Extração de Aspectos Utilizando Arquitetura Transformers

José Carlos Ferreira Neto[1]

[1]Programa de Pós-Graduação em Engenharia de Sistemas e Automação – Universidade Federal de Lavras (UFLA)

Abstract. *Sentiment analysis is an area of research that generates interest both in academia and industry. Aspect-based opinion mining is a major challenge in this field of research as it relates to opinion sources from social media and online advertising. In this work, the Transformers architecture is used to perform the aspect extraction task. The models are evaluated using two datasets, TV and ReLi, and compared with results obtained by the CRF model.*

Resumo. *A análise de sentimentos é uma área de pesquisa que gera interesse tanto no meio acadêmico quanto na indústria. A mineração de opinião baseada em aspectos é um grande desafio neste campo de pesquisa no que se refere a opiniões de origem das mídias sociais e publicidade online. Neste trabalho é empregado o uso da arquitetura Transformers para executar a tarefa de extração de aspectos. Os modelos são avaliados utilizados dois datasets, o TV e o ReLi e comparados com resultados obtidos pelo modelo CRF.*

1. Introdução

A análise de sentimentos é uma área de pesquisa que gera interesse tanto no meio acadêmico quanto na indústria, por conta dos potenciais benefícios que podem ser aproveitados em muitas aplicações da vida real. A mineração de opinião baseada em aspectos é um grande desafio neste campo de pesquisa no que se refere a opiniões de origem das mídias sociais e publicidade online [Poria et al. 2014]. Aspectos são atributos ou componentes de uma entidade, como por exemplo, um produto ou um serviço.

Muitas técnicas podem ser empregadas para a extração de aspectos, a maioria deles podem ser classificadas como: modelos baseados em regras, modelos baseado em tópicos e modelos sequenciais. O primeiro utiliza correspondência de padrões de escrita e a relação entre as palavras, a segunda leva em consideração que um documento é um conjunto de documentos distintos, e a última leva em consideração a ordem das palavras [Cardoso and Pereira 2020].

Neste trabalho faremos uso dos modelos sequenciais, mais especificamente da arquitetura Transformer. Essa arquitetura possui desempenho de estado da arte em muitas tarefas de processamento de linguagem natural, e por conta disso, foi a escolhida.

Os resultados obtidos neste trabalho serão comparados com os obtidos em [Cardoso and Pereira 2020], que utiliza o modelo sequencial baseado em *Conditional Random Fields* (CRF).

A seção 2 apresenta toda a fundamentação teórica das arquiteturas que são utilizadas neste trabalho. A seção 3 apresenta as características dos conjuntos de dados utilizados para avaliar o desempenho dos modelos. A seção 4 apresenta estratégias utilizadas. A seção 5 apresenta as características do *hardware* e das ferramentas utilizadas. A

seção 6 apresenta os resultados obtidos e por fim, a seção 7 aponta as conclusões sobre os resultados obtidos.

2. Fundamentação Teórica

Os tópicos a seguir abordarão conceitualmente o Transformers e o BERT.

2.1. Transformers

A arquitetura Transformers (Figura 1) foi proposta em [Vaswani et al. 2017]. O transformers é um modelo *sequence-to-sequence*, sendo que, sua arquitetura baseia-se em mecanismos de atenção e é composta por duas partes: *encoder* (bloco esquerdo da Figura 1) e o *decoder* (bloco direito da Figura 1).

O *Transformers* é uma arquitetura que é sensível tanto a ordem das palavras quanto ao contexto. Nos itens a seguir, serão apresentados cada elemento que compõe esta arquitetura.

2.1.1. *Self-Attention*

Em níveis neurocientíficos, [Lindsay 2020] descreve atenção como sendo, em sua forma mais genérica, um nível geral de alerta ou capacidade de se envolver com o ambiente. Em Aprendizado de Máquina, a ideia por trás do mecanismo de atenção é de que, nem toda informação de entrada é igualmente importante para o problema em questão. Alguns modelos vão dar mais atenção a alguns atributos e menos atenção a outros [Chollet 2021].

