ADRIANO AUGUSTO ANTONGIOVANNI

MINERAÇÃO DE ARGUMENTOS: APLICAÇÃO DE TRANSFORMERS NA CLASSIFICAÇÃO DE RELAÇÕES

São Paulo
2022

ADRIANO AUGUSTO ANTONGIOVANNI

MINERAÇÃO DE ARGUMENTOS: APLICAÇÃO DE TRANSFORMERS NA CLASSIFICAÇÃO DE RELAÇÕES

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro Mecatrônico.

Orientador:
Fabio Gagliardi Cozman

Co-orientador:
Victor Hugo Nascimento Rocha

São Paulo
2022

Dedico este trabalho à Carolina Saraiva Rector, meu grande suporte e chamada para ação. Além disso, dedico cada feito à minha família, que sempre colocou a educação como norte da minha criação.

AGRADECIMENTOS

Agradeço ao professor Fabio Gagliardi Cozman, que me acolheu em seu laboratório e apresentou diversos caminhos apra seguir com este trabalho. E ao meu coorientador Victor Hugo Nascimento Rocha, que me acompanhou por um ano, tendo sido muito importante nas tomadas de decisão do projeto.

Agradeço também a IBM, que na pessoa de Yoav Katz disponibilizou as ferramentas do projeto IBM Debater para que pudesse estabelecer valores de referência para este trabalho

Por fim, gostaria de agradecer a todos os colegas, professores e funcionários da Escola Politécnica e da Universidade de São Paulo que particiapram direta ou indiretamente em minha formação.

Sócrates é um homem;
Todos os homens são mortais;
Então Sócrates é mortal.

-- Aristóteles

RESUMO

Em 2021, a Organização das Nações Unidas estabeleceu de 2020 a 2030 como a década do oceano. Frente a este momento, o Centro para Inteligência Artificial, delimitou uma de suas áreas de pesquisa ao território marítimo brasileiro, denominado de Amazônia Azul. Diante desse cenário, estabeleceu-se o projeto Aprendizado de Máquina Enriquecido por Conhecimento sobre Dados do Oceano, que tem como um de seus objetivos criar o *framework* Cérebro da Amazônia Azul para executar tarefas de respostas à perguntas e argumentação. O objetivo desse projeto é combater a desinformação gerada por notícias falsas no tema. Sendo assim, o desenvolvimento de um argumentador computacional se mostra útil no sentido dessa área relacionar a experiência de usuário com modelos de argumentativos e que o raciocínio humano se de forma argumentativa. Dentro de um argumentador computacional, há uma necessidade de se aprimorar técnicas de mineração de argumentos necessárias para embasar uma argumentação. Em especial, destaca-se a necessidade de desenvolver a tarefa de classificar a relação de ataque ou suporte de um argumento em relação a um determinado tópico. Comercialmente, a IBM disponibiliza o IBM Debater, ferramenta de argumentação computacional que conta com um módulo de Mineração de Argumentos e uma ferramenta de classificação de frases argumentativas em relação a um tópico, com última publicação sobre seu funcionamento em 2017. Desde então, diversas técnicas surgiram para o Processamento de Linguagem Natural, como é o caso da técnica Transformers, sendo que não foram encontradas evidências, segundo a pesquisa feita na literatura, dessa sendo aplicada para a classificação de frases argumentativas em relação a um tópico. Desta maneira, este trabalho de conclusão de curso propõe um novo método para classificação de argumentos em relação a um tópico. Este novo método é baseado na arquitetura Transformer e foi aplicado por meio do modelo BERT para duas diferentes metodologias: utilizar o tópico e a frase argumentativa como uma entrada única do modelo; classificar o sentimento do tópico, da frase argumentativa e da frase argumentativa em relação ao tópico, como é descrito no último artigo publicado pelo projeto Debater da IBM sobre classificação e argumentos. Essas duas metodologias foram avaliadas entre si e com um valor de referência, obtido por meio da execução da ferramenta correspondente a classificação de argumentos em relação a um tópico do IBM Debater. Por meio dessa avaliação, foi observada uma melhora significativa de acurácia, precisão F1-Score, recall e especificidade da nova metodologia proposta em relação à ferramenta disponível comercialmente e um desempenho superior em relação a outra metodologia testada.

Palavras-Chave – Mineração de Argumentos; Classificação de Argumentos; Ataque e Suporte em Argumentos; Transformers; BERT.

ABSTRACT

In the context of the ocean decade established by UN, the C4AI created the project Knowledge-Enhanced Machine Learning for Reasoning about Ocean Data that has as one of its objectives to create the Blue Amazon Brain that will execute tasks such as question answering and argumentation. This project aims to combat misinformation from fake news. In that sense, the development of a computational argumentation tool is the path chosen, because this area can relate user experience with argumentative models and human reasoning is argumentative. In a computational argumentation tool, there is a need to improve argument mining, in order to provide support to the argumentation process. In particular, the task of classifying the support and attack relation between an argument and a topic needs to be better developed. There is in the market a tool developed by IBM, the IBM Debater, that consists of a computational argumentation that has a module for detecting the mentioned relation. The last article about this module is from 2017. Since then, several Natural Language Processing tools emerged, such as Transformers, with no evidence found in the literature by the present project of those tools being applied in the task of finding the relation between an argumentative sentence and a topic. Given this scenario, this paper presents a new method for classifying arguments related to a topic. This method is based on the Transformer architecture and the model BERT. This method was evaluated against itself and based on a benchmark established by the execution of the corresponding IBM Debater module. This evaluation showed an improvement in accuracy, precision, F1-Score, recall, and specificity when comparing the proposed method and the corresponding module of IBM Debater.

Keywords – Argument Mining; Argument Classification; Attack and Support Arguments; Transformers; BERT

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Cobrindo mais de 70% da superfície terrestre, os oceanos e os ecossistemas que os integram orientam atividades humanas como o lazer e a alimentação, além de exercer importante papel na regulação climática [2].

Pensando na relevância dos oceanos e em mapear sua situação ao redor do mundo, a Organização das Nações Unidas (ONU) publicou em 2021 a Segunda Avaliação Mundial do Oceano. Essa fornece subsídios para ações referentes ao 14º objetivo de desenvolvimento sustentável da ONU [3], definindo o período de 2020 e 2030 como a Década do Oceano e mostrando que algumas atividades predatórias vêm trazendo riscos ao oceano [4].

Considerando a relevância do tema e na missão de avançar a Inteligência Artificial (IA) no Brasil, o laboratório C4AI [1] (Center for Artificial Intelligence, em português Centro para Inteligência Artificial) delimitou uma de suas áreas de pesquisa ao território marítimo brasileiro, denominado de Amazônia Azul (AA).

Referido termo foi cunhado em 2004 para denominar o território marítimo correspondente à Zona Econômica Exclusiva e Plataforma Continental brasileira, com área de aproximadamente metade de seu território terrestre (Figura 1) [6].

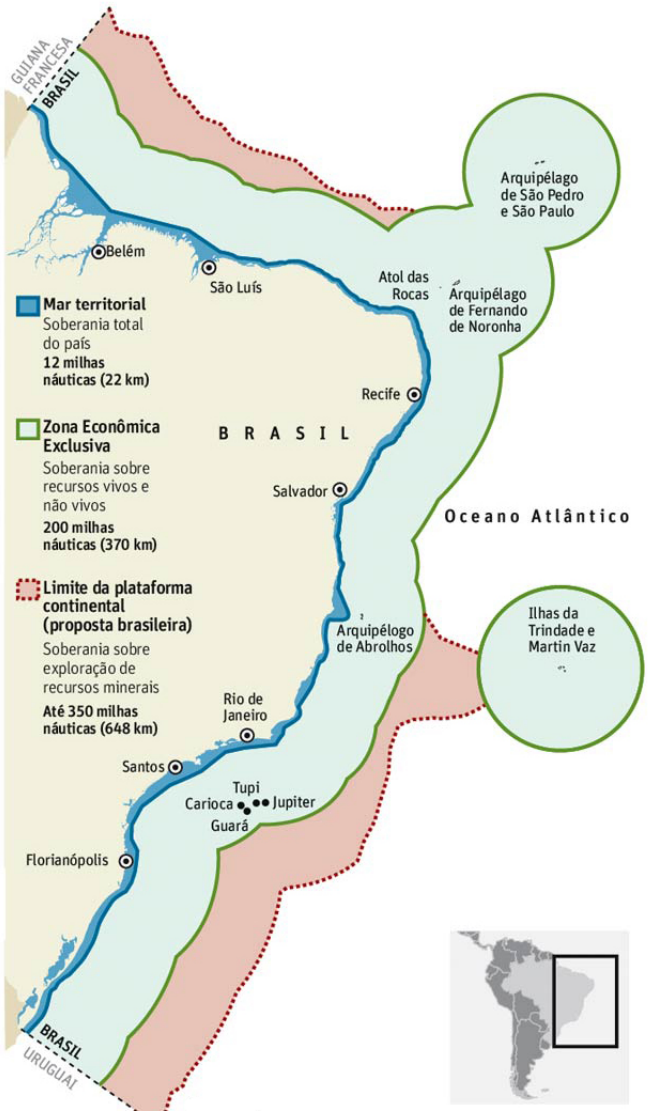
Diante desse cenário, o C4AI estabeleceu o projeto KELM (Knowledge-Enhanced Machine Learning for Reasoning on Ocean Data, em português Aprendizado de Máquina Enriquecido por Conhecimento sobre Dados do Oceano), que tem como um de seus objetivos criar o framework BLAB (Blue Amazon Brain, em português Cérebro da Amazônia Azul) para executar tarefas como QA (Question Answering, em português Resposta a Pergunta) e argumentação baseado em dados da AA.

Dentro da linha de pesquisa de argumentação do BLAB está sendo desenvolvido um projeto que visa criar um argumentador computacional para combater a desinformação e notícias falsas sobre a AA.

A Argumentação computacional é um campo que estuda o processo de debate e

[1] <https://c4ai.inova.usp.br/>

Figura 1: Mapa Amazônia Azul.



Fonte: Retirada de [5].

raciocínio, à luz de campos de conhecimento como filosofia, linguística, retórica e ciência da computação [7].

A escolha por essa linha para o problema de notícias falsas se dá por esse campo conjugar a representação necessária para aplicações relacionadas a usuários e modelos computacionais para argumentação [7], além de ajudar no seu combate identificando, rebatendo e encontrando suas fontes [8].

Considerando as fortes evidências de que a natureza do raciocínio humano seja argumentativo [8] e tendo em vista o projeto de argumentação computacional sendo desenvolvido no C4AI, há uma demanda para que se colete e classifique entre ataque e suporte argumentos sobre oceano e AA.

Nesse sentido, o presente Trabalho de Conclusão de Curso (TCC) aborda o campo de AM (Argument Mining, em português mineração de argumentos) para propor um método para classificar a relação de suporte ou ataque de uma frase argumentativa associada a determinado tópico.

Esse método poderá ser posteriormente utilizado para determinar essa relação em argumentos de AA e ser um dos componentes do argumentador computacional em desenvolvimento pelo C4AI.

Para desenvolver esse método, foi utilizada a arquitetura Transformer por meio do modelo BERT, uma vez que a literatura pesquisada para área de mineração de argumentos não abarca uma aplicação dessa arquitetura ou modelo, sendo os principais artigos encontrados inclusive de antes dessa arquitetura ter sido publicada.

Dentre as metodologias testadas, a que foi chamada de BERT Puro, que tem em uma única entrada uma frase argumentativa acompanhada de seu tópico, foi a que obteve melhor desempenho, tendo superado os *benchmarks* estabelecidos e, portanto, se mostrando como um modelo melhor que os disponíveis comercialmente para essa tarefa.

