DOI: 10.5748/18CONTECSI/PSE/DSC/6866

ANALYSIS OF SENTIMENTS IN PRODUCT REVIEWS TEXTS USING DEEP LEARNING METHODS

Diego Dos Santos Crives - UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE - Orcid: https://orcid.org/0000-0001-6315-2632

José Alfredo Ferreira Costa - UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE - Orcid: https://orcid.org/0000-0002-1290-6454

Diógenes Carlos Albuquerque Araújo - UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE - Orcid: https://orcid.org/0000-0002-9859-9255

João Pedro Da Silva Lima - UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE - Orcid: https://orcid.org/0000-0002-5706-6065

Studying text sentiment analysis techniques using machine learning and artificial intelligence. Use of relevant techniques for a problem still with little automation in the market, in addition to proposing a new way of measuring the result. Training and application of neural networks in a vast text base. A solid way of analyzing texts and giving numerical meaning, reaching a sentiment analysis of a mass of texts. Proposition of a new technique for measuring results with ordinal classification of texts. A way to perform automated text analysis.

Keywords: Text Processing, Natural Language Processing, Machine Learning, Deep Learning, Sentiment Analisys

ANÁLISE DE SENTIMENTO EM TEXTOS DE AVALIAÇÕES DE PRODUTOS USANDO MÉTODOS DE APRENDIZAGEM PROFUNDA

Estudar técnicas de análise de sentimento de textos com o uso de aprendizado de máquina e inteligência artificial Uso de técnicas relevantes para um problema ainda com pouca automatização no mercado, além de propor uma nova forma de metrificar o resultado. Treinamento e aplicação de redes neurais em uma base vasta de textos. Uma forma sólida de analisar textos e dar um significado numérico, chegando a uma análise de sentimento de uma massa de textos. Proposição de uma nova técnica de medição de resultados com classificação ordinal de textos. Uma forma de realizar análise de textos de forma automatizada.

Palavras-chave: Processamento de texto, Processamento de Linguagem Natural, Aprendizado profundo, Aprendizado de Máquina, Análise de sentimento

ANALYSIS OF SENTIMENTS IN PRODUCT REVIEWS TEXTS USING DEEP LEARNING METHODS

ANÁLISE DE SENTIMENTO EM TEXTOS DE AVALIAÇÕES DE PRODUTOS USANDO MÉTODOS DE APRENDIZAGEM PROFUNDA

ABSTRACT

The following work deals with an analysis of natural language processing models using review texts from online stores. An explanation of the basic concepts of Machine Learning, Deep Learning and Natural Language Processing will be carried out. What are the main algorithms, how they are applied in the most diverse Natural Language problems. For the experimental part, a database of customer comments from the B2W group stores was used (Submarino, Americanas, Shoptime, etc.) on various purchased products in order to make a sentiment forecast according to purchased product reviews texts. Based on the users' own grades, which ranged from 1 to 5. We obtained 58% accuracy using Support Vector Machine algorithms and 64% using the Transformer BERTimbau model. The work demonstrates the differences between NLP methods currently used. The Transformer model had the best performance within the techniques used for the prediction chosen in the work. Where it can be improved upon using more data, differente hyperparameters and more processing power

Keywords: Text Processing, Natural Language Processing, Machine Learning, Deep Learning, Sentiment analysis

RESUMO

O trabalho a seguir trata de uma análise de desempenho de modelos de processamento de linguagem natural usando textos de avaliações de produtos de lojas online. Será realizada uma explicação dos conceitos básicos de Aprendizado de Máquina, Aprendizado Profundo e Processamento de Linguagem Natural. Quais são os principais algoritmos, como são aplicados nos mais diversos problemas de Linguagem Natural. Para a parte experimental, um banco de dados de comentários de clientes das lojas do grupo B2W (Submarino, Americanas, Shoptime, etc.) sobre diversos produtos adquiridos, a fim de fazer uma previsão de sentimento de acordo com os textos de avaliações dos produtos adquiridos. Com base nas notas dos próprios usuários, que variaram de 1 a 5. Obtivemos 58% de acerto com os algoritmos Support Vector Machine e 64% com o modelo Transformer BERTimbau. O trabalho demonstra as diferenças entre os métodos de PNL usados atualmente. O modelo Transformer teve o melhor desempenho dentro das técnicas utilizadas para a previsão escolhida na obra. Onde pode ser melhorado usando mais dados, hiperparâmetros diferentes e mais poder de processamento.

Palavras-chave: Processamento de Texto, Processamento de Linguagem Natural, *Transformers*, Aprendizado de Máquina, Aprendizado Profundo, Análise de sentimento