O propósito do *self-attention* é gerar representações para um token de acordo com as representações dos tokens relacionados presentes na sequência, produzindo novas representações que são sensíveis ao contexto [Chollet 2021].

Para que estas representações sejam geradas, as pontuações de atenção (*attention scores*) são calculadas a partir de cada vetor de palavra em relação aos outros vetores presentes em uma sequência. A pontuação de atenção é um valor que mede a relevância entre palavras. Em seguida, estes valores são usados para ponderar a soma dos vetores desta sequência, gerando uma nova representação para estas palavras [Chollet 2021]. Esse processo pode ser resumido no esquema a seguir:

$$outputs = sum(values * pairwise\_scores(query,\ keys)) \tag{1}$$

Onde, para cada token presente na *query*, é calculado o quanto o token é relacionado com cada token em *keys*. O resultado dessa operação é usado para ponderar a soma dos tokens de *values* [Chollet 2021].

O arquitetura do Transformers aplica o *self-attention* através de um conceito chamado *multi-head attention*, em que, os vetores de entrada são subdivididos e cada uma destas divisões são destinada a uma camada que aplica o *self-attention*. Ao final, os resultados de cada camada são concatenados [Chollet 2021].

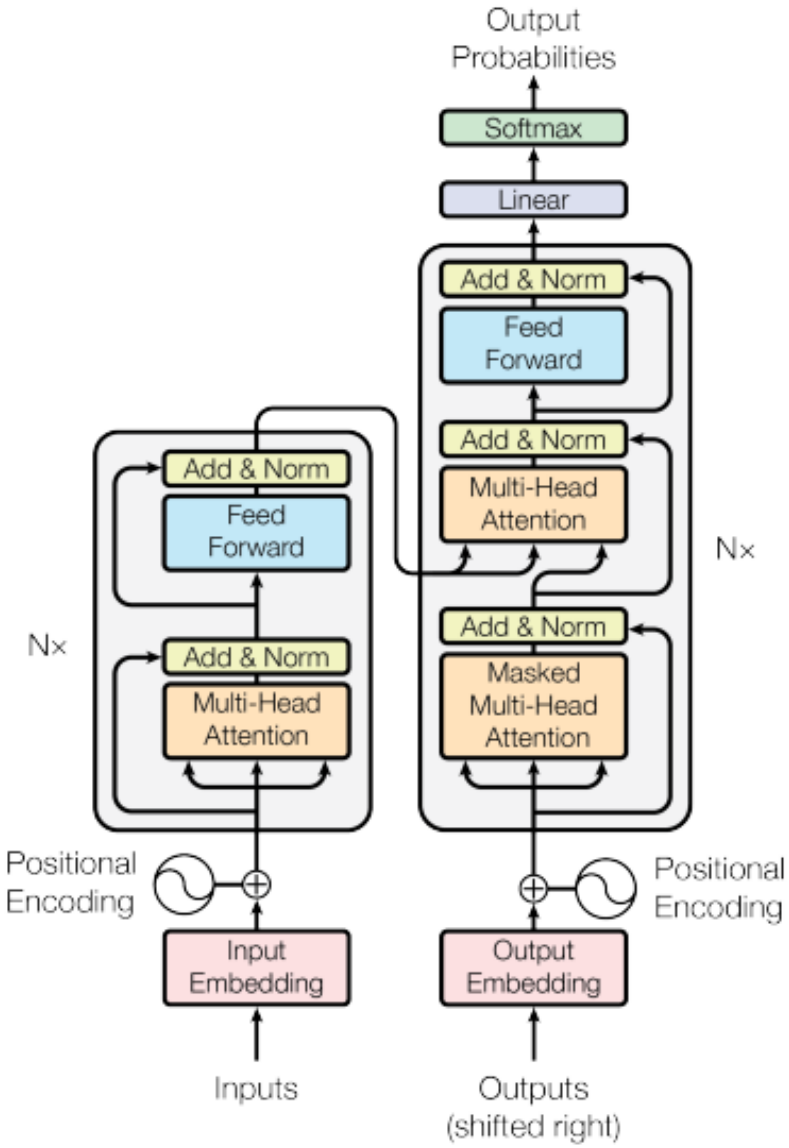

Figura 1. Arquitetura Transformers [Vaswani et al. 2017]