Nas sessões seguintes serão apresentados: o embasamento teórico deste projeto na revisão bibliográfica; como esse projeto foi desenvolvido, quais foram os parâmetros testados e as métricas nas quais esses parâmetros foram avaliados, apresentando as ferramentas, métodos e metodologias utilizadas; o desenvolvimento de cada modelo; os resultados de cada modelos; e as conclusões que se podem tirar deste trabalho.

Todo o código desenvolvido para executar as tarefas mencionadas está disponível aqui[1].

[1] <https://github.com/adriano-antongiovanni/ArgumentMining>

2 REVISÃO BIBLIOGRÁFICA

Nesta parte do TCC serão apresentados estudos mostrando tanto a evolução quanto o estado da arte de diversos campos que embasam o trabalho. Dessa maneira, será tratado sobre argumentos, argumentação, argumentação computacional e suas ferramentas e mineração de argumentos e suas divisões.

2.1 Argumentos e Argumentação

A definição do que é o argumento remonta à antiguidade, sendo registrada por Aristóteles, na obra 'Analíticos Anteriores', como Silogismo, a estrutura de argumentos perfeita [9].

Esta estrutura é composta de premissas que devem ser sempre totalmente verdadeiras e podem ser singulares ou, preferencialmente, gerais, e cuja interação levam a uma conclusão.

Um exemplo dessa estrutura é o presente na própria obra Analíticos Anteriores:

Sócrates é um homem;
Todos os homens são mortais;
Então Sócrates é mortal.

Nesse, a primeira frase é uma premissa particular, que somada com a segunda frase, uma premissa geral, permite levar a uma conclusão sobre Sócrates.

A didática utilizada por Aristóteles para definir argumentos por meio de uma estrutura argumentativa e de estudar e descrever os componentes dessa estrutura foi amplamente difundida, sendo diversas as tentativas de expandir o que foi apresentado por Aristóteles por meio do acréscimo de mais elementos.

Em particular, diagramas argumentativos passaram de ferramentas didáticas para uma técnica conceitual de modelagem do argumento [8].

Whately propôs um dos primeiros desses diagramas argumentativos [10] sob a premissa de que a construção do diagrama era feita a partir de uma raiz, sendo a conclusão e os argumentos que levam a essa conclusão folhas conectadas a esta raiz.

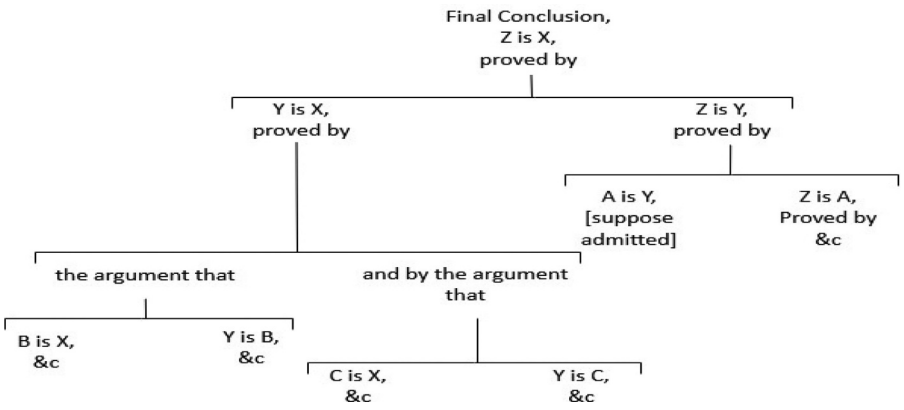
Aplicado ao exemplo de Aristóteles, a raiz seria a premissa de que que Sócrates é mortal e a ela estariam conectadas duas frases anteriores, as quais seriam as folhas dessa raiz.

No entanto, a abordagem de Whately se revela problemática, seja para a existência de argumentos mais complexos, em que a árvore pode crescer indefinidamente, ou na subjetividade da classificação da posição em que cada folha deve estar [8].

Posteriormente, Beardsley introduziu um conceito utilizado até hoje, a fim de representar que afirmações conectadas entre si geram argumentos conectados entre si. Essas conexões poderiam ser convergentes (convergent), divergentes (divergent) ou seriais (serial).

Todavia, na perspectiva atual, o modelo argumentativo impede que as afirmações sejam rebatidas [8], aproximando-as das premissas aristotélicas, o que faz com que o diagrama não seja válido para argumentos aristotelicamente imperfeitos.

Figura 2: Diagrama de Whately para análise de argumentos.



Fonte: Retirada de [8].

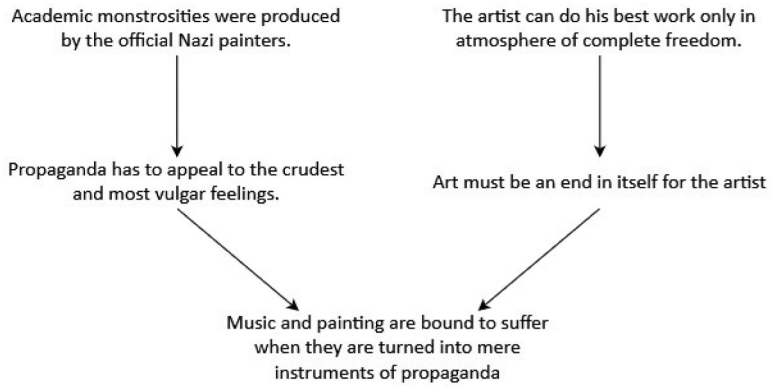
Na sequência, Toulmin[11] propôs que um argumento pode ser tão longo quanto uma sucessão de várias páginas dentro de um texto ou horas em um discurso e que um mesmo argumento pode ser apresentado em diversos formatos.

Contudo, há pontos chaves que de fato avançam entre a proposição de um problema inicial e uma conclusão sobre esse que se alongam muito menos, por parágrafos ou minutos.

Ainda, há frases nas quais uma estrutura argumentativa pode ser definida. É nessa estrutura que estudos de lógica são principalmente conduzidos e nessa ideia de forma

lógica que a validade de um argumento deve ser aceita ou rebatida, reforçando o estudo do argumento a partir de diagramas argumentativos.

Figura 3: Diagrama de Beardsley para análise de argumentos.



Fonte: Retirada de [8].

Adicionalmente, na obra 'Os Usos do Argumento' [11], Toulmin aponta que ao longo de uma argumentação, o argumento sendo utilizado para sustentar uma afirmação pode ser, por exemplo, de natureza estatística, não conferindo o caráter de uma verdade absoluta como é a premissa de Aristóteles.

Diante disso, Toulmin traz o seguinte exemplo, devidamente traduzido do inglês para o português, do que ele chama de 'quasi Silogismo':

Petersen é sueco;
Poucos suécos são católicos romanos;
Então Petersen não é, muito provavelmente, um católico romano.

O 'quasi Silogismo' é definido pelo autor para exemplificar sua expansão de componentes na estrutura argumentativa feita em cima dos conceitos de premissas e conclusões de Aristóteles: o dado (datum), que é a base para se fazer uma afirmação (claim); uma regra de inferência (warrant) que os liga; um elemento que mostra o quão certo se está da afirmação (qualifiers); as condições para que a afirmação seja verdadeira (rebuttal); e a justificativa para regra de inferência (backing).

Contudo, o modelo proposto por Toulmin é demasiado complexo e tem seus componentes descritos de maneira informal e ambígua, o que se apresenta como um desafio ao utilizá-lo em dados reais [12].

Um outro tipo de estrutura argumentativa é o IBIS (Issue Based Information System, em português Sistema de Informação com base em questões). Essa estrutura foi proposta

para dar suporte ao processo de decisão em um projeto por meio de debate e argumentação, estruturando e resolvendo questões levantadas pelos projetistas de sistemas [13].

Observa-se que o IBIS se distancia do Silogismo e 'quasi Silogismo' mostrados anteriormente ao focar seu processo argumentativo em um campo específico.

O IBIS é composto por quatro partes principais: as questões (issues), posições (positions), argumentos (arguments) e relações (relationships).

As questões são perguntas controversas com o intuito de coletar diferentes pontos de vista, as quais podem ser: factuais, quando a questão tem a ver com o que pode ser; detônica, quando se refere ao que é permitido; explanatória, quando se busca o porquê de algo ser como é; instrumental, que busca maneiras de mudar algo; e conceitual que busca o sentido das coisas.

Alguns exemplos de cada tipo de questão fornecido pelo autor, devidamente traduzidos do inglês para o português são:

Factual: É o caso para X?;
Detônico: Poderia ser o caso de X?;
Explanatório: O que causa X?;
Instrumental: Como X pode ser concebido?;
Conceitual: O que você quer dizer por X.

As posições abarcam tanto respostas categóricas positivas ou negativas e uma lista de possíveis respostas, como também a possibilidade de questionar a própria validade da questão proposta.

Os argumentos nesse caso são as evidencias que dão suporte ou que se opõe a uma posição, podendo uma posição ter vários argumentos e argumentos podendo se referir a várias posições.

Dessa forma, as relações que buscam estabelecer a maneira como questões, posições e argumentos se relacionam são a base do IBIS.

Pensando nas relações de um argumento com questões e posições, o autor indica as seguintes possibilidades: dar suporte, quando um argumento dá suporte a uma posição; objeção, quando um argumento objeta uma posição; e questionamento, quando um argumento questiona uma posição ou outro argumento.

Em função do exposto, observa-se que os modelos de argumentos e argumentação estão

presentes desde a antiguidade, sendo importante à lógica e ao discurso, motivo pelo qual o estudo nesse campo vem acrescentando novas nuances e complexidades aos seus modelos, a fim de expandi-los e adequá-los a ramos específicos do estudo.

Com a compreensão destes modelos e impulsionamento tecnológico, através do advento de computadores, muitos estudiosos neste campo construíram modelos que puderam ser representados computacionalmente, desenvolvendo o campo de argumentação computacional.

É interessante também observar que o próprio advento de computadores e da vontade de representar a argumentação computacionalmente simplificou modelos de argumentação.

2.2 Argumentação Computacional

A junção de modelos argumentativos com técnicas de IA, especialmente nos campos de representação de conhecimento e raciocínio não monotônico levou a criação de um novo campo denominado Argumentação Computacional [7].

Um trabalho considerado fundador desse campo foi o de Pollock, que em 1987 criou e implementou o sistema argumentativo e programa de computador OSCAR [14], o qual apresenta algumas simplificações em relação aos modelos apresentados anteriormente.

A primeira simplificação é a de que o sistema contempla apenas argumentos lineares, ou seja, os argumentos são uma verdade absoluta como uma premissa aristotélica ou são entendidos a partir de argumentos anteriores. A segunda é a de que o programa restringe o tempo para rodar, a fim de não rodar infinitamente. Por fim, a terceira é a de que argumentos a favor e contra determinada conclusão terão o mesmo peso, ainda que na prática haja argumentos mais fortes para cada um dos lados.

Mesmo com essas limitações, Pollock reporta ter conseguido resultados interessantes, na medida em que o sistema falhou nos casos em que não estavam abarcados pelas simplificações.

Um avanço no trabalho de Pollock foi o trabalho de Dung que, a partir da base de raciocínio não monotômica estabelecida por Pollock, acrescentou programação lógica para criar um argumentador computacional e demonstrou que tanto a programação lógica é similar a argumentação, quanto a argumentação é similar a programação lógica [15].