1. INTRODUÇÃO

Com o crescente uso de algoritmos cada vez mais complexos e a necessidade de se automatizar as mais diversas funções na indústria, nunca foi tão necessário a melhoria de processos nos mais diversos campos da Inteligência Artificial para fazermos os computadores gerarem os próprios códigos e automatizarem as funções. Dentro desse escopo, um dos campos que mais tiveram mudanças atualmente foi o de Processamento de Linguagem Natural, que é responsável pela criação de chatbots, analisar sentimento de textos, processar enormes quantidades de texto e extrair informações e etc. Este trabalho se propõe a descrever, usar e comparar técnicas de *Machine Learning* e Redes Neurais para realizar análise de sentimento textual utilizando-se de uma base de dados de comentários de clientes das lojas do grupo B2W acerca de produtos comprados nelas. Nesses textos foram aplicadas técnicas clássicas de *Machine Learning* (Naive Bayes e *Support Vector Machine*), além de técnicas baseadas em modelos da arquitetura *Transformers* (VASWANI et al, 2017) de redes neurais, como a BERTimbau (SOUZA et al, 2020). As técnicas de aprendizado de máquina e *Deep Learning* foram treinadas em mais de 130 mil textos retirados dos sites do grupo B2W para, a partir de algum texto de entrada, dar uma nota de 1 a 5 representando quanto satisfeito o cliente está com o produto comprado. Os resultados do trabalho servem de base sólida para estudos no estado da arte em aplicações de aprendizado de máquina e *Deep Learning* para processamento de linguagem natural. Além de comparação das técnicas mais clássicas (*Naive Bayes* e *Support Vector Machines*) com as mais novas arquiteturas usando redes neurais. O trabalho apresentará uma breve introdução ao *Machine Learning* e suas técnicas clássicas, redes neurais e suas aplicações, além de explicar como funciona a arquitetura *Transformers* para processamento de linguagem natural.

2. MACHINE LEARNING

Aprendizado de Máquina, ou *Machine Learning,* é o estudo de algoritmos de computador que se melhoram pelo uso recorrente e pelo treinamento usando bases de dados (MITCHELL T, 1997:15). É como computadores podem executar tarefas sem serem programados explicitamente para isso, aprendendo a partir de dados fornecidos anteriormente. Existem diferenças conceituais entre *Machine Learning* e Inteligência artificial, IA é um termo muito mais amplo que denota a habilidade de criação de inteligência em máquinas (RUSSEL, 1995:4) criadas pelo homem a partir das mais variadas técnicas. Para alguns algoritmos é fácil um ser humano criar uma lógica e aplicá-la a máquina para realização de uma tarefa, em outras situações é mais eficiente ajudar a máquina a gerar o próprio algoritmo para a realização das tarefas.

2.1. TIPOS DE APRENDIZADO

As tarefas de aprendizado de máquina são tipicamente classificadas em três categorias amplas, de acordo com a natureza do "sinal" ou "feedback" de aprendizado disponível para um sistema de aprendizado (RUSSEL, 1995:528).

No **aprendizado supervisionado** são apresentadas ao computador exemplos de entradas e saídas desejadas com o objetivo de aprender uma regra geral que mapeia as entradas para as saídas. Os dados são conhecidos como dados de treino e consistem numa série de exemplos de treino. Cada exemplo tem uma ou mais entradas e uma saída desejada.

No **aprendizado não supervisionado** nenhum tipo de rótulo é dado ao algoritmo de aprendizado, deixando-o sozinho para encontrar estrutura nas entradas fornecidas. O aprendizado não supervisionado pode ser um objetivo em si mesmo (descobrir novos padrões nos dados) ou um meio para atingir um fim. Em aprendizado não supervisionado, os algoritmos usam uma base de dados que apenas contém entradas e acham estruturas nos dados, agrupando e formando clusters. Desta forma os algoritmos não supervisionados aprendem de dados que não foram previamente divididos em classes ou categorias, sendo o próprio algoritmo que realiza essa divisão como treinamento.

No **aprendizado por reforço** um programa de computador interage com um ambiente vivo, em que o programa deve desempenhar determinado objetivo (por exemplo, dirigir um veículo). É fornecido, ao programa, feedback quanto a premiações e punições, na medida em que é navegado o espaço do problema.

3. MACHINE LEARNING PARA PROCESSAMENTO DE LINGUAGEM NATURAL

Dentro do *Machine Learning*, que se subdivide em várias áreas, temo o Processamento de Linguagem Natural (JOHRI et al, 2020:23), que é uma subárea da inteligência artificial que estuda problemas de geração e compreensão automática de línguas quando faladas ou escritas de um interlocutor humano. O objetivo é criar sistemas computacionais capazes de transformar informações em um banco de dados em textos naturais para um ser humano e também de receber informações e interpretá-las.

3.1 *BAG OF WORDS*

Bag of Words (ZHANG et al, 2010:43) é a representação mais simples de NLP usada atualmente, neste modelo, um texto (uma frase ou um parágrafo) é representado como um vetor contendo a representação quantitativa de cada palavra. Ignorando a gramática e a ordem das palavras. Por exemplo se usarmos as frases “O rato roeu a roupa do rei de Roma” e “Roma é cheia de ratos” é gerado um vetor que cada posição dele corresponde a uma palavra que existe nas sentenças, é formado um vetor que representa as palavras "rato", roeu", "o", "a", "roupa", "do", "rei", "de", "Roma", "é", "cheia, "ratos". Gerando dois vetores representativos: [1,1,1,1,1,1,1,1,1,0,0,0] e [0,0,0,0,0,0,0,1,1,1,2,1].