2.1.2. *Encoder*

O *encoder*, isto é, o codificador, tem como objetivo processar os dados de entrada e gerar novas representações para estas entradas. Ele conta com uma camada de *multi-head attention*, camadas densas e camadas de normalização. A primeira é utilizada para enriquecer cada *token* (*word embedding*) com informações contextuais de toda a sentença, a segunda para gerar representações mais úteis de acordo com a tarefa que se deseja solucionar e a última é responsável por auxiliar o gradiente a fluir melhor durante a etapa de atualização dos pesos (*backpropagation*), restringindo a movimentação da média e desvio padrão dos vetores [Chollet 2021].

Este módulo recebe como entrada as representações vetoriais das palavras (*word embeddings*) sem nenhuma informação de contexto, e um vetor contendo a posição de cada elemento na sequência (*positional embedding*) [Chollet 2021].

É possível empilhar estes módulos uns sobre os outros, sendo que, a saída de cada *encoder* é utilizado para alimentar a entrada do próximo, assim por diante. A saída do último bloco é utilizado para alimentar o *decoder* [Chollet 2021].

2.1.3. *Decoder*

O *decoder*, isto é, o decodificador, é semelhante ao *encoder*, exceto que este calcula a atenção da sequência alvo. Este módulo lê os *tokens* de 0 a N, onde N é o comprimento da sequência, e tenta prever o *token* $N + 1$. Ao fazer isto, é possível identificar qual ou quais *tokens* da entrada já codificados estão relacionados ao *token* em que o *decoder* está tentando prever [Chollet 2021].

2.2. BERT

BERT é acrônimo para *Bidirectional Encoder Representations from Transformer*, que em tradução livre, seria Representações do Codificador Bidirecional do Transformer.

No BERT, faz-se uso da arquitetura Transformer, removendo apenas o *decoder*. Por conta disso, o BERT não tenta prever a próxima palavra em uma sequência, ao invés disto, ele faz uso de uma técnica chamada *Masked LM* (MLM). Nesta técnica, algumas palavras de uma sentença são escolhidas de forma aleatória e mascaradas para que o BERT possa predizê-las. Além desta estratégia, o pré-treinamento contou com a tarefa de previsão da próxima frase (*Next Sentence Prediction*). Neste caso, o modelo deveria prever se uma frase é a continuação de uma outra frase ou não [Devlin et al. 2018].

O BERT é treinado de forma bidirecional, isso significa que, uma sentença é processada tanto da esquerda para a direta, quanto da direta para a esquerda [Devlin et al. 2018]. Este processo é feito ao mesmo tempo, por conta disso pode-se dizer que o BERT é agnóstico quanto a direção de "leitura"da sentença.

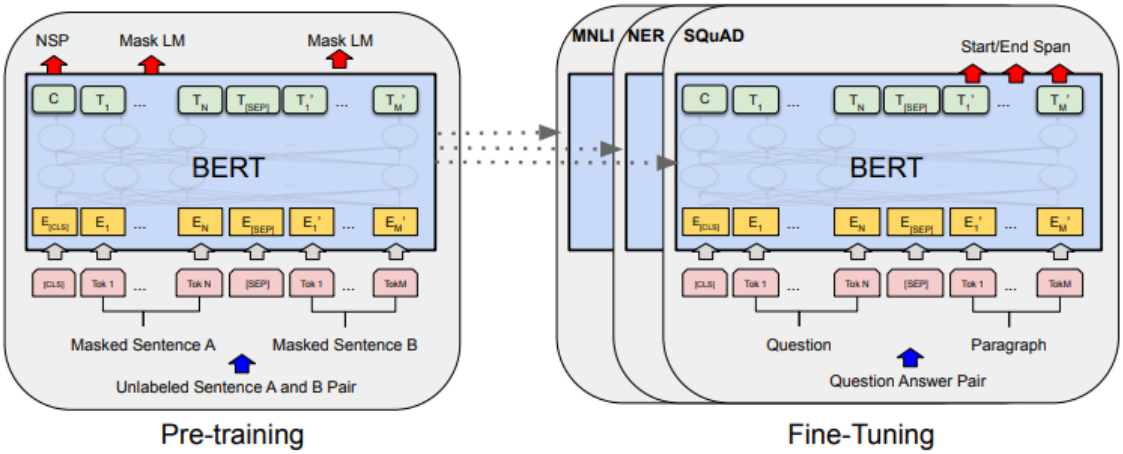

Figura 2. BERT [Devlin et al. 2018]