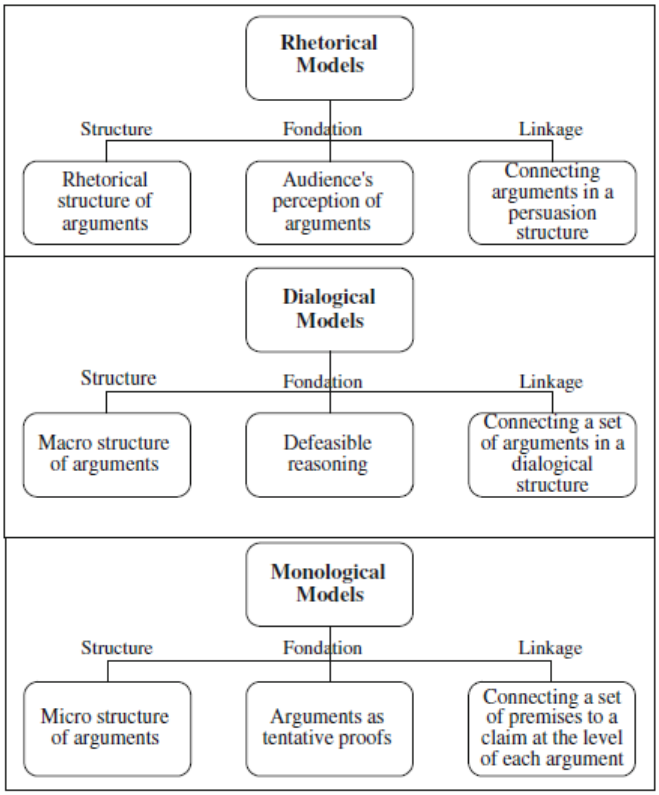
Em uma abordagem parecida com Dung de argumentação lógica, foi possível criar um sistema argumentativo que superava a primeira simplificação do sistema OSCAR de

Pollock, estabelecendo um sistema capaz de estimar a força de um argumento [16].

Diante disso, Pollock relacionou a força do argumento com o grau de justificativa que esse confere para a conclusão, provando que essa força do argumento não tem caráter probabilístico.

Apesar de desenvolver uma teoria extensa quanto a isso, ele resume ela em um exemplo simples: se 100 premissas independentes fossem altamente prováveis, e se essas fossem combinadas, a sua probabilidade conjunta seria menor, o que não corresponde a uma argumentação padrão na qual o conjunto de argumentos fortes na verdade fortalece uma conclusão.

Figura 4: Organização de Bentahar para modelos de argumentação.



Fonte: Retirada de [17].

Além disso, Pollock prova que dois argumentos fracos, que em princípio não seriam suficientes para suportar uma conclusão sozinhos, podem suportar uma conclusão juntos [18].

Em meio aos avanços na argumentação computacional, Bentahar propõe que os modelos desenvolvidos em diferentes trabalhos podem ser divididos em três categorias: retóricos (rethorical), dialógicos (dialogical) e monológicos (monological).

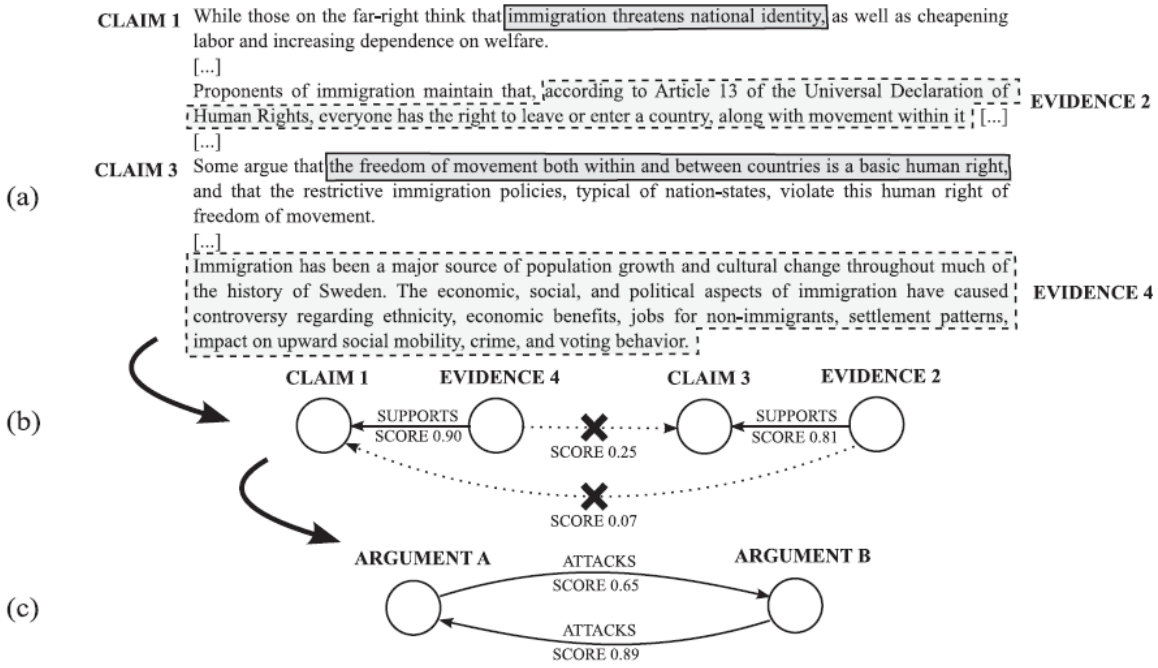
O sistema retórico coloca ênfase em uma audiência e na intenção de convencê-la, o

sistema dialógico descreve como argumentos estão conectados em suas estruturas e o sistema monológico foca na estrutura do argumento propriamente dita.

Outras classificações de argumentação computacional são a argumentação abstrata e argumentação estruturada. A primeira considera cada argumento como uma entidade única sem estrutura, sendo, portanto, um modelo dialógico útil para análise de relação de ataque entre argumentos.

Já a segunda se baseia tipicamente em modelos monológicos, propondo que cada argumento tem uma estrutura interna, o que é útil para extrair argumentos da linguagem natural, sendo, portanto, o tipo de argumentação utilizada na mineração de argumentos [7].

Figura 5: Organização de Bentahar para extração de argumentos.



Fonte: Retirada de [7].

Dessa maneira, o sistema monológico proposto por Bentahar pode ser utilizado para identificar argumentos dentro de um texto como mostrado nas partes (a) e (b) da figura 5, enquanto o modelo dialógico pode ser utilizado para estabelecer relações de ataque entre argumentos, como mostrado na parte (c) da mesma imagem [7].

Em razão do exposto, o trabalho destes autores, ao avançar na definição do que pode ser considerado um argumento e na sua representação em computador, estabeleceu técnicas computacionais essenciais para tarefas de Mineração de Argumentos como a identificação

do componente argumentativo e métodos para estabelecer a relação entre argumentos.

2.3 Mineração de Argumentos

Na obra 'De Inventione', Cícero define os cinco componentes da retórica clássica: invento (inventio), acordo (dispotio), estilo (elocutio), memória (memoria) e pronúncia (pronuntiato) [19].

O inventio pode ser definido como uma busca sistemática por argumentos a favor ou contra uma determinada afirmação, o que também confere o requisito de que um argumento só é relevante em uma argumentação se esse contribuir a favor ou contra uma afirmação [20].

Essa abordagem pode ser utilizada por NLP (natural language processing, em português processamento de linguagem natural) no problema denominado Argument Invention, que visa extrair argumentos de textos, como no trabalho de Walton [20], muito semelhante a metodologia empregada em AM.

Ao uso de NLP, somado ao KRR (Knowloge Representation Reasoning, em português representação e raciocínio baseado em conhecimento) e técnicas de IA, dá-se o de Mineração de Argumentos [21], cujo objetivo principal é extrair automaticamente argumentos de textos para alimentar sistemas de computação argumentativa [7].

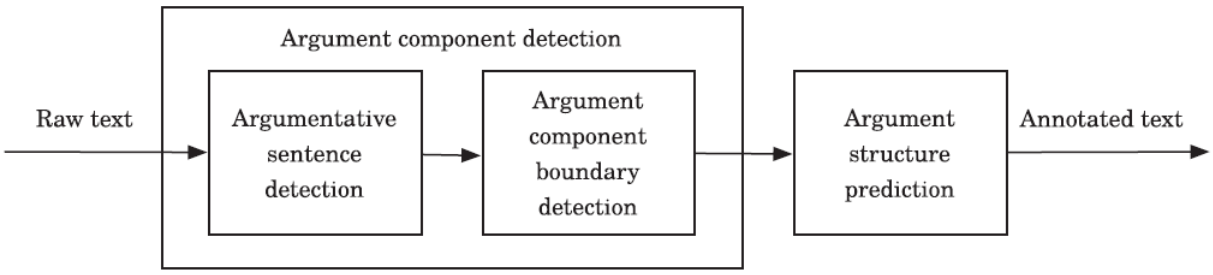
Cumpre ressaltar que a AM se assemelha ao NLP, especialmente em análise de sentimento e mineração de opinião. Contudo, possuem objetivos distintos: mineração de opinião trata de entender o que a pessoa pensa sobre algo, enquanto mineração de argumentos se interessa pelo porquê dessa opinião [22], ou seja, a análise do processo que leva seres humanos a racionalmente aceitar ou rejeitar algo [7].

Para tanto, como mostra o esquema na Figura 6, a partir de um texto qualquer, é possível identificar quais frases possuem argumentos, detectar as bordas desse argumento e depois fazer uma predição da estrutura desse argumento para gerar ao fim um novo texto estruturado, muitas vezes em forma de grafo.

Diante disso, é importante frisar que a AM enfrenta os seguintes problemas, conforme demonstrado na figura 6: granularidade de entrada (granularity of input), gênero de entrada (genre of input), modelo de argumento (argument model), granularidade do objetivo (granularity of target) e objetivo da análise (objective goals).

A granularidade indica o nível de detalhe em que o argumento será buscado, em

Figura 6: Organização de Lippi e Torroni para Mineração de Argumentos.



Fonte: Retirada de [7].

geral procurando frases em parágrafos. O gênero define qual será o tipo de dado que será colocado com entrada, sendo alguns exemplos notícias, artigos e peças jurídicas. Os modelos de argumentos são como os definidos em seção anterior. A granularidade do objetivo refere-se ao objetivo do processo de mineração quanto ao tamanho da estrutura do argumento desejada. Por fim, o objetivo refere-se a qual é objetivo da extração de argumentos, que poderá classificar ou relacionar esses argumentos [7].

Nesse sentido, para endereçar esses problemas, a áreas de AM delimita tarefas como a de detecção do componente argumentativo, a predição da estrutura do argumento e a classificação da relação entre dois argumentos ou entre um argumento e um tópico, que serão tratados a seguir.

2.3.1 Componente Argumentativo

A detecção do componente argumentativo é a motivação da área de AM e consiste em encontrar argumentos em um texto e definir seus limites, incorrendo em uma tarefa de classificação, que busca identificar se existem argumentos em frases de um determinado texto.

Essa classificação tem diversas abordagens, podendo ser um classificador binário, multiclasse ou um conjunto de classificadores binários.

Referidos classificadores incorrem em uma gama diversa de algoritmos de ML (Machine Learning, em português aprendizado de máquina) como Support Vector Machines (SVM), Naive-Bayes (NB), Random Forest (RF) e K-Nearest Neighbors (KNN), não existindo conclusão na literatura a respeito da melhor recomendação desses modelos [7]. Os recursos passados para o treinamento desses modelos, entretanto, tem maior uniformidade.

Um tipo muito utilizado é a analise de frequências, como por exemplo a representação do texto em *Bag-of-Word* (BoW), para medir a variável TI-IDF [7]: *Term Frequency* (frequência do termo) e *Inverse Document Frequency* (frequência inversa no documento).

Essa medição é feita para comparar a presença de determinada palavra em uma frase versus sua raridade em um texto. Apesar de ter limitações, como desconsiderar a ordem das palavras e a similaridade dessas, essa técnica é amplamente utilizada.

Além desse, importante citar técnicas relacionadas a frequências de classes gramaticais e sinais de pontuação.

Apesar de limitar a capacidade de generalizar alguns problemas, a contextualização da aplicação específica a qual o sistema de argumentação irá atuar pode trazer grandes benefícios na identificação de argumentos [7].

Esse é, inclusive, o modelo utilizado pelo minerador de argumentos do projeto IBM Debater [1], pelo qual um tema é dado e procura-se argumentos nesse sentido.