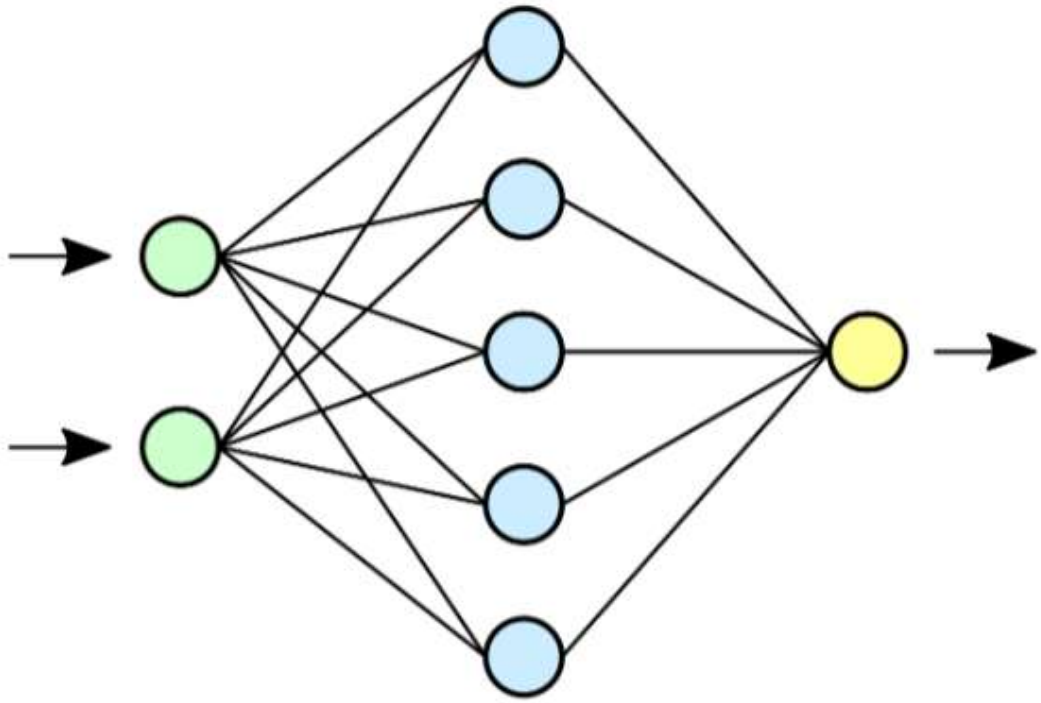
Essa representação vetorial pode ser usada em diversos modelos de aprendizado de máquina, como os já apresentados no capítulo anterior. Uma das principais limitações desse tipo de representação é que ele ignora totalmente a ordem das palavras, que são muito importantes para a compreensão total do sentido do que é falado. Uma forma de contornar esse problema é representar as palavras em bigramas e trigramas, que são conjunto de duas ou três (ou até mais) palavras. Mas a quantidade de conjuntos de palavras facilmente chega a valores altíssimos em textos grandes. Isso gera impossibilidade de se trabalhar com dados tão grandes por limitações de hardware.

3.2 REDES NEURAIS ARTIFICIAIS

São modelos computacionais inspirados pelo cérebro humano (Hopfield 1982). Normalmente são representadas por neurônios interconectados que podem comportar vários valores de entrada. A inspiração original para essa técnica advém do exame das

estruturas do cérebro, em particular do exame de neurônios. A propriedade mais importante das redes neurais é a habilidade de aprender de seu ambiente e com isso melhorar seu desempenho. Isso é feito através de um processo iterativo de ajustes aplicado aos seus pesos, o treino. Conforme a figura 1

Figura 1: Esquema de uma rede neural



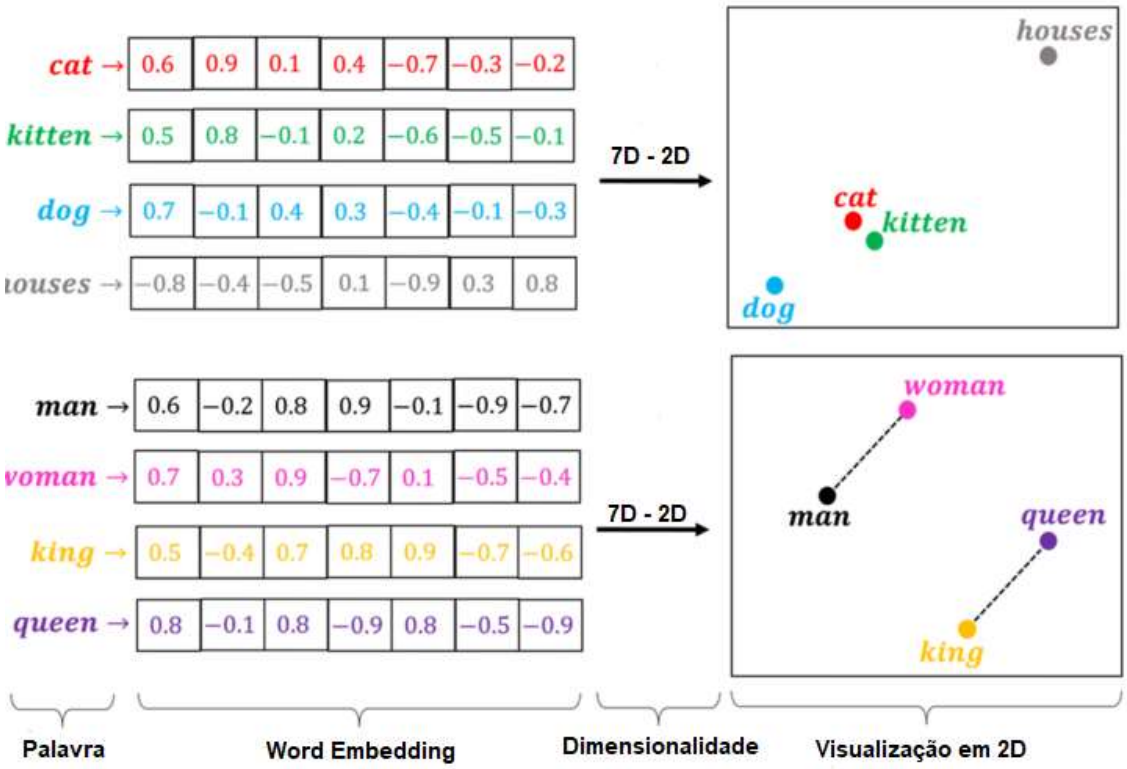
Fonte: Wikipédia - Redes Neurais Artificiais (2021).