A Figura 2 exibe a arquitetura usada para o pré-treinamento (bloco esquerdo) e a usada para ajuste fino (bloco direito), nota-se que trata-se da mesma arquitetura para ambos.

3. Conjunto de Dados

Para avaliar o desempenhos dos modelos em tarefas de extração de aspectos, a base de dados utilizada foi produzida por [Cardoso and Pereira 2020]. Esta base conta com revisões sobre um modelo de televisão coletadas a partir de sites de comércio eletrônico. Essa base conta com 1.091 revisões do produto, 2.329 sentenças, 2.388 aspectos marcados, sendo 350 distintos. Ao todo são 61.7% de sentenças com aspectos marcados. A Figura 3 exibe exemplos de frases e aspectos presentes neste conjunto de dados.

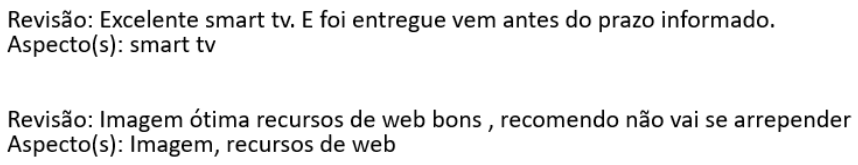

Figura 3. Exemplos de Sentenças e Aspectos

A outra base uilizada é a ReLi [Freitas et al. 2014], essa base contém comentários sobre 13 livros de 7 autores diferentes, na língua portuguesa. Essa base conta com 1.435 revisões, 10.507 sentenças, 1.983 aspectos marcados, sendo 493 distintos. Ao todo são 16.2% de sentenças com aspectos marcados.

4. Estratégias Adotadas

A tarefa de extração de aspectos é tratada como uma tarefa de classificação, uma vez que, para cada *token* de uma sentença, deseja-se atribuir o rótulo de: aspecto ou não-aspecto, isso significa que, a saída possui a mesma dimensão que a entrada após o pré-processamento. Por tanto, para solucionar este problema, não necessita-se do *decoder*, por conta disso, foi utilizado arquitetura BERT.

Para este trabalho, utilizamos três modelos pré-treinados, que estão disponíveis gratuitamente na internet, são eles:

- bert-base-portuguese-cased[1] - bbpc: Esse modelo foi pré-treinado utilizando a coleção de documentos BrWaC (*Brazilian Web as Corpus*) para o idioma português. Essa versão conta com 12 camadas de *encoders*, 768 é o comprimento (dimensão) do *embedding* produzido, 12 *attention-heads* e 108M de parâmetros.
- bert-large-portuguese-cased[2] - blpc: Semelhante ao anterior, porém, com 24 camadas de *encoders*, 1.024 comprimento do *embedding*, 16 *attention-heads* e 333M de parâmetros.
- bert-base-multilingual-cased[3] - bbmc: Este modelo foi pré-treinado para 104 idioma diferentes, incluindo o português, com dados provenientes do Wikipedia. Conta com 12 camadas de *encoders*, 768 é o comprimento (dimensão) do *embedding* produzido, 12 *attention-heads* e 177M de parâmetros.

O modelo espera receber os dados em formato numérico, por conta disso, é necessário que as frases passem por pré-processamento transformando as palavras em valores numéricos. Para cada *token* é atribuído um índice correspondente. O conjunto de *tokens* e índices formam um vocabulário.