Ainda dentro do componente argumentativo, tem-se a tarefa de segmentação, para determinar as fronteiras do argumento, ou seja, o início e o fim de um argumento, para cada frase identificada como podendo conter um argumento.

Dessa forma, a granularidade da entrada deve ser considerada, sabendo-se de antemão a característica do texto para determinar se: uma frase ou a frase inteira coincide com argumento, se dois argumentos podem estar na mesma frase e se um argumento pode estar em mais de uma frase [7].

Sendo assim, não há homogeneidade quanto aos métodos utilizados para detecção desse argumento. Levy, por exemplo, utiliza o algoritmo de máxima semelhança para cortar uma frase em 10 partes com base nas palavras e pontuações que antecedem, começam, terminam e vem após a frase, sendo um algoritmo de Regressão Logística com um contexto definido utilizado para determinar qual das partes corresponde melhor a um argumento [23].

2.3.2 Predição da Estrutura do Argumento

Predizer a estrutura de argumentos, ou seja, a interligação entre argumentos, é uma tarefa complexa, uma vez que requer entender conexões e relações entre os argumentos detectados. Essa estrutura em geral é apresentada na forma de um grafo no qual geralmente

[1] <https://research.ibm.com/interactive/project-debater/>

estão representadas relações de suporte ou ataque [7].

A estrutura é também muito dependente do sistema de argumentação escolhido, sendo necessário identificar todos os componentes, o que se revela um desafio em sistemas como os de Toulmin, que são mais complexos por terem mais componentes a se classificar e que não tem a previsão de argumentos implícitos, o que obrigaria alguns componentes do argumento a serem construídos [7].

Dessa maneira, a predição da estrutura é feita com diferentes hipóteses simplificadoras, como no trabalho de Mochales, em que regras gramaticais foram utilizadas com os mesmos padrões de retórica e estrutura que textos jurídicos [24].

2.3.3 Classificação de Argumentos

Uma característica essencial para um argumentador computacional é conseguir selecionar, dependendo do contexto em que está inserido, argumentos contra ou a favor de determinada tese para apresentá-los em sua arguição.

Pensando nisso, Bar-Haim desenvolveu um trabalho de classificação de frases argumentativas [1] que, a partir de um tópico que contenha uma posição contrária ou a favor deste próprio tópico, classifica frases que contenham argumentos sendo contra ou a favor desse tópico.

Nesse sentido, ele propõe analisar frases em uma análise de sentimento, tanto do tópico quanto da frase argumentativa, para extrair dessas frases seu sentimento, seguida de uma análise de sentimento da frase argumentativa em relação ao tópico, para determinar consentimento ou não consentimento.

Por fim, determina-se a relação entre frase argumentativa e tópico, sendo contra ou a favor, pela combinação de consentimento ou não consentimento entre tópico e frase argumentativa e sentimento negativo ou positivo de cada frase.

Assim, frases podem ser classificadas como descrito na Tabela 1:

	Consensual	**Não Consensual**
Sentimento Positivo	Argumento a Favor	Argumento Contra
Sentimento Negativo	Argumento Contra	Argumento a Favor

Tabela 1: Classificação de Frases Argumentativas Segundo [1].

Contudo, segundo Bar-Haim há limitações para esse modelo, como em frases argumen-

tativas que não apresentam sentimento definido. Sendo assim, o autor aprimora a análise de sentimento, acrescentando a ela mais léxicos e expandindo a técnica para não se limitar à análise de sentimento, acrescentando características contextuais [25].

Para expandir a análise de sentimento, o autor utilizou componentes classificados de um léxico relacional para prever o sentimento de palavras não vistas. Ou seja, a partir de palavras classificadas previamente como positivas ou negativas, o autor analisou a relação dessas com palavras não mapeadas e classificou o sentimento dessas.

Para acrescentar características contextuais, o autor utilizou a premissa de que frases vizinhas em um texto tem o mesmo sentimento. Dessa maneira, Bar-Haim estabeleceu os seguintes novos componentes do modelo: componente de cabeçalho, que leva em consideração que frases dentro de uma mesma seção no texto tendem a concordar; componente de afirmação, quando uma frase contém uma palavra que denota o sentimento da frase, como exemplo, a palavra 'infelizmente'; componente de frases vizinhas, contando o número de frases com sentimento semelhante à frase sendo analisada que a precede; componente de afirmações vizinhas, quando duas afirmações em uma mesma frase ou parágrafo tem mesmo sentimento e, portanto, o autor junta essas.

Observa-se, portanto, que a classificação de argumentos é um campo restrito e que depende de premissas quanto ao tipo de classificação que se pretende fazer desses argumentos.

Como visto acima, a classificação poderá ser de uma frase argumentativa em relação à um tópico, mas em outras arquiteturas de argumentadores computacionais, classificações diferentes poderiam ter sido feitas.

2.3.4 Base de Dados

Uma base de dados para AM é, em geral, uma coleção de dados classificados por especialistas. Isso leva a um problema de tempo de construção e custo, uma vez que as bases de dados requerem uma quantidade grande de trabalho humano, em especial de especialistas em uma área para que esses dados tenham consistência [7].

Além disso, mesmo entre especialistas, identificar as fronteiras de argumentos, seus componentes e a maneira como se relacionam os argumento entre si e com uma afirmação é uma tarefa difícil [7].

Nesse sentido, muitas das bases de dados utilizadas, por escolha dos autores, já consideram que a tarefa de detecção de argumento foi realizada, não contendo, portanto,

partes não argumentativas que poderiam ser úteis para treinar modelos para achar esses argumentos [22].

A quantidade de conteúdo presente na internet também é foco de atenção para montagem de bases de dados. O maior banco de dados de AM desenvolvido é o da IBM [7]. Mais notadamente, este banco de dados busca afirmações e premissas dependentes de contexto em 33 tópicos, por 315 páginas da Wikipédia, encontrando cerca 2000 argumentos em aproximadamente 5000 frases [26].

2.4 Ferramentas Existentes para Argumentação

Com o crescimento no campo de AM, aumenta a necessidade de ferramentas específicas, como separadores gramaticais, classificadores de argumentos e extratores de argumentos.

Muitas ferramentas de processamento de linguagem natural também acrescentaram em suas funcionalidades tarefas específicas para AM como a extração de argumentos de textos.

Duas destas ferramentas dedicadas à classificação de textos são a BRAT [1] e a WebAnno [2]. Ambos são projetas de código aberto que são capazes de realizar tarefas como menção de elemento, extração de elemento, normalização e podem ser modificados para tarefas de AM, como detecção de argumentos e identificação de relações.

Uma ferramenta não encontrada online, mas que acrescenta a possibilidade de gerar grafos e de se fazer a classificação automática de um texto, é o GraPAT [27], que ainda acrescenta ferramentas para análise de sentimento e estrutura de argumento.

Já uma ferramenta que permite a classificação colaborativa de textos, além das ferramentas anteriores, é a GATE [3].

Uma ferramenta mais completa e com conexões com ferramentas mencionadas anteriormente, como o BRAT, é o Stanford OpenIE toolkit [4], disponível para diversas línguas, embora não em português. Ela é flexível, podendo ser conectada com linguagens de programação, executada na CoreNLP [5] ou baixada. Ainda, é capaz de executar tarefa como as mencionadas para outras ferramentas, além de poder mostrar a relação entre duas entidades.

[1] <http://brat.nlplab.org/index.html>
[2] <https://webanno.sfs.uni-tuebingen.de/>
[3] <https://gate.ac.uk/teamware/>
[4] <https://stanfordnlp.github.io/CoreNLP/>
[5] <http://corenlp.stanford.edu/>

Para além dessas ferramentas de uso geral, muitas das ferramentas de AM foram construídas com propósitos específicos, tanto comercialmente quanto didaticamente, sendo que poucas foram construídas com propósito de ser uma ferramenta de classificação como as de NLP.

Uma dessas ferramentas é a MARGOT [6], capaz de extrair argumentos de um texto automaticamente definindo as frases argumentativas desses e suas fronteiras. Essa ferramenta é utilizada pela CLAUDETTE [7], ferramenta que busca verificar termos de uso abusivos.

Ainda no campo de usos específicos, uma ferramenta didática é, por exemplo, o Arggugrader [8], que permite avaliar a qualidade de argumentos de estudantes.

Outra ferramenta é a de busca de argumentos é o Args [9], pela qual pode-se entrar com um tópico e esse retorna argumentos contra ou a favor d tópico.

Já para ferramentas comerciais de uso específico, destaca-se a Rationale, criada pela empresa Austhink Pty, que visa auxiliar na visualização e mapeamento de argumentos, e pela criação do bCisve, que ajuda em decisões empresariais formando hipóteses e mapas de decisão.

A ferramenta comercial mais completa e de melhor desempenho encontrada, é o IBM Debater (IBMD), um projeto de um sistema autônomo para debater com humanos [28].

Esse sistema já foi avaliado por expectadores em debates reais e teve o desempenho de superar diferentes níveis de discursos humanos, tendo apenas não superado a avaliação de um humano especialista em determinado assunto e em debates.

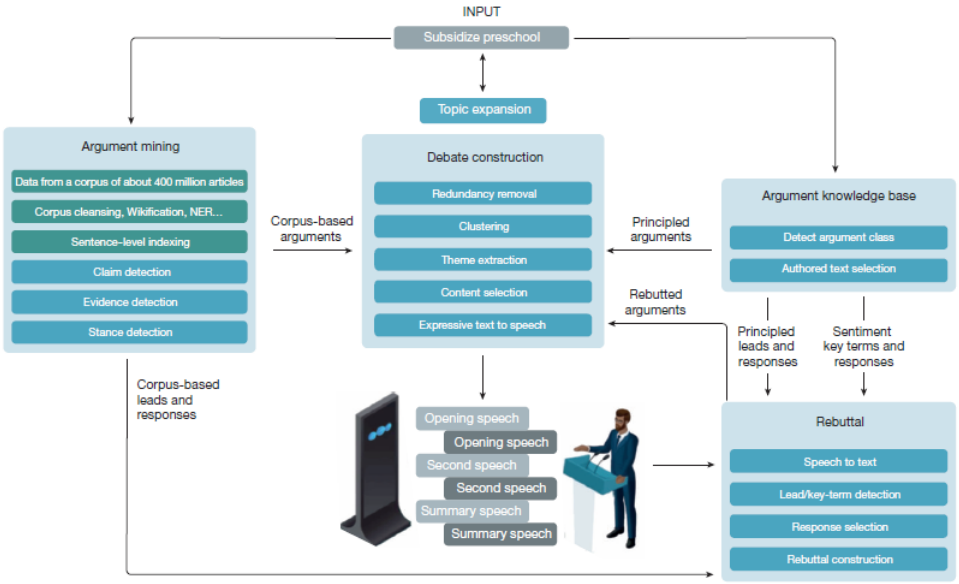
Contudo, ressalta-se a dificuldade e subjetividade de que a avaliação de um debate pode ter e que há uma diferença de avaliação entre um sistema de debate com humanos contra um sistema, por exemplo, para jogar um jogo como xadrez: no que tange que um sistema baseado a superar um humano em um jogo, esse consegue usar táticas até então incompreendidas por humanos, enquanto, dado o propósito do sistema argumentativo proposto ser para apresentação para humanos, esse sistema não pode usar técnicas que não sejam compreendidas por esses.

O sistema de debate IBMD é composto por quatro partes principais, cada uma dedicada a uma tarefa que compõe o debate: Argument Mining, Argument Knowlodge

[6] <http://margot.disi.unibo.it/>
[7] <http://claudette.eui.eu/index.html>
[8] <http://www.argugrader.com/>
[9] <https://www.args.me/index.html>

Base (em português, base de argumentos), Argument Rebuttal (em português, rebatedor de argumentos) e Debate Construction (em português, construtor de debate) [28].

Figura 7: Fluxograma do Funcionamento do IBM Debater.



Fonte: Retirada de [28].

