×10^{-4} se destaca. Por outro lado, a taxa de 2×10^{-2} falha em convergir o modelo, resultando na geração de *tokens* inesperados e na classificação de todos os exemplos como 'neutros', este comportamento de gerar muitos *tokens* inesperados se repetiu em configurações que ficaram longe da convergência. Enquanto isso, a taxa de 2×10^{-6} mostra alguma melhoria, mas sua convergência parece ser significativamente mais lenta em comparação com a taxa de 2×10^{-4}.

Portanto, optamos pela taxa de 2×10^{-4} para testes adicionais. Inicialmente, fica claro que uma temperatura alta não produz resultados satisfatórios (Figuras 1a e 2a), uma vez que buscamos resultados objetivos e consistentes. Ao examinarmos temperaturas mais baixas (Figuras 1b e 2b), observamos que, além destas configurações proporcionarem resultados semelhantes, há uma diminuição significativa na melhoria por volta da quinta época.

7 Conclusão

Neste relatório, exploramos a aplicação da análise de sentimentos no contexto financeiro, utilizando o modelo de linguagem LLaMa 2 ajustado para classificar notícias, fornecendo a possibilidade de uma ferramenta útil para investidores e acionistas que desejam monitorar o sentimento do mercado como parte de suas estratégias de investimento.

Os experimentos demonstraram que o modelo original LLaMa 2, sem ajustes, não era adequado para a tarefa específica de análise de sentimentos de notícias financeiras, apresentando baixa acurácia e F1-*score*, com acurácia de 0.389 e F1-*score* de 0.333. Porém, com o ajuste fino, ao testarmos diferentes taxas de aprendizado, temperaturas e épocas, pudemos identificar que existem combinações de parâmetros que melhoram significativamente as métricas, sendo que com a melhor combinação obtivemos o F1-*score* e acurácia de 0.867.

Os resultados indicam que, ao ajustar adequadamente os hiperparâmetros, o modelo pode capturar de forma mais precisa as nuances da análise de sentimentos neste contexto. Em nosso caso, a taxa de 2×10^{-4} forneceu os melhores resultados, junto de temperaturas mais baixas (1 ou 0.1) para obter consistência e objetividade, além disso, o melhor resultado foi obtido com treze épocas, porém a melhora relevante em desempenho acaba por volta de cinco épocas de treinamento.

Esta abordagem não só valida a importância do *fine-tuning* em modelos de linguagem para tarefas específicas, mas também destaca o possível potencial da análise de sentimentos como uma ferramenta para a tomada de decisões no mercado financeiro, visto que estes modelos estão sendo capazes de produzir análises cada vez mais precisas. Futuros trabalhos podem explorar outras configurações de hiperparâmetros, tratar outros idiomas, estudar outros modelos ou até expandir a aplicação para diferentes mercados e tipos de dados, possibilitando estudar ainda mais a utilidade dos modelos de linguagem.

A Apêndices

A.1 Diagramas Transformer

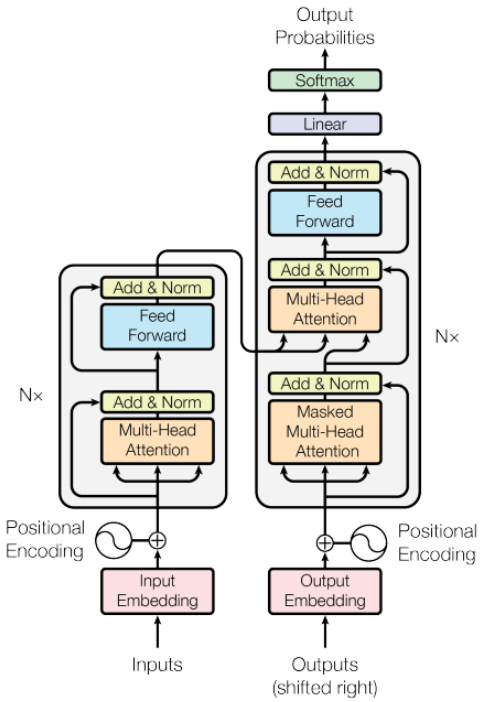


Figura 3: Arquitetura do Transformer (Ashish Vaswani et al., 2017.).