[1]Disponível em: https://huggingface.co/neuralmind/bert-base-portuguese-cased
[2]Disponível em: https://huggingface.co/neuralmind/bert-large-portuguese-cased
[3]Disponível em: https://huggingface.co/bert-base-multilingual-cased

A etapa de pré-processamento dos dados conta então com seguintes passos: formatação dos dados, tokenização, alinhamento do rótulo de saída para os seus *tokens* correspondentes e conversão dos *tokens* em valores numéricos para que o modelo possa processá-los. Para cada modelo pré-treinado, a tokenização empregada é a mesma utilizada durante o pré-treinamento de cada um destes modelos.

5. Experimentos

Para utilizar modelos descritos na seção anterior, utilizou-se a biblioteca *Transformers*, que fornece facilidade de importação dos modelos pré-treinados e para o pré-processamento dos dados, utilizou-se a biblioteca *Datasets*.

Todos as bibliotecas mencionadas estão disponíveis para linguagem Python, e portanto, todos os experimentos foram executados utilizando esta linguagem. Os scripts foram todos executados utilizando o ambiente da ferramento Google Colab, que dispõe de 13Gb de RAM e GPU para processamento acelerado. As informações técnicas da CPU e GPU que estão disponíveis neste ambiente não são abertas, porém, a Google apresenta um *benchmark* para processos de convolução de imagens onde a GPU padrão obteve processamento 35 vezes mais rápido do que a CPU.

Todos os códigos utilizados para executar os experimentos mencionados neste trabalho podem ser acessados no GitHub[4].

6. Resultados

Os modelos foram treinados e avaliados cada um cinco vezes, sendo que a separação de dados em treino, validação e teste foi diferente em cada uma destas execuções. Os dados foram separados em 80% para treino, 10% para validação e 10% para teste.

Tabela 1. Resultados Obtidos com o Conjunto de Dados TV

Modelo	Precisão	Revocação	F1
bbpc	0.743	**0.829**	0.783
blpc	0.769	0.825	**0.795**
bbmc	0.704	0.793	0.746
CRF*	**0.817**	0.584	0.681

Tabela 2. Resultados Obtidos com o Conjunto de Dados ReLi

Modelo	Precisão	Revocação	F1
bbpc	0.487	0.505	0.496
blpc	0.502	**0.556**	**0.528**
bbmc	0.497	0.503	0.500
CRF*	**0.656**	0.379	0.480

Os resultados do modelo CRF são os valores médios obtidos por [Cardoso and Pereira 2020].

[4]Códigos disponíveis em: https://github.com/jcfneto/aspect_extraction

Em termos de precisão, o modelo CRF produziu o melhor desempenho, em termos de revocação, os modelos BERT pré-treinados com textos em português empataram, e em F1 o BERT-large foi o que obteve a melhor pontuação.

7. Conclusão

A arquitetura dos modelos BERT, que fazem aproveito de parte da arquitetura do Transformers, gerou bons resultados para a tarefa de extração de aspectos. Ao comparar os resultados obtidos neste trabalho para o conjunto de dados TV e ReLi com o trabalho de [Cardoso and Pereira 2020], o BERT ajustado obteve melhoras significativas em relação a revocação e F1, porém, com valores inferiores para precisão.

Referências

Cardoso, B. and Pereira, D. (2020). Evaluating an aspect extraction method for opinion mining in the portuguese language. In *Anais do VIII Symposium on Knowledge Discovery, Mining and Learning*, pages 137–144. SBC.

Chollet, F. (2021). *Deep learning with Python*. Simon and Schuster.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Freitas, C., Motta, E., Milidiú, R. L., and César, J. (2014). Sparkling vampire... lol! annotating opinions in a book review corpus. *New language technologies and linguistic research: a two-way Road*, pages 128–146.

Lindsay, G. W. (2020). Attention in psychology, neuroscience, and machine learning. *Frontiers in computational neuroscience*, 14:29.

Poria, S., Cambria, E., Ku, L.-W., Gui, C., and Gelbukh, A. (2014). A rule-based approach to aspect extraction from product reviews. In *Proceedings of the second workshop on natural language processing for social media (SocialNLP)*, pages 28–37.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.