A parte de AM é de especial interesse para este projeto, uma vez que executa tarefas como detecção de frases que contenham afirmações, fronteiras de afirmações, qualidade de argumentos e classificação entre prós e contras [29].

O detector de frases que contenham afirmações detecta se dentro de uma frase há uma afirmação que se relacione com determinado tópico. Uma vez detectada essa frase, um método detecta as fronteiras da afirmação e assim pode-se determinar a qualidade do argumento ou alimentar a entrada de um programa que determine se trata de um argumento a favor ou contra determinado tópico. Todos esses serviços do IBMD estão disponíveis em APIs que são gratuitas para fins acadêmicos [29].

Os principais desafios para melhora do IBMD são erros locais, como identificar e classificar uma frase argumentativa, adição de elementos que não estão contextualizados ou são incoerentes com a narrativa [28].

Acrescenta-se aqui que todo o desenvolvimento do IBMD é feito em língua inglesa, ou seja, todos os argumentos, modelos argumentativos, de argumentação computacional e avaliação de desempenho estão limitados a esse idioma.

Em particular, a fim de avaliar as tarefas de classificação de frases, detecção de evidência e qualidade de argumento para as línguas espanhol, francês, italiano, alemão e holandês, o estudo de Toledo [30] realiza a comparação nessas tarefas e dessas línguas com

o inglês, traduzindo automaticamente os dados de entrada do IBMD.

Esse estudo mostrou para esse sistema a validade da metodologia traduzir-treinar, pela qual o texto em língua original é traduzido para língua em estudo, para as tarefas de classificação de frase e detecção de evidências. Contudo, houve perda de desempenho com essa metodologia para tarefas ligadas a qualidade dos argumentos.

3 OBJETIVO

O objetivo deste trabalho é propor um novo método para classificar a relação de suporte ou ataque de uma frase argumentativa associada a um determinado tópico.

Esse método é baseado na arquitetura Transformers e é aplicado com o modelo BERT. Com isso, está-se verificando se os novos paradigmas de NLP melhoram o desempenho de modelos na tarefa de classificar de uma frase argumentativa em relação a um tópico.

O método teve como entrada duas frases: uma afirmação em forma de frase argumentativa e um tópico contendo um tema e uma posição de suporte ou ataque a esse tema. A partir dessas, o classificador identificará se a afirmação é contra ou a favor do tópico determinado.

O método proposto teve desempenho comparado com o classificador de suporte ou ataque de argumentos em relação a um tópico disponibilizado por meio de API pela IBMD.

Para essa comparação, foi utilizada uma base de dados aberta que contem afirmações em forma de frase argumentativa, tópicos que contenham um tema e uma posição sobre esse tema e a classificação das afirmações em relação ao tópico entre contra ou a favor desse.

Dessa maneira, é possível comparar a classificação do novo método com a classificação feita pelo classificador do IBMD em termos de acurácia, precisão, especificidade e F1-Score.

Nesse processo de comparação foram testados diferentes classificadores, visando avaliar como diferentes técnicas de processamento de linguagem e suas combinações afetam o desempenho de modelos de aprendizagem de máquina para classificar argumentos, entre dando suporte ou rebatendo determinado tópico.

Portanto, uma vez finalizado este projeto de TCC tem-se uma maior clareza de como diferentes técnicas de processamento de linguagem e aprendizado de máquina podem ser utilizadas na tarefa de classificar argumentos entre contra ou a favor de um tópico.

Além disso, sabe-se qual é o desempenho dessas comparadas com a ferramenta conside-

rada no estado da arte, o classificador do IBMD. Com isso, o C4AI tem um novo método para realizar essa tarefa.

4 FERRAMENTAS E MÉTODOS

Além das áreas e tecnologias apresentadas na seção de revisão bibliográfica, foram selecionadas ferramentas que serão diretamente implementadas nos testes deste TCC para um maior aprofundamento, servindo como embasamento teórico paras as decisões de projeto.

4.1 Processamento de Linguagem Natural

Apesar do NLP ser um campo muito amplo com diversas aplicações, esta seção se dedicará a apresentar técnicas de NLP com desempenho positivo comprovado em diversos campos de atuação, mas que tiveram pouca ou nenhuma aplicação no campo de mineração de argumentos.

Entende-se com isso que as arquiteturas e modelos apresentados a seguir são o estado da arte em seu domínio de atuação e pretende-se compreender melhor essas arquiteturas para testá-las no campo de argumentação computacional, mais especificamente na tarefa de classificar frases argumentativas entre dando suporte ou ataque a um tópico.

4.1.1 Transformer

O modelo Transformer foi proposto por Vawani como uma arquitetura que propositalmente não faz uso de modelos de recorrência e convoluções, baseando-se unicamente no mecanismo de atenção (*attention mechanism*), sendo a explicação de como o modelo e arquitetura funcionam presentes nesta seção publicadas em seu artigo [31].

Apesar de avanços em modelos de linguagem com base em redes neurais recorrentes, referidos modelos estavam sendo desenvolvidos com base em modelos de codificação e decodificação sequencial, motivo pelo qual têm uma alta dependência da posição de cada camada codificada, o que impede a paralelização do treinamento, gerando limitações de memória para entradas longas e dificulta modelar dependências entre entradas que estejam

posicionalmente distantes.

Para contornar isso, pode-se utilizar um modelo totalmente dependente do mecanismo de atenção e sem recorrência ou convolução, o que torna possível criar relações globais e paralelizar o processamento. Essas duas características levam a tempos de processamento menores e a possibilidade de relacionar entradas posicionalmente distantes.

Um dos pontos chaves para se obter as vantagens expostas anteriormente é manter constante o número de operações independente da distância da posição entre duas entradas. Isso leva à uma consequência negativa da redução da resolução efetiva do modelo, o que é contornado por meio do uso da atenção multi-head, que será explicada adiante.

Dessa maneira, considerando a aplicação original de Transformer para tarefas de NLP, a motivação para criação desse modelo foi, para além da redução do custo computacional, a de se conseguir estabelecer uma relação entre dois elementos de uma mesma entrada que estejam posicionalmente distantes.

Considerando a situação em que frases sejam as entradas, um modelo Transformer consegue estabelecer uma relação entre duas palavras de frases diferentes com mais facilidade que outros modelos.

Isso ocorre pois, a arquitetura Transformer é baseada em uma estrutura de codificação de entrada com decodificação na saída. Sendo assim, a entrada é mapeada em uma representação simbólica e transformada em uma representação contínua.

Cada elemento dessa representação contínua é processado de maneira auto regressiva, isolada e levando em conta processamentos feitos anteriormente de modo a gerar uma saída com uma nova sequência de símbolos.

Ainda na situação de uma frase como entrada do modelo, a frase será transformada de sua representação simbólica, ou seja, das palavras que a constituem, para uma representação contínua, ou seja, por números e símbolos internos que representem cada uma ou um conjunto das palavras da frase original.

A figura 8, ilustra o processo para um modelo da arquitetura Tranformer que será apresentado nesta seção 4.1.2.

A partir da Figura 8, observa-se que uma frase de entrada é dividida em suas palavras ou conjunto de palavras. Em seguida, são feitos processos específicos ao modelo, como acrescentar elementos reservados na entrada, usados para tarefas como indicar início e fim de uma frase.

Figura 8: Processo de Tokenização do modelo BERT.

Fonte: Retirada de [32].

Na penúltima etapa, as palavras são transformadas na representação contínua referente a cada modelo. A última etapa do processamento é também específica ao modelo proposto.

A etapa principal da arquitetura do modelo Transformer é a função de atenção, que na figura 9 observa-se como estando acoplado às camadas de codificação e decodificação explicadas anteriormente.

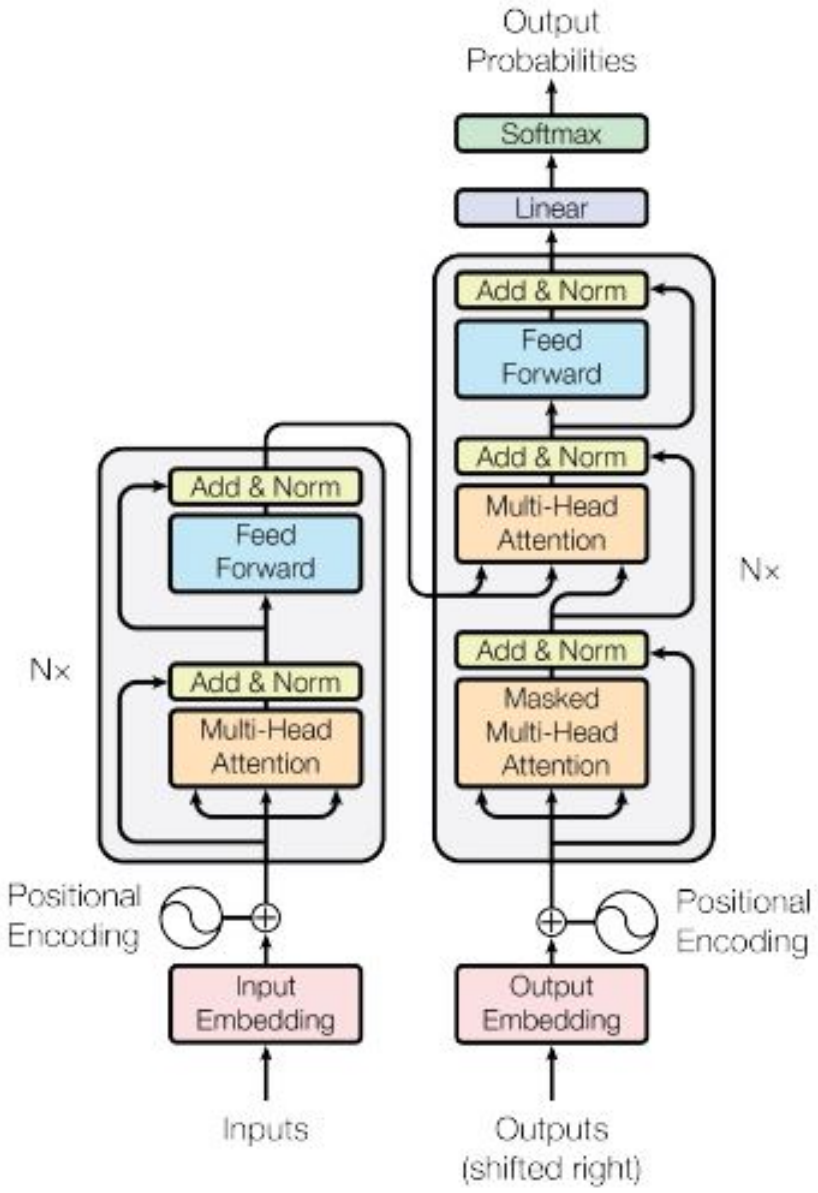
Uma função de atenção mapeia uma consulta e estabelece um par de valor e chave para uma saída. A saída então é processada como a soma ponderada dos valores em que o peso de cada valor é dado por uma função de compatibilidade proveniente da consulta de uma chave correspondente.

A função atenção implementada no modelo Transformer consiste em uma entrada de consultas e chaves de dimensão K e valores de dimensão V. Esses valores são aglutinados em uma matriz Q e processados pela seguinte fórmula:

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

Para implementar a função atenção, os valores de consulta, chave e valores foram projetados linearmente pela matriz W_i^Q h vezes a partir de projeções aprendidas para cada valor de K, V e Q.

Figura 9: Arquitetura do modelo Transformer.



Fonte: Retirada de [31].

Dessa maneira, quando o modelo está tratando de um determinado elemento de uma entrada, como uma palavra dentro de uma frase, são atribuídos a essa entrada um vetor de consulta, um de chave e um de valor.