Uma rede neural biológica é composta por grupos de neurônios quimicamente conectados. Um neurônio pode se conectar a vários outros neurônios e o número total de neurônios e conexões pode ser enorme. Paralelamente, as redes neurais artificiais seguem esse modelo podendo ter muitas camadas de neurônios para modelar o sistema.

3.2.1 *WORD EMBEDDING*

Word Embeddings (Amid Mandelbaum, 2016:1-2) são uma classe de algoritmos amplamente utilizadas atualmente em redes neurais que converte palavras em vetores numéricos n-dimensionais em que a proximidade dos vetores no espaço vetorial reflete a proximidade de significado entre as palavras dentro de um contexto. Entre os algoritmos mais conhecidos de *Word Embedding* estão o *Word2Vec, GloVe*, *WordRank*, *FastText* e etc. Todos esses algoritmos geram diferentes representações vetoriais das palavras, as quais podem ser utilizadas para alimentar um modelo de aprendizado de máquina. *Word Embeddings* são a base para tarefas de análise de sentimento, sumarização de texto, tradução e qualquer outra tarefa de NLP atual.

Figura 2: Esquema do *Word Embedding*

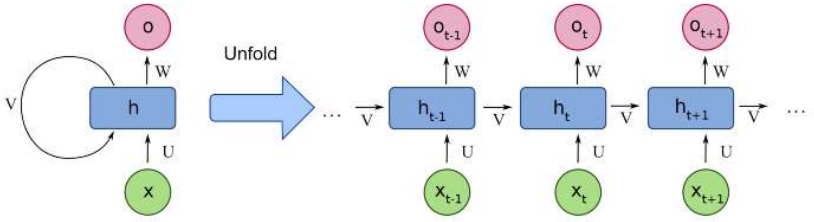


Fonte: BORAH (2021)

3.2.2 REDES NEURAIS RECORRENTES

A próxima evolução em modelos para NLP foi o uso de Redes Neurais Recorrentes (RNNs) para uso com textos naturais (RUMELHART et al, 1985:28). Que podem trabalhar melhor entradas de tamanhos variáveis (os textos) para chegar a uma saída correta. As RNNs são uma classe de Redes Neurais que permitem que as saídas anteriores possam ser usadas como entradas. A figura 3.1 mostra um modelo de um bloco de RNN.

Figura 3: Esquema de um Rede Neural Recorrente



Fonte: BORGES (2018)

Onde x, que é a entrada (uma palavra ou sentença) passa pelo bloco da rede neural recorrentemente. A representação dobrada e desdobrada são equivalentes, apenas que a desdobrada mostra o que acontece em cada tick da rede. Importante lembrar que mesmo que se mostre o bloco h várias vezes, na verdade é o mesmo bloco sendo usado, o bloco envia a saída para ele mesmo continuamente até ser mandado parar.

A representação mais simples do bloco h pode ser dada como o bloco simplesmente combinar a entrada com a saída através de uma função tanh para ter a saída do tick atual. A limitação com esses tipos de abordagem é que pode ficar muito pesado computacionalmente armazenar entradas de sequências longas para se manter a conexão entre as palavras. Com isso as RNN "esquecem" das palavras para manter a performance aceitável. Isso limita a capacidade de aprendizado da rede em comparação a outros métodos

3.2.3 *TRANSFORMERS*

Com o trabalho Ashish *et al*. no *paper* "*Attention is all you need"* descreveram uma nova arquitetura de redes neurais que não se baseava em redes neurais de múltiplas camadas, mas sim em camadas de atenção.

A camada de atenção (BAHDANAU et al, 2016:4) é uma camada da rede neural que dá ao texto a capacidade de, ao gerar palavras uma a uma, saber para qual palavra do texto a próxima palavra "atende", de acordo com o que aprendeu dentro do corpus em que ela foi treinada. O que, dentro da arquitetura *Transformers* (lembrando que *attention* é uma camada dentro da arquitetura) pode virtualmente eliminar o problema de memórias das redes anteriores. Funciona com cada token (palavra) da entrada sendo comparada com cada palavra da saída e da própria entrada, atribuindo pesos para cada possível saída e retornando os tokens com os maiores pesos. A arquitetura *Transformers* usa a camada de atenção como mecanismo principal. Comum a estrutura de codificação e decodificação (entrada e saída).

3.2.3.1 CAMADA DE CODIFICAÇÃO

O primeiro passo é alimentar a entrada com um vetor que representa cada palavra num espaço de n dimensões. Como não há recorrência como nas RNNs, é necessária uma codificação posicional para a arquitetura entender a ordem das palavras. A solução é usar transformadas de seno e cosseno, dividindo entre entradas pares e ímpares:

$$PE_{pos,2i} = sen(pos/10000^{2i/d_{model}})$$

$$PE_{pos,2i+1} = sen(pos/10000^{2i/d_{model}})$$

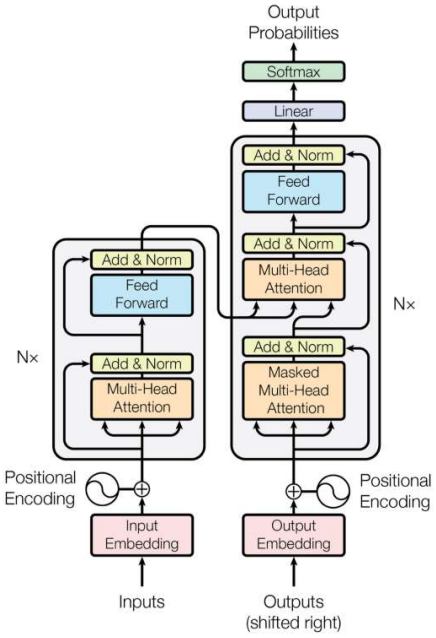
Após isso temos a camada de codificação, que inicia usando uma camada de atenção especial chamada *Multi-Headed Attention*. O *Multi-Headed Attention* aplica um algoritmo de atenção chamado *self-attention* que associa cada palavra da entrada com ela mesma e outras palavras. Assim ele pode aprender a associar a palavra "você" com "como" e "está", além de aprender que palavras nessa ordem em geral são perguntas e

pode responder apropriadamente. O bloco de *Multi-Headed Attention* transforma os vetores de entrada em 3 vetores, Q, K e V (*Query*, *Key* e *Value*). Os vetores Q e K são multiplicados de forma escalar para formar a matriz de *scores* que determina o quanto do foco de uma palavra deve ser destinado a outras. Em seguida a matriz é dividida pela raiz quadrada da dimensão de K para normalização e prevenir explosão de valores em multiplicações maiores. Em seguida, a função Softmax é aplicada a matriz de *scores* escalonada para aumentar as distâncias entre os valores altos e valores baixos. A função usada é a seguinte:

$$softmax(x)_i = \frac{exp(x_i)}{\sum_j exp(x_j)}$$

O resultado é multiplicado pelo vetor V, quanto maior o valor da saída da softmax da palavra, mais importante ela é nesse contexto. As de score baixo serão descartadas. Sendo assim, em resumo, a camada de *Multi-Headed Attention* é um módulo da arquitetura *Transformers* que calcula os pesos de atenção da entrada e gera uma saída codificada que diz como cada token (vetor n dimensional) deve ser relacionar com todas os outros em sequência. A inovação dessa camada é que ela pode ser interpretada dependendo do que o foco dela está voltado. Podemos rodar a camada para aprender gramática, aprender conjugação, aprender vocabulário e etc. O resultado da camada de atenção é adicionado ao vetor de entrada original (que é chamado de conexão residual) e passa por uma camada de normalização, que realiza transformações lineares e uma função de ativação ReLU e é novamente adicionado a entrada dessa camada de normalização. A conexão residual ajuda a arquitetura a treinar, permitindo que os gradientes fluam pela rede diretamente, ajudando a projetar a camada de atenção, gerando uma representação mais rica.

Figura 4: Representação da Arquitetura Transformers



Fonte: VASWANI (2017)

3.2.3.2 CAMADA DE DECODIFICAÇÃO

O trabalho da camada de decodificação é gerar sequências de textos. O *decoder* tem blocos similares a camada de codificação: têm dois blocos de *Multi-Headed Attention*, 2 camadas de *feedforward* com ReLU, conexões residuais e uma camada de normalização após cada subcamada, finalizando com uma camada de *Softmax* que serve de classificador. O *decoder* é auto-regressivo, ou seja, começa com um *token* e pega a lista de saídas anteriores como entradas assim como as saídas do *encoder* que contém as informações da camada de atenção. Ele inicia da mesma forma que o *encoder*, com os *tokens* de entrada passando pelo decodificador posicional e após isso entrando na primeira camada de atenção do decodificador. Como o decodificador funciona palavra a palavra, essa camada de atenção tem uma diferença para impedir que as palavras tenham acessos a palavras futuras. A necessidade da máscara é, já que os termos negativos infinitos virarão zeros ao passar pela função *Softmax*, forçando o modelo a voltar sua "atenção" às palavras anteriores. Esse mascaramento é a única diferença na primeira camada de atenção do decodificador. Daí segue para a segunda camada de atenção que as entradas Q e K são as saídas da camada de codificação e o V é a saída da primeira camada de atenção do decodificador. Por fim, a saída da segunda camada de atenção segue para uma camada de normalização, uma camada linear que serve como classificador e depois passa pela função *Softmax* que vai dar os pesos para cada palavra do seu corpus e usará a com maior peso como a palavra que será prevista. Palavra esta que entrará novamente na camada de decodificação como uma entrada.

3.2.3.3 MODELOS BASEADOS EM TRANSFORMERS

A arquitetura *Transformers* deu início a uma revolução na tecnologia de NLP, com ela foram criados vários modelos, todos baseados em *Transformers* e seu uso da camada de atenção.

Modelos Autoregressivos são pré-treinados na tarefa clássica de modelagem de linguagem: vão prever o próximo *token*, tendo lido todos os anteriores. Corresponde a camada de codificação do modelo original do *Transformers*. Podendo ser treinados para uma vasta gama de tarefas. Exemplos: GPT, GPT2, GPT3, *Transformer*-XL e etc.

Modelos de auto-codificação são pré-treinados corrompendo os *tokens* de entrada e tentando reconstruir a sentença original. Correspondem a camada de decodificação do modelo original do *Transformers* só que sem a máscara de atenção na camada de decodificação. Exemplos: BERT, distilBERT, BERTimbau, roBERTa e etc.