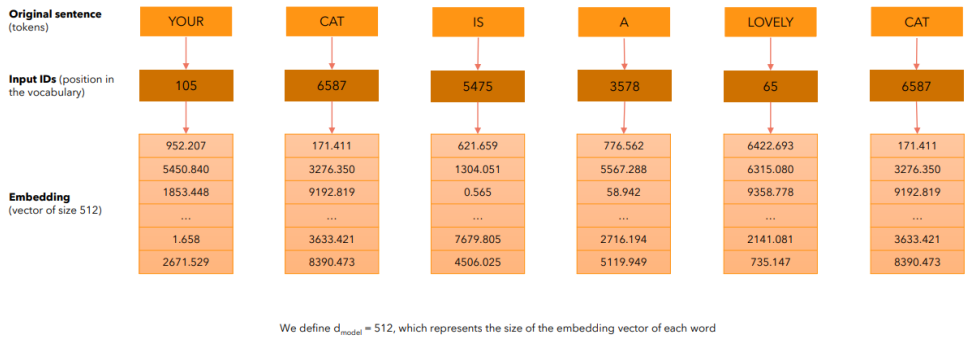
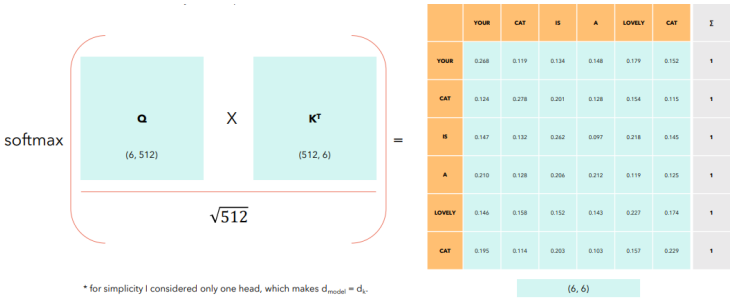
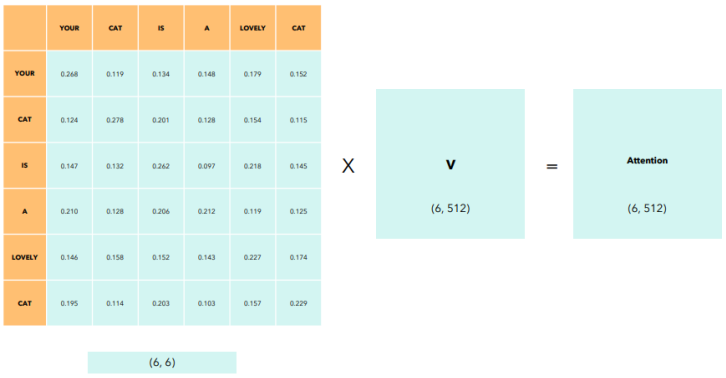


Figura 4: Exemplo de Input Embedding (Umar Jamil, 2023.).



(a)



(b)

	YOUR	CAT	IS	A	LOVELY	CAT
YOUR	0.268	0.119	0.134	0.148	0.179	0.152
CAT	0.124	0.278	0.201	0.128	0.154	0.115
IS	0.147	0.132	0.262	0.097	0.218	0.145
A	0.210	0.128	0.206	0.212	0.119	0.125
LOVELY	0.146	0.158	0.152	0.143	0.227	0.174
CAT	0.195	0.114	0.203	0.103	0.157	0.229

(c)

Figura 5: Cálculo auto-atenção.(Umar Jamil, 2023.)

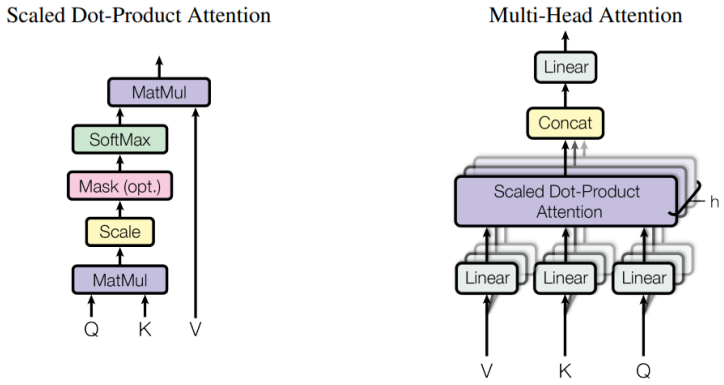


Figura 6: Mecanismo de atenção (Ashish Vaswani et al., 2017.).