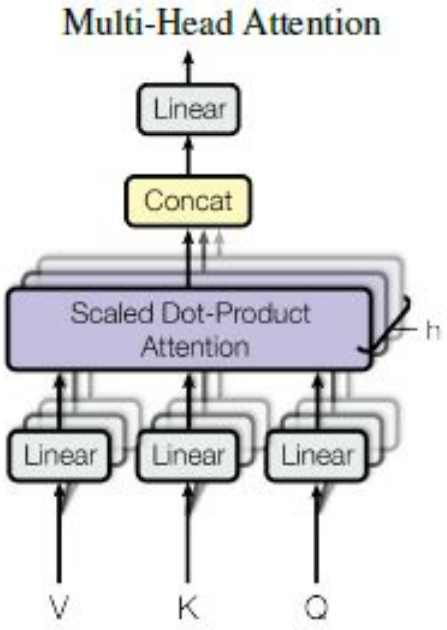
Esses vetores são processados segundo a fórmula acima para criar um vetor de atenção, que nada mais é do que um conjunto de pesos que deve ser dado a cada palavra de uma frase em relação a uma palavra sendo processada.

Esse mecanismo induz o modelo a dar mais peso ao elemento analisado e aos elementos próximos a ele quando está processando um texto, fazendo com que o treinamento de um modelo baseado nessas informações crie conexões mais informadas entre os elementos.

A atenção multi-head permite que o modelo trate informações de diferentes subespaços em diferentes pontos ao mesmo tempo. Esse processo permite que os valores projetados possam ser processados, concatenados e novamente projetados para se obter um valor final, como mostrado na fórmula a seguir e na Figura 10:

$$MultiHead(Q, K, V) = Concat(h_1, ..., h_n)W^O$$
$$h_i = Ateno(QW^O, KW_i^K, VW_i^V)$$

Figura 10: Atenção multi-head.



Fonte: Retirada de [31].

De maneira prática, a atenção multi-head faz com que múltiplos elementos ou palavras de uma mesma entrada ou frase possam ser processados simultaneamente e gerem um vetor de atenção cada.

A partir disso, os vetores de atenção são multiplicados entre si de maneira a gerar um único vetor, que definirá os valores de atenção para aquelas palavras ao final daquele processamento, permitindo um processamento mais rápido e estabelecendo uma conexão com palavras mais distantes da posição da palavra sendo processada.

Para garantir que durante todo o processo a ordem das posições de entrada seja mantida na saída, a entrada tem uma informação de codificação de posição. Essa codificação é feita utilizando-se funções *sen* e *cos* e a informação do tamanho da entrada, o que permite o modelo se adaptar a diversos tamanhos de entrada.

Quanto a complexidade computacional, uma camada de atenção adicional aumenta de forma constante $O(1)$ a quantidade de operações executadas e de conexões entre entrada e saída. Já um modelo recorrente o aumento de complexidade é de $O(n)$ e $O(n)$, respectivamente a entrada e saída e para uma rede convolucional é de $O(1)$ e $O(log_k(n))$.

A complexidade das operações de uma camada de atenção, recorrente e convolução são, respectivamente: $O(n^2d)$, $O(nd^2)$ e $O(knd^2)$.

Dessa maneira, sempre que o tamanho da sequência de entrada n for menor que a dimensão da representação d, com tamanho de Kernel k, o modelo Transformer será mais rápido do que um modelo convolucional ou recorrente.

Quanto a aplicabilidade, a arquitetura Transformer permite a criação de modelos pré-treinados que possam ser utilizados em domínios específicos, possibilitando que aplicações para as quais se tenham grandes quantidades de dados possam ser usadas em aplicações com uma quantidade inferior de dados.

Nesse sentido, a implementação da arquitetura Transformer pode ser feita por meio da biblioteca Transformers [33], que reúne recursos para distribuir, melhorar e colocar em produção modelos criados pela comunidade.

A biblioteca Transformers é projetada para executar toda o pipeline de NLP, a saber, processar dados, aplicar um modelo e fazer predições.

Além disso, todos os modelos dessa biblioteca são compostos de: um tokenizador (*tokenizer*), que transforma o texto em uma codificação esparsa de índices; um transformador (*transformer*, que transforma a codificação esparsa em um conjunto contextualizado; e um *head*, que usa o conjunto contextualizado para fazer predições.

Todas as arquiteturas implementadas em Transformers seguem a arquitetura de atenção multi-head de Tranformer, apesar de se diferenciarem em alguns aspectos, como a representação de posição. Ainda assim, todos os modelos seguem a mesmo hierarquia de abstração que o Transformer.

Os tokenizadores são implementados em classes tokenizadoras que variam de modelo para modelo, ou mesmo podem ser personalizadas para uma tarefa específica. Essas classes têm uma referência entre token e índice e podem ter tokens específicos para tarefas do domínio de determinado modelo, como um token de separação entre duas frases, para os casos de classificação de frases contextualizadas ou de QA.

Por fim, os heads acrescentam ao treinamento informações de camadas de saída adicionais e funções de perda além das que a arquitetura Transformer já prevê.

4.1.2 Modelo Baseado na Arquitetura Tranformer

Nesta seção será apresentado o modelo de representação de linguagem que contaa atualmente com métodos de classificação de textos e, portanto, foi considerado relevantes para o TCC. Além disso, esse modelo está entre os mais bem aceitos e desenvolvidos na representação de linguagem. Trata-se do modelo BERT.

O BERT (Bidirectional Encoder Representation from Transformers) é um modelo que usa MLM (Masked Language Model) na etapa de pré-treino para ser capaz de codificar o texto bidireccionalmente. Toda a explicação de como o modelo funciona presente a seguir foi proposta no artigo [34].

A decodificação bidirecional do modelo é possível, uma vez que o MLM mascara alguns do tokens dos dados de entrada e tenta predizer o *id* original dessa palavra mascarada somente pelo seu contexto.

Com esse processo, é possível utilizar tanto contextos provenientes de entradas anteriores quanto posteriores a entrada sendo processada no momento. De maneira prática, uma frase de entrada pode ter seu contexto definido por frases anteriores ou posteriores no modelo.

Isso é especialmente importante em tarefas que lidam com entradas de frases que necessitam de contexto, uma vez que esse contexto pode ser fornecido bidireccionalmente em casos reais.

Quanto à criação da máscara de atenção, essa pode ser observada na última etapa do processamento da entrada do BERT e define para cada entrada quais são os elementos que devem ser considerados e quais não devem ser considerados para aquela entrada. Essa definição pode ser observada na Figura 11.

Observa-se que essa definição é importante uma vez que a entrada do modelo BERT tem tamanho definido, de maneira que palavras são omitidas em frases com mais elementos que esse tamanho definido e posições nulas são acrescentadas em frases que tenham menos elementos.

Na prática, essa priorização é feita após um vetor de atenção ser multiplicado pela máscara de atenção, de maneira a atribuir pesos para cada elemento relevante de uma entrada enquanto ignora os elementos que não são relevantes.

Quanto ao processo de execução do modelo, as duas etapas principais para implementação do BERT são o pré-treino e o fine-tunning. No pré-treino, o modelo é treinado com

dados não rotulados. No fine-tunning, o modelo é iniciado com os parâmetros de pré-treino e os parâmetros são melhorados com dados catalogados.

Dadas essas etapas de execução, o mais comum é que a partir do treino de um novo modelo BERT, parta-se de um pré-treino disponibilizado pela biblioteca Transformers e, então, utilize dados rotulados para fazer uma predição. Isso se deve ao fato de que diversos modelos pré-treinados estão disponíveis, sendo que esses contêm um grande léxico otimizado para realização de tarefas de NLP.

A exemplo dos pré-treinos publicados originalmente e que serão objeto do presente TCC, tem-se o *bert-base-uncased* e o *bert-large-uncased*. Ambos são treinados com textos em minúsculas e em inglês, sendo que o primeiro conta com 12 camadas, 768 camadas escondidas, 110 milhões de parâmetros e 12 cabeças para execução da attention multi-head. Já, o segundo, conta com 24 camadas, 1024 camadas escondidas, 340 milhões de parâmetros e 16 cabeças para execução da attention multi-head.

Todas essas características do modelo podem ser utilizadas para executar diversas funções em NLP, sendo todas elas processadas segundo a mesma arquitetura no BERT. Essa arquitetura, para além dos detalhes específicos ao BERT descritos anteriormente, segue também as etapas descritas para o modelo Transformer.

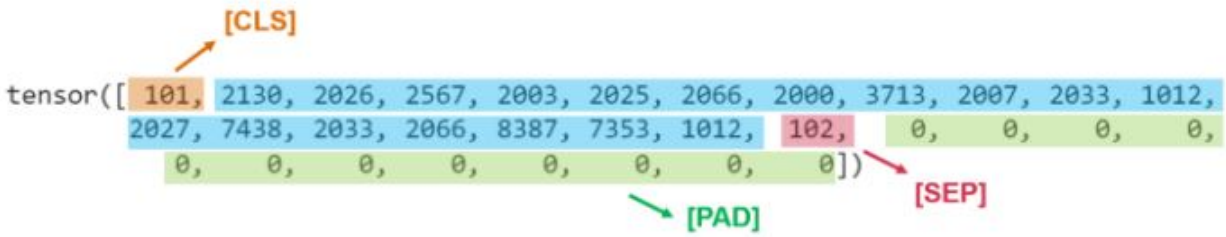
Quanto a entrada do modelo, por suportar uma grande variedade de tarefas, tem-se que a entrada do BERT é flexível, uma vez que é possível utilizar separadores reservados do modelo para, por exemplo, colocar duas frases em uma mesma entrada.

No modelo BERT, essa entrada é posteriormente tokenizada, utilizando-se o WordPiece [35], que conta com 30.000 (trinta mil) tokens de vocabulário. Nessa etapa, a representação simbólica da frase é passada para representação contínua.

Quanto ao tratamento da entrada, cumpre ressaltar que são acrescentados tokens reservados na entrada, antes de entrar no modelo. Para o modelo BERT, esses tokens são *[CLS]*, *[SEP]* e *[PAD]*. Os tokens *[CLS]* e *[SEP]* indicam, respectivamente, o começo de uma frase e o fim de uma frase. Já o token *[PAD]* é utilizado para gerar a máscara de atenção. Vale ressaltar que tokens não significativos são adicionados para adequar a frase ao tamanho de entrada definido no modelo.

Dessa maneira, uma frase de entrada com seus elementos simbólicos será processada para uma conjunto de elementos contínuos, nos quais cada elemento representam ou um token do léxico *WordPiece* ou um token reservado do modelo BERT. Essa representação contínua pode ser observada na Figura 11.

Figura 11: Exemplo de Entrada do modelo BERT.



Fonte: Retirada de [32].

Portanto, o BERT é um modelo que recebe uma ou mais frases como entrada, cada uma tendo suas palavras como elementos. Esses elementos são tokenizados e gera-se uma máscara de atenção. Por fim, por meio da arquitetura Transformer, são estabelecidas relações entre as palavras de uma ou mais frases e a saída de uma determinada tarefa. Em um exemplo de tarefa de classificação, essa saída é a categoria de determinada frase.

4.2 Base de Dados

A base de dados utilizada foi originalmente formulada pela IBM [36] com 33 tópicos que foram posteriormente estendidos para 55 tópicos [1]. A seleção dos tópicos foi feita aleatoriamente da base de dados da Associação Educacional de Debates Internacional.

Os 2394 tópicos sempre começam com a frase 'Essa casa acredita que' e as frases argumentativas dessa base de dados foram extraídas de artigos da Wikipédia.

Todas as frases argumentativas da base de dados são ou contra ou a favor de seu respectivo tópico. O processo para determinar se um argumento era contra ou a favor de um tópico foi feito a partir da classificação dada por 5 especialistas. A posição que recebesse mais voto seria a posição registrada na base de dados.

Contudo, para poder classificar a frase argumentativa em relação ao tópico, os especialistas precisaram primeiro acertar o sentimento do tópico e da frase argumentativa. Se a maioria não conseguisse fazer essa classificação corretamente, a frase era descartada.