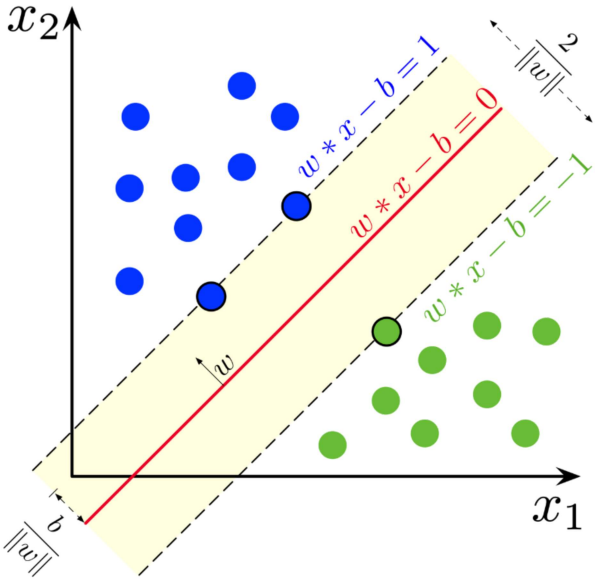
Modelos de sequência-a-sequência usam tanto a camada de codificação quanto a camada de decodificação. Também podem ser usados em diversas tarefas. Exemplos: BART, Pegasus, MT5 e etc.

3.3 MÁQUINA DE VETORES DE SUPORTE

Formalmente, uma máquina de vetores de suporte (Cristianini Nello, 2008), ou SVM (*Support Vector Machines*), constrói um hiperplano ou uma série de hiperplanos que podem ser usados para classificação, regressão e outras tarefas.

Intuitivamente uma boa separação é alcançada quando o hiperplano tem a maior distância do dado de treinamento mais próximo. Em outras palavras, o modelo SVM cria planos de separação dos dados de entrada em um espaço vetorial de *n* dimensões.

Figura 5: Esquema de um SVM



Fonte: Wikipédia - Máquinas de Vetores de Suporte (2021).

3.4 NAIVE BAYES

É um classificador probabilístico supervisionado fundamentado no Teorema de Bayes com algumas simplificações adicionais (RISH, 2001:41-42). Ele assume que a presença ou ausência de uma característica particular não está relacionada com nenhuma outra característica dos dados de entrada. Fazendo que cada uma das características dos dados de entrada influencie independentemente do resultado final. É dado com a probabilidade de um "A" dado um evento "B" vezes a probabilidade de "A" independentemente de "B" e da probabilidade de "B" independentemente de "A". A definição matemática do classificador é dada por:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

4. RESULTADOS

Para demonstrar o funcionamento na prática de algumas das técnicas mencionadas aqui, foi escolhido um base de dados (disponível no github do projeto) de comentários de clientes acerca de produtos comprados nas lojas da B2W (Americanas, Submarino, Shoptime e etc.). O *dataset* contém 132 mil comentários com notas dadas pelos próprios usuários que são 1 (muito ruim), 2 (ruim), 3 (razoável),4 (bom) até 5 (muito bom). Os modelos foram treinados com os comentários (textos curtos) para prever a nota desses textos. O *dataset* completo tem uma gama ampla de informações dos usuários que fizeram os comentários, como: idade, sexo, localização geográfica e etc. Para os objetivos do trabalho apenas iremos utilizar a nota (saída do sistema) e o texto de avaliação do usuário (entrada do sistema). Segue a seguir exemplos de dados retirados diretamente do dataset:

5 - Máscara de cílios incrível, tenho pouquíssimo cílios e ela deu um super efeito! Amei!

5 - Aparelho de ótima qualidade e sem defeitos, entrega com pontualidade e qualidade !

5 - O produto chegou no prazo e em ótimo estado, tenho sido muito bem atendida quando solicito algo no site.

3 - Tem bom acabamento e é pratico. Serviu para o que eu precisava!

4 - Ótimo equipamento. Som forte, portátil, prático. Uso muito.

5 - Perfeito, funciona muito bem. Ótimo acabamento. Recomendo

4.1 MODELOS UTILIZADOS

Para realizar os treinamentos, foram utilizadas duas técnicas tradicionais (SVM e Naive Bayes) para base de comparação e o modelo *Transformer* BERTimbau (Souza et al. 2019), que é um tipo de BERT (Turc et al. 2019) treinado especificamente para tarefas em português. Todos eles usando a linguagem de programação *Python* dentro da IDE VScode.

Os datasets foram processados usando NLTK, Pandas e *Numpy* e treinados com o uso da biblioteca *Ktrain* (Maiya 2020).

4.2 PRÉ-PROCESSAMENTO

Para uso na maioria das técnicas e modelos de Machine Learning, é indicado um pré-processamento nos textos para diminuição da carga de trabalho com a retirada das palavras que não carregam significado e facilitar a tokenização das palavras do texto (Faceli 2011). Nessa etapa foi aplicado normalização de palavras, stematização, remoção de caracteres especiais e remoção de *stopwords,* retirada de palavras sem sentido (ex: ‘ctbhhhhvvcxxxxcxcxxyggewssx’). Como o *dataset* tem um grande volume de textos foi dividido em 90% para treino (118 mil textos) e 10% para teste (12 mil textos).

4.3 APLICAÇÃO DE NAIVE BAYES E SVM

Para efeito de comparação com técnicas mais tradicionais em relação ao modelo transformers foi realizado o treinamento e aplicação dos modelos usando as técnicas de Naive Bayes padrão e *Support Vector Machines* com kernel linear. A técnica de *word embedding* para vetorização das palavras foi a *FastText* (BOJANOWSKI, P. et al, 2017:2).