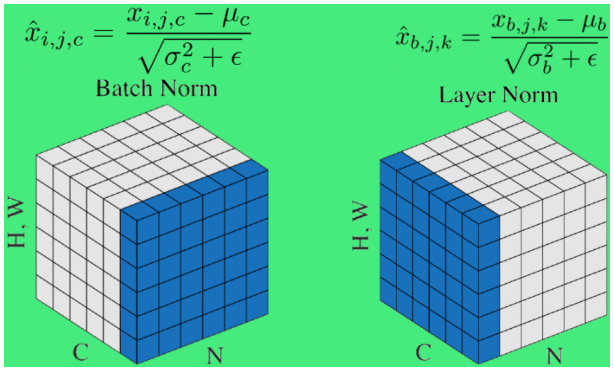


Figura 7: Batch Norm x Layer Norm (Vladimir Ilievski, 2023.).

A.2 Diagramas LLaMa 2

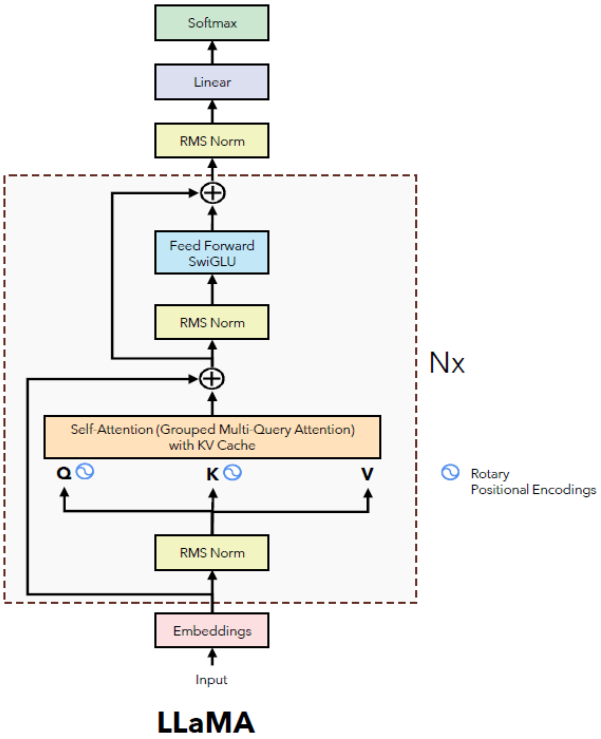


Figura 8: Arquitetura do LLaMa 2 (Umar Jamil, 2023.).

$$\bar{a}_i = \frac{a_i - E[a]}{\sqrt{Var[a] + \epsilon}}\gamma + \beta \qquad \bar{a}_i = \frac{a_i}{RMS[a]}\gamma$$

(a) Layer Norm (b) RMSNorm

Figura 9: Layer Norm x RMSNorm

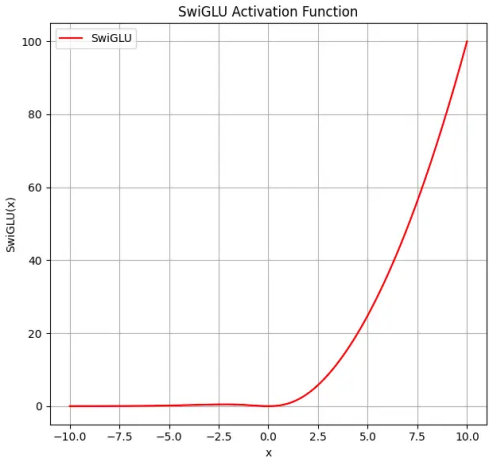


Figura 10: Gráfico de SwigLU.

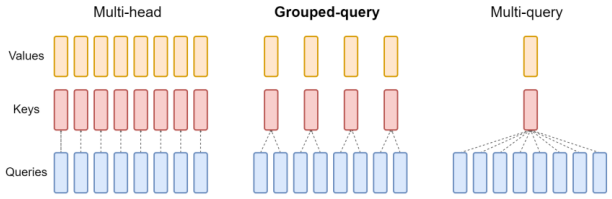


Figura 11: Variações na computação de atenção. (Joshua Ainslie; James Lee-Thorp; Michiel de Jong et al, 2023.)