Quanto ao balanceamento da base de dados, há 55.3% de frases a favor e 44.7% de frases contra o tópico. Um balanço semelhante se dá para o sentimento da frase argumentativa (55% positivo e 45% negativo). Contudo, em apenas 20% dos casos o sentimento da frase argumentativa em relação ao tópico se mostra negativa, o que demostra uma base de dados desbalanceada nesse sentido.

Dessa maneira, o próprio artigo analisa que essa métrica seria difícil de ser treinada por um modelo.

5 METODOLOGIA

O Projeto IBMD publicou uma base de dados contendo frases argumentativas em conjunto com os tópicos dessas frases e classificados por especialistas entre a favor ou contra este tópico [1]. O artigo que descreve a metodologia para se elaborar essa base de dados está descrito em 4.2.

Essa base de dados tem 21 colunas e 2394 linhas. Dentre essas colunas, destacam-se as de: *split*, que contém a divisão entre treino e teste proposta pelo artigo; *topicText*, que contém a frase de um tópico com uma posição de contra ou a favor em relação a esse tópico; *topicSentiment*, que contém uma classificação do sentimento, entre positivo e negativo, do tópico; *claims.claimCorrectedText*, que contém uma frase argumentativa sobre um tema relacionado ao tópico; *claims.claimSentiment* que contém a análise de sentimento dessa frase argumentativa; *claims.targetsRelation*, que contém uma análise de sentimento da frase argumentativa em relação ao tópico; e *claims.stance*, que contém uma classificação da frase argumentativa em relação ao tópico como sendo pró ou contra. Todas as colunas dessa base de dados podem ser observadas na Figura 12.

Figura 12: Colunas presentes na base de dados.

```
['topicId', 'split', 'topicText', 'topicTarget', 'topicSentiment',
 'claims.claimId', 'claims.stance', 'claims.claimCorrectedText',
 'claims.claimOriginalText', 'claims.article.rawFile',
 'claims.article.rawSpan.start', 'claims.article.rawSpan.end',
 'claims.article.cleanFile', 'claims.article.cleanSpan.start',
 'claims.article.cleanSpan.end', 'claims.Compatible',
 'claims.claimTarget.text', 'claims.claimTarget.span.start',
 'claims.claimTarget.span.end', 'claims.claimSentiment',
 'claims.targetsRelation'],
```

Fonte: Geração própria.

Um exemplo de um par de frases de tópico e afirmação argumentativa é o apontado a seguir. Neste, é possível ver que o tópico segue uma estrutura de apresentar um tema e oferecer uma posição contra ou a favor desse tema e que a frase argumentativa está relacionada a esse tema.

Nesse sentido, sempre que um tópico ou uma frase argumentativa for refletida adiante,

refere-se à essa estrutura. Além disso, observa-se que as frases originais estão em inglês, como todo o resto da base de dados.

> *Tópico: This house believes that the sale of violent video games to minors should be banned (esta casa acredita que a venda de jogos eletrônicos violentos deveria ser proibida);*
>
> *Frase Argumentativa: Exposure to violent video games causes at least a temporary increase in aggression and this exposure correlates with aggression in the real world; (A exposição a jogos eletrônicos violentos causa um aumento, ainda que temporário, na agressão e essa exposição está correlacionada com agressões no mundo real).*

Essa base de dados foi criada para testar uma metodologia de classificação de argumentos em relação à um tópico com base na análise de sentimento da frase contendo o tópico, da frase contendo o argumento e da relação de consentimento entre essas duas frases 2.4.

Segundo essa metodologia, cada frase seria classificada entre positiva (1) e negativa (-1) por si só e em relação a outra, e esses valores seriam multiplicados entre si para determinar se a frase argumentativa é contra ou a favor do tópico.

Contudo, essa metodologia foi publicada em 2017, sendo que apenas posteriormente, em 2018, o *Google* viria a publicar a arquitetura Transformer e o modelo BERT, que mostraram melhoras significativas para tarefas de NLP, inclusive dentro do próprio campo de análise de sentimento.

Ainda assim, não há referência nos artigos do projeto IBMD de que a metodologia para classificação de argumentos tenha sido alterada, apenas uma menção de que as tarefas englobadas dentro de mineração de argumentos, como a própria classificação de um argumento em relação a um tópico são cruciais para melhorar o desempenho do argumentador computacional desenvolvido no projeto.

Mesmo além do projeto IBMD, não foram encontradas referências que aplicassem Transformer na tarefa de estabelecer uma relação contra ou a favor entre um argumento e um tópico que tenha uma posição sobre esse tópico na base de dados apresentada e em nenhuma outra base.

De todo modo, o presente TCC obteve acesso a API do IBMD e, portanto, pode-se testar o desempenho dessa API em classificar argumentos em relação à um tópico.

Tendo em vista esse cenário, o projeto aplicou a arquitetura Transformer, por meio

do modelo BERT, para propor um novo método de classificação de frases argumentativas entre a favor ou contra um determinado trópico.

Sendo assim, a base de argumentos e tópicos classificados mencionada anteriormente foi utilizada para treinar e testar o modelo BERT em duas diferentes metodologias.

A primeira metodologia visa replicar a metodologia proposta em [1]. Dessa maneira, o modelo BERT foi utilizado primeiramente para classificar o sentimento do tópico, da frase argumentativa e da frase argumentativa em relação ao tópico entre positivo ou negativo. Para classificar o sentimento da frase argumentativa em relação ao tópico, ambas as frases foram aglutinadas em uma única entrada do BERT.

O sentimento positivo, tal como na metodologia original, será expresso por 1 e o negativo será expressa por -1. Esses três valores serão multiplicados, de tal maneira que se o resultado for 1 a frase argumentativa será a favor do tópico e se o resultado for -1 a frase argumentativa será contra o tópico. Essa metodologia será referida doravante como Metodologia IBM.

Vale ressaltar que a base de dados tem o resultado de cada uma das três etapas mencionadas anteriormente, então pode-se avaliar individualmente as métricas de cada etapa.

A segunda metodologia é inspirada na aplicação do BERT para casos de resposta à pergunta. Nesses casos, uma pergunta e suas possíveis alternativas são colocadas como uma mesma entrada do modelo e é fornecida uma classificação que é a alternativa que responde corretamente essa pergunta. Dessa maneira, as tarefas de resposta à pergunta podem ser processadas em lotes contendo perguntas e respostas em uma mesma entrada.

Essa ideia de ter mais de uma frase como entrada do modelo foi estendida para o caso de classificação de frases argumentativas sendo contra ou a favor de um determinado tópico. É proposto neste TCC que essas duas frases sejam unidas em uma única entrada de um modelo BERT e seja fornecida diretamente a classificação da frase argumentativa ser contra ou a favor do tópico.

Dessa maneira, o modelo terá apenas uma entrada e uma classificação para cada par de frases, o que permite que o BERT retorne diretamente uma classificação de suporte ou ataque. Essa metodologia será referida doravante como Metodologia BERT Puro.

Percebe-se que a etapa de classificação da frase argumentativa em relação ao tópico entre positivo ou negativo e a metodologia do BERT Puro tem a semelhança de terem uma única entrada contendo a frase e o tópico para o modelo.

Contudo, para a metodologia IBM, a avaliação dessa classificação se dará em relação a análise de sentimento entre essas duas frases e, na metodologia do BERT Puro, a avaliação da classificação será feita baseada em se o argumento é contra ou a favor de determinado tópico.

Além da diferença de metodologia, foram testados também pré-treinos diferentes do BERT: ambas as metodologias foram aplicadas com os modelos pré-treinados *bert-base-uncased* e *bert-large-uncased*, para verificar as suas diferenças de desempenho e performance entre esses dois pré-treino para a tarefa proposta.

Esses modelos pré-treinados foram escolhidos por terem sido treinados com textos em inglês, que é a língua na qual serão fornecidos os dados de entrada e serem os casos bases explicitados no artigo original do modelo BERT [34], e portanto terem mais informações disponíveis sobre suas características.

Vale ressaltar que tanto a arquitetura Transformer, o modelo BERT e os dois modelos pré-treinados foram implementadas utilizando a biblioteca Transformers.

Além das metodologias serem testadas com dois dos pré-treinos do modelo BERT, foram testadas também duas separações de base de dados entre treino e teste. Uma é a originalmente proposta em [1] e explicada em 4.2, que visa testar se um modelo treinado com um conjunto de argumentos de determinados tópicos pode ser utilizado para classificar argumentos de outros tópicos. Para essa divisão, há 1039 dados de treino e 1355 dados de teste.

A outra divisão de dados proposta é uma divisão aleatória da base de dados entre 80% dos dados para treino e 20% dos dados para teste. Com isso, teremos 1915 dados de treino e 479 dados para teste.

Por meio da comparação entre essas duas divisões de dados, avaliou-se a performance do modelo em cada situação, bem como se um treinamento aleatório tem um desempenho melhor do que um treinamento baseado no aprendizado de determinados tópicos ser estendido para outros tópicos, como é a proposta da divisão presente na base de dados.

Todas os 8 treinamentos que foram feitos e suas variações entre metodologias, pré-treinos e divisão entre base de treino e base de teste estão sintetizadas na Tabela 1.

Quanto à avaliação dos modelos, essa foi feita primeiramente utilizando os dados de teste. Dessa forma, o modelo gerado retornou uma saída para os dados de teste que foi comparada com a resposta verdadeira quanto a acurácia, precisão, recall, F1-Score e especificidade.

Os modelos também foram avaliados quanto ao seu tempo de treinamento e foram registrado o parâmetro de *train loss*, que serviu de indicador de como um modelo poderia ser treinado mais vezes para ter resultados melhores.

Metodologia	**Divisão**	**Pré-Treino**
BERT Puro	**Artigo IBM**	**bert-base-uncased**
BERT Puro	**Artigo IBM**	**bert-large-uncased**
BERT Puro	**Aleatória**	**bert-base-uncased**
BERT Puro	**Aleatória**	**bert-large-uncased**
Metodologia IBM	**Artigo IBM**	**bert-base-uncased**
Metodologia IBM	**Aleatória**	**bert-large-uncased**
Metodologia IBM	**Artigo IBM**	**bert-base-uncased**
Metodologia IBM	**Aleatória**	**bert-large-uncased**

Tabela 2: Caracterísitcas dos modelos treinados no TCC.

Além dos fatores de desempenho de modelo, esses também foram comparados com um *benchmark*. Por ser considerada a ferramenta de estado da arte em termos de classificação da relação entre frases argumentativas e um tópico e também por ser uma solução comercial disponível no mercado, o classificador do IBMD foi utilizado para estabelecer um *benchmark*.

Esse *benchmark* foi determinado para todos os conjuntos de testes gerados aleatoriamente e o conjunto de teste definido na base de dados foi colocado como entrada do classificador do IBMD.

A resposta desse classificador será comparada com a classificação real de valores de referência de acurácia, precisão, recall, F1-score e especificidade para cada divisão feita na base de dados.

Os modelos gerados foram avaliados como modelos melhores ou piores que o estado da arte com base na comparação das métricas mencionadas anteriormente com o *benchmark*.

6 DESENVOLVIMENTO

Neste capítulo, será detalhado como as metodologias e avaliações mencionadas no capítulo anterior foram implementadas.

De maneira geral, todo o desenvolvimento se deu na linguagem Python 3, utilizando como IDE o Jupyter. A escolha da IDE deu-se pelo fato de o desenvolvimento ter sido feito de forma modular, separando a parte de tratamento de dados, separação dos dados, transformar dados em entradas dos modelos, iniciar os modelos, treinar os modelos, avaliar os modelos e estabelecer *benchmark*.

A separação dessas partes deu-se, para além de organizar o código, para garantir uma modularidade na execução desse, sendo que partes previamente estabelecidas para um modelo poderiam ser utilizadas para outro modelo sem a necessidade de executar todo o código novamente.