Os resultados da aplicação das técnicas SVM e Naive Bayes são demonstrados nas figuras a seguir. O SVM teve uma acurácia de 58% e Naive Bayes apenas 10%. Isso significa que o SVM conseguiu acertar perfeitamente a nota, de 1 a 5, em 58% dos casos, enquanto Naive Bayes apenas 10%. Os resultados do SVM foram razoáveis, mas ainda não satisfatórios para uso em ambiente real, enquanto que o Naive Bayes teve um resultado totalmente inadequado para uso real.

Tabela 1: Resultados SVM

Revocacao	Precisao	F-medida	Classe
0.859	0.728	0.788	1
0.185	0.309	0.232	2
0.255	0.399	0.311	3
0.322	0.435	0.370	4
0.782	0.617	0.690	5
---	---	---	---
0.481	0.498	0.478	Média macro
0.582	0.582	0.582	Média micro

Acuracia: 0.582

Tabela 2: Resultados Naive Bayes

Revocacao	Precisao	F-medida	Classe
0.091	0.537	0.155	1
0.827	0.063	0.116	2
0.074	0.143	0.098	3
0.038	0.272	0.066	4
0.038	0.624	0.072	5
---	---	---	---
0.214	0.328	0.102	Média macro
0.104	0.104	0.104	Média micro

Acuracia: 0.104

Fonte: Criação Própria

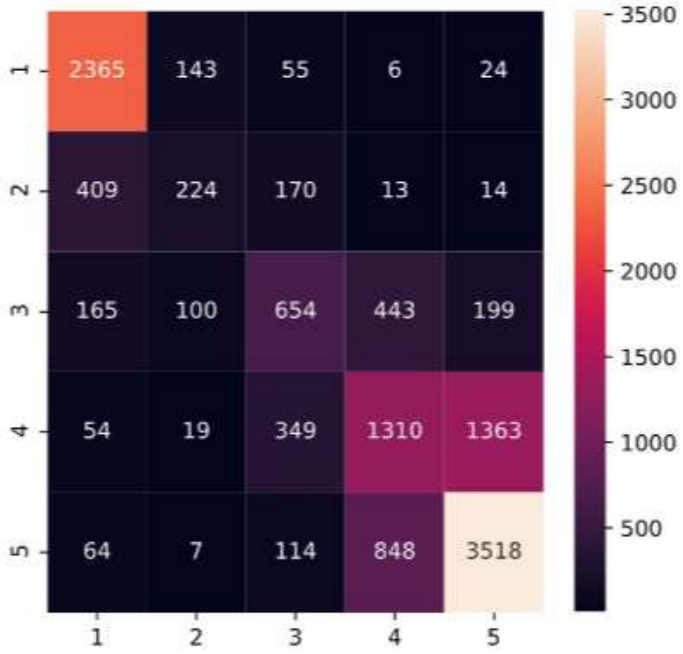
4.4 APLICAÇÃO DO TRANSFORMERS

Para aplicação da arquitetura Transformers usando o modelo BERTimbau e o *wrapper Ktrain*, não é necessário um pré-processamento tão robusto. Apenas retiramos textos que contêm apenas palavras sem sentido. Para treinamento foi usado *batch size* de tamanho 6, uma *learning rate* de 0,00003 escolhida através de algoritmo provido pelo *Ktrain*. O modelo nos deu uma acurácia de 64% (10% de melhora em comparação ao SVM). Também foi gerado o mapa de calor correspondente a matriz de confusão desse resultado na figura 8.

Tabela 3: Resultados Transformers

	precision	recall	f1-score	support
1	0.77	0.92	0.84	2716
2	0.42	0.23	0.30	837
3	0.51	0.40	0.45	1615
4	0.49	0.37	0.42	3232
5	0.67	0.81	0.73	4838
accuracy			0.64	13238

Figura 6: Mapa de calor gerado pelo modelo Transformers



Fonte: Criação Própria

Essa visualização nos ajuda a visualizar por outro ângulo: Mesmo que, aparentemente, nossa precisão não seja tão alta comparada a outros modelos. Podemos ver que a grande maioria dos resultados está ao redor da diagonal principal da matriz. O que nos mostra que, para o objetivo de capturar a tendência geral dos sentimentos dos usuários, o modelo está fazendo um ótimo trabalho. Usando o O*rdinal Classification Index* (CARDOSO & SOUSA, 2011:5), que gera um índice comparativo de performance entre matrizes de confusão que usam classes com ordinalidade foi obtido, sendo 0 o mínimo e 1 o máximo, um resultado de 0.46.

4.4.1 MÉTODO PROPOSTO

Para um número que reflita mais especificamente a realidade do caso estudado e o quão importante cada resultado e sentimento é, pois para cada caso de uso um "bom" pode valer quase tanto um "muito bom" ou não. Criam-se matrizes de pesos, realizar um produto Hadamard (MILLION, 2007:1), somar todos os valores da matriz resultante e dividir pela quantidade total de textos de teste (12630). Capturando numericamente em um único valor o quanto o resultado do modelo reflete a tendência de sentimento (favorável ou não favorável) real. Vamos fazer os cálculos com algumas diferentes matrizes de pesos:

Tabela 4: Matrizes propostas

MATRIZ DE CONFUSÃO				
365	143	55	6	24
409	224	170	13	14
165	100	654	443	199
54	19	349	1310	1363
64	7	114	848	3518

MATRIZ A				
1	0,8	0,5	0	0
0,8	1	0,5	0	0
0	100	1	0,5	0
0	0	0,5	1	0,5
0	0	0	0,8	1

MATRIZ B				
1	0,5	0	0	0
0,5	1	0,5	0	0
0	0,5	1	0,5	0
0	0	0,5	1	0,5
0	0	0	0,5	1

MATRIZ C				
1	0,75	0,5	0	0
0,75	1	0,75	0	0
0	0,5	1	0,5	0
0	0	0,75	1	0,75
0	0	0,5	0,75	1

Com a multiplicação dessas matrizes pela matriz de confusão inicial pelo produto Hadamard obtivemos os seguintes valores de acerto de tendência:

Tabela 5: Valores de confiança do produto da MC com cada matriz

MATRIZ A	82%
MATRIZ B	79%
MATRIZ C	86%

5. CONCLUSÕES

O trabalho executado mostrou que para determinados tipos de tarefa, as técnicas e os modelos devem ser pesados muito bem na escolha de uma aplicação de IA, pois nem todos se adequam ao tipo de texto que será trabalhado (Naive Bayes sendo um exemplo claro). O BERTimbau foi uma escolha clara aqui por já ter sido treinado previamente em português, diferente de outros modelos mantidos no Huggingface (Site indexador de modelos *Transformers*). Outro ponto a se destacar é a quantidade de material de livre acesso em português é ainda pequeno em comparação com materiais em inglês.

A imensa maioria dos modelos Transformers pré-treinados não tem suporte nativo a português (mesmo que alguns funcionam bem mesmo assim) e a quantidade de *datasets* ou *corpus* de uso livre de diversos tipos não é tão grande quanto em outras línguas. A arquitetura Transformers melhorou muito o problema de memória das Neural Networks, em especial a LSTM (Sepp Hochreiter, 1997:1-3) que ainda é amplamente utilizada, criando a possibilidade de se trabalhar com textos de tamanhos muito maiores, mas com isso cresce a necessidade de alto poder de processamento e tamanho de memória. No uso do BERTimbau, que é um BERT com um treinamento específico em português gerou resultados comparativamente melhores que as técnicas mais tradicionais: 58% para o SVM, 10% para Naive Bayes e 64% para o BERTimbau.

Uma grande dificuldade desse tipo de *dataset* usado, é de que os textos e suas notas são respostas totalmente objetivas. Por exemplo, um "Muito bom!" pode ser tanto um muito bom ou um bom. Além disso, em muitos casos os comentários feitos são totalmente contrários às notas que são dadas. Ainda assim, foi demonstrado que a técnica consegue capturar o sentimento (negativo contra positivo) dos textos de modo bem satisfatório. Sendo possível o uso para detecção de sentimento geral de um grupo de comentários.

5.1 MELHORIAS E EXTENSÕES

Para melhoria dos resultados obtidos, alguns pontos podem ser melhorados: a utilização de um *dataset* maior, nos testes foi visto que com os modelos utilizados ainda não alcançado um platô de melhoria com o aumento da quantidade de dados de treino.

Além disso, é possível treinar uma maior quantidade de modelos Transformers disponíveis para descobrir qual se adequa mais ao trabalho de análise de sentimento com ordinalidade. Os parâmetros e hiper-parâmetros utilizados também podem receber um tratamento mais específico, com o uso de outras bibliotecas com diferentes níveis de controle.

REFERÊNCIAS

BAHDANAU, D.; CHO, K.; BENGIO, Y. **Neural Machine Translation by Jointly Learning to Align and Translate**, 2016.

BOJANOWSKI, P. et al. **Enriching Word Vectors with Subword Information**, 2017.

CARDOSO, J.; SOUSA, R. Measuring the Performance of Ordinal Classification. **IJPRAI**, v. 25, p. 1173–1195, 2011.

CRISTIANINI, N.; RICCI, E. Support Vector Machines. In: KAO, M.-Y. (Ed.). . **Encyclopedia of Algorithms**. Boston, MA: Springer US, 2008. p. 928–932.

DAVID E. RUINEIHART, G. E. H. .; WILLIAMS, AND R. J. Learning Internal Representations through Error Propagation. **ICS Report 8506**, p. 28–29, 1985.

HOCHREITER, S.; SCHMIDHUBER, J. Long Short-term Memory. **Neural computation**, v. 9, p. 1735–1780, 1997.

HOPFIELD, J. J. Neural networks and physical systems with emergent collective computational abilities. **Proceedings of the National Academy of Sciences**, v. 79, n. 8, p. 2554–2558, 1982.t

JOHRI, P. et al. **Natural Language Processing: History, Evolution, Application and Future Work**. [s.l: s.n.].

MAIYA, A. S. **ktrain: A Low-Code Library for Augmented Machine Learning**, 2020.

MANDELBAUM, A.; SHALEV, A. **Word Embeddings and Their Use In Sentence Classification Tasks**, 2016. Disponível em: <http://arxiv.org/abs/1610.08229>

MITCHELL, T. **Machine Learning**, p. 15-16, 1997

RISH, I. An Empirical Study of the Naïve Bayes Classifier. **IJCAI 2001 Work Empir Methods Artif Intell**, v. 3, p. 41–42, 2001.

RUSSEL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. [s.l: s.n.].

SOUZA, F.; NOGUEIRA, R.; LOTUFO, R. **BERTimbau: Pretrained BERT Models for Brazilian Portuguese**. (R. Cerri, R. C. Prati, Eds.)Intelligent Systems. **Anais**...Cham: Springer International Publishing, 2020

VASWANI, A. et al. Attention is all you need. **Advances in Neural Information Processing Systems**, v. 2017-Decem, n. Nips, p. 5999–6009, 2017.

ZHANG, Y.; JIN, R.; ZHOU, Z.-H. Understanding bag-of-words model: A statistical framework. **International Journal of Machine Learning and Cybernetics**, v. 1, p. 43–52, 2010.