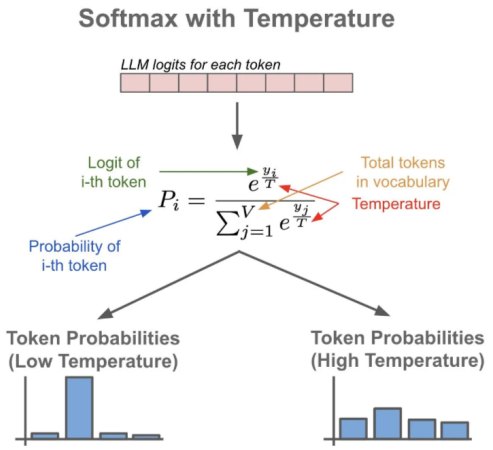


Figura 12: Visualização da temperatura na softmax. (Amansinghalml, 2023.)

Referências

Mayur Wankhade; Annavarapu Chandra Sekhara Rao and Chaitanya Kulkarni, 2022. - *A survey on sentiment analysis methods, applications, and challenges*. https://link.springer.com/article/10.1007/s10462-022-10144-1)

Bhumika Gupta et al., 2017. - *Study of Twitter Sentiment Analysis using Machine Learning Algorithms on Python*. https://www.researchgate.net/publication/317058859_Study_of_Twitter_Sentiment_Analysis_using_Machine_Learning_Algorithms_on_Python)

Biodoumoye George Bokolo and Qingzhong Liu., 2023. - *Deep Learning-Based Depression Detection from Social Media: Comparative Evaluation of ML and Transformer Techniques*. https://doi.org/10.3390/electronics12214396)

Kostadin Mishev et al., 2020. - *Evaluation of Sentiment Analysis in Finance: From Lexicons to Transformers*. https://ieeexplore.ieee.org/abstract/document/9142175/)

Michele Costola et al., 2023. - *Machine learning sentiment analysis, COVID-19 news and stock market reactions*. https://doi.org/10.1016/j.ribaf.2023.101881)

Boyu Zhang et al., 2023. - *Enhancing Financial Sentiment Analysis via Retrieval Augmented Large Language Models*. https://doi.org/10.1145/3604237.3626866)

Kelvin Du et al., 2024. - *An Evaluation of Reasoning Capabilities of Large Language Models in Financial Sentiment Analysis*. http://ww.sentic.net/llm-reasoning-capabilities-in-financial-sentiment-analysis.pdf)

Bahdanau et al., 2014. - *Neural machine translation by jointly learning to align and translate.*(https://arxiv.org/abs/1409.0473)

Ashish Vaswani et al., 2017. - *Attention Is All You Need.* (https://arxiv.org/abs/1706.03762)

Umar Jamil, 2023. - *Attention is all you need (Transformer) - Model explanation (including math), Inference and Training.* (https://github.com/hkproj/transformer-from-scratch-notes)

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun., 2016. - *Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* (https://ieeexplore.ieee.org/document/7780459)

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton.; 2016. - *Layer normalization.*(https://arxiv.org/abs/1607.06450)

Vladimir Ilievski, 2023 - *Batch vs Layer Normalization in Deep Neural Nets. The Illustrated Way!* (https://isquared.digital/blog/2023-03-15-illustrated-batch-vs-layer-norm/)

Hugo Touvron et al., 2023 - *Llama 2: Open Foundation and Fine-Tuned Chat Models.*. (https://arxiv.org/abs/2307.09288)

Umar Jamil, 2023. - *LLaMA explained: KV-Cache, Rotary Positional Embedding, RMS Norm, Grouped Query Attention, SwiGLU.* (https://github.com/hkproj/pytorch-llama-notes/)

Biao Zhang and Rico Sennrich, 2019. - *Root Mean Square Layer Normalization* . (https://arxiv.org/abs/1910.07467)

Jianlin Su et al., 2021. - *RoFormer: Enhanced Transformer with Rotary Position Embedding.* (https://arxiv.org/abs/2104.09864)

Noam Shazeer, 2020. - *GLU Variants Improve Transformer* . (https://arxiv.org/abs/2002.05202)

Joshua Ainslie; James Lee-Thorp; Michiel de Jong et al. 2023. - *GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints.* (https://arxiv.org/abs/2305.13245)

Amansinghalml, 2023. - *Temperature — LLMs.* (https://medium.com/@amansinghalml_33304/temperature-llms-b41d75870510)