Isso foi importante sobretudo para a implementação da metodologia IBM, a fim de registrar as predições feitas das análises de sentimento das frases em cada etapa.

Vale ressaltar que em ambos os arquivos houve uma tentativa de se generalizar em um só programa todo o treinamento, mas por conta do exposto anteriormente essa tentativa foi ignorada para a execução de fato do programa, portanto, não será descrita.

Os códigos desenvolvidos podem ser encontrados aqui[1].

6.1 Tratamento e Segmentação de Dados

A base de dados utilizada foi disponibilizada no formato de *.csv*, sendo que nenhum tratamento adicional foi necessário para abrir o arquivo. Todo o tratamento da base foi realizado com a biblioteca Python *pandas*, possibilitando a criação de um dataframe a partir do arquivo.

[1] <https://github.com/adriano-antongiovanni/ArgumentMining>

Uma das colunas do dataframe é *claims.Compatible*, um indicador interno dessa base para sinalizar se uma linha de dados tem uma frase argumentativa coerente com o tópico. Nessas linhas, havia também valores nulos, então esses dados foram excluídos do dataframe tanto para atender a hipótese inicial do modelo de que a frase argumentativa tinha que ter relação com seu tópico, quanto para não lidar com os dados nulos.

Além disso, a classificação das frases argumentativas em relação ao seu tópico foi feita expressa com o tipo de dado *string* do Python por 'PRO' e 'CON'. Essas foram substituídas pelos valores 1 para 'PRO' e 0 para 'CON', para que a classificação pudesse ser compreendida pelo modelo.

Ainda no tratamento da base de dados, foi acrescentada uma coluna *text*, que junta o tópico com a frase argumentativa em uma única entrada, já com os separadores de frase próprios do modelo BERT.

Apesar do aspecto modular que um notebook confere, preferiu-se dividir em dois arquivos o treinamento dos modelos, um para o modelo do BERT Puro e outro para o modelo da metodologia IBM. Essa escolha se deu pelo fato de que cada método tem suas particularidades de execução e misturar essas particularidades só traria complexidade para o arquivo sem trazer nenhum benefício adicional.

Dessa maneira, a primeira diferença entre os dois arquivos dá-se na separação dos dados. No arquivo dedicado ao BERT Puro a separação é feita criando-se um dataframe *text* e um dataframe *labels*, representando as entradas de tópico mais frase argumentativa e os rótulos das entradas.

Já para o arquivo dedicado à metodologia IBM, são criadas duas listas *text* e um dataframe *labels*. A primeira contendo dataframes de texto do tópico, frase argumentativa e tópico mais frase argumentativa, e o segundo contendo os respectivos rótulos.

Ao final destas etapas, o programa está com os dados de entrada dos modelos organizados, mas ainda não totalmente tratados para que possam ser fornecidos a ele.

6.2 Preparação de Dado para Treino e Teste

6.2.1 Tokenização

Para preparar os dados separados para treino é preciso que cada frase seja tokenizada e tenha uma máscara de atenção associada. Para tanto, utiliza-se o método *tokenizer* da biblioteca Transformers.

Nesse momento, pode-se escolher treinar um modelo do zero, todavia, como comentado anteriormente, escolheu-se utilizar um modelo pré-treinado.

Essa escolha é feita manualmente no arquivo para o BERT Puro e é feita como entrada do usuário para o arquivo da metodologia IBM. Para que essa escolha seja possível, é preciso associar o método *tokenizer* ao método *from pretrained*.

Logo em seguida, os tokens, a máscara de atenção e os rótulos de dados são transformados em tensores torch por meio da biblioteca Python *torch*. Isso é necessário uma vez que é um requisito do modelo que a entrada esteja nesse formato.

Dessa maneira, os dataframes e listas de *text* e *labels* dos arquivos são transformados em variáveis contendo os tokens das frases, com suas respectivas máscaras de atenção e rótulos no formato de tensores.

Passa-se, portanto, de dois dataframes para três tensores no programa BERT Puro e de duas listas com seis elementos cada para dezoito tensores no programa do método IBM.

Para organizar essas dezoito novas variáveis, estas foram nomeadas com três nomes separados por um sublinhado. O primeiro nome refere-se ao tipo daquela variável (token, máscara ou rótulo), o segundo nome referente a qual dado se trata essa (tópico, frase argumentativa ou relação entre duas frases) e o terceiro se refere à segmentação do dado feita (treino ou teste).

6.2.2 Separação entre Treino e Teste

A separação entre as bases de treino e teste para o arquivo da metodologia da IBM é feita na seção de segmentação das bases de dados. Essa escolha foi feita uma vez que se julgou que seria mais fácil separar os dados já rotulados entre treino e teste antes que esses passassem por qualquer transformação.

Já para a separação entre treino e teste do arquivo da metodologia BERT puro, essa foi feita utilizando-se o método *train test split* da biblioteca *sklearn*.

Essa divisão foi feita posteriormente à tokenização das entradas e, portanto, o retorno do método foram os *id* que deveriam ser usados para treino e teste, e não as bases divididas, sendo que a segmentação de fato entre treino e teste foi feita nos tensores.

Por fim, ambos os tensores foram colocados dentro de um método *DataLoader* da biblioteca *torch*, um para treino e outro para teste. A seleção de qual tensor colocar no *DataLoader* foi feita com uma entrada de usuário no caso do método da IBM, que

escolhe se deseja treinar para análise de sentimento do tópico, da frase argumentativa ou da relação entre as duas frases.

Além desses parâmetros, o *DataLoader* recebeu o parâmetro de *batch size*, que é o número de entradas aglutinadas para serem processadas de uma vez. Esse parâmetro foi definido como 16 para ambos os arquivos.

Dessa maneira, ao final da etapa de preparação dos dados para treino e validação, os dataframes que continham as frases e os rótulos de cada frase foram processados em um objeto da classe *DataLoader*, que será utilizado como entrada do modelo e contêm as frases tokenizadas, a máscara de atenção e os rótulos no formato de tensores.

6.3 Preparação para Treinamento do Modelo

Para que possa ocorrer o treinamento do modelo, esse deve ser inicializado e para manter a coerência entre as duas metodologias, a inicialização foi feita com os mesmos parâmetros nos dois arquivos.

Objetivando implementar a tarefa de classificação do modelo BERT, a biblioteca Transformers disponibiliza o método *BertForSequenceClassification*. Por meio deste, é possível que seja feito um treinamento do zero, contudo, como dito anteriormente, serão utilizados os modelos pré-treinados *bert base uncased* e *bert large uncased*, sendo preciso associar o método *BertForSequenceClassification* ao método *from pretrained*.

Com essa combinação, dois parâmetros importantes são colocados como hiperparâmetros do modelo: o pré-treino a ser utilizado, que varia conforme o treinamento desejado, e o número de rótulos de dados, que foi definido em todos os casos para dois rótulos, uma vez que se está querendo sempre determinar uma relação dual entre sentimento positivo ou negativo e argumento contra ou a favor.

Além da preparação do treinamento do modelo, é preciso definir também um otimizador para ser rodado junto com o modelo. Esse otimizador possui um parâmetro importante que é a taxa de aprendizado *learning rate*. O otimizador escolhido foi o AdamW.

A taxa de aprendizado foi determinada no valor de 0,00005 por [34]. Como não faz parte do escopo do trabalho encontrar a melhor taxa de aprendizado, a taxa ótima presente no artigo foi reproduzida para o otimzador do projeto.

Dessa maneira, após a etapa de preparação do modelo para treino, tem-se um modelo e otimizador definidos prontos para serem treinados.

6.4 Treinamento dos Modelos

Uma vez realizados todos os processamentos acima, a etapa de treinamento do modelo é direta, devendo-se definir apenas dois parâmetros: a máquina que irá rodar o programa (se GPU ou CPU) e o número de épocas pelas quais o modelo será treinado.

A máquina foi escolhida como sendo uma CPU, uma vez que se optou por um treinamento local e o número de épocas foi escolhido como 4, para balancear tanto o tempo de treinamento quanto a possibilidade de bons resultados.

Dessa maneira, a implementação do treinamento é feita de tal forma que o modelo seja colocado em modo de treinamento e esse seja repetido pelo número de épocas, respeitando os dados já aprendidos anteriormente. A avaliação desse modelo é feita tanto enquanto ele está sendo treinado, em cada época, quanto depois que ele é treinado.

6.5 Implementação da Avaliação de Modelos

Para avaliar o modelo, utiliza-se dentro do loop de épocas do modelo uma mudança do estado do modelo para teste. Essa mudança faz com que o estado atual do modelo gere uma predição para os dados de teste, que são então comparados com os rótulos reais dos dados para se obter acurácia, precisão, recall especificidade. Além disso, a cada época é retornado o tempo de treinamento e o *train loss*.

Com essa avaliação implementada, é possível acompanhar como o modelo está evoluindo a cada época e entender se pode estar ocorrendo algum problema de *overfitting* nos dados.

Para o arquivo da metodologia BERT Puro, esse processo é o suficiente para avaliar o modelo, uma vez que o resultado da última época será o resultado do modelo. Ainda assim, foi implementada uma predição dos valores da base de teste para comparação dessas predições com o rótulo dos dados.

Contudo, para a metodologia IBM, é preciso que sejam feitos três treinos, um para cada análise de sentimento, o que impede a avaliação da metodologia como um todo por época. Ainda assim, essa foi registrada para se observar a evolução de cada modelo no decorrer de cada época.

Para contornar esse problema, após o treinamento de cada modelo foi implementada uma predição dos valores da base de teste e essa predição foi registrada em uma lista dentro de outra lista, convertendo o rótulo 0 para -1.

Dessa maneira, ao final do treinamento dos três modelos basta multiplicar posicionalmente os valores de cada elemento de cada lista para se obter a predição segundo a metodologia IBM.

A escolha de qual modelo será treinado é uma entrada do usuário, sendo que, uma vez escolhida, seleciona automaticamente os dados que serão utilizados no modelo e guarda a resposta do modelo no local correto.

Uma vez que essa etapa de treinamento tenha sido executada para as duas metodologias, com os dois tipos de divisão dos dados e com os dois modelos pré-treinados, é possível que os modelos sejam avaliados entre si para determinar qual dos treinamentos obteve melhores resultados. Ainda assim, fica faltando estabelecer de uma comparação absoluta com uma ferramenta já presente no mercado.

6.6 Estabelecimento do *Benchmark*

Estabelecer o *benchmark* é importante para que os modelos gerados possam ser comparados de maneira absoluta com ferramentas existentes e não apenas entre si.

Dessa maneira, escolheu-se a ferramenta do IBMD para classificação de argumentos contra ou a favor de um tópico. Essa ferramenta é comercial, podendo ser contratada por empresas, mas é disponibilizada gratuitamente para realização de trabalhos acadêmicos.

A disponibilização gratuita é feita por meio da requisição de uma chave de API diretamente para o time que desenvolve e mantêm o IBMD, sendo que o presente projeto conseguiu esse acesso, com o intuito de utilizar a ferramenta para estabelecer o *benchmark*.

Para implementar o classificador do IBMD e gerar o *benchmark*, a API de classificação do IBMD foi requisitada para classificar cada uma das frases ou pares de frases dos conjuntos de teste de cada modelo.

O retorno da API do IBMD foi então comparado com o rótulo dos dados para determinar sua acurácia, precisão, recall, F1-Score e especificidade.

Diante disso, a API foi requisitada para cada um dos quatro testes envolvendo uma divisão de treino e teste aleatória e uma vez para a divisão original da base de dados na medida em que, como os dados não se alteravam, não seria preciso reestabelecer um *benchmark* a cada treino para a divisão feita pela base de dados.

Com essas requisições, é possível comparar a acurácia, precisão, recall, especificidade e F1-Score dos modelos criados com o *benchmark* do IBMD e obter uma comparação

absoluta do desempenho do modelo com modelos presentes no mercado.

7 RESULTADOS

Nesta seção serão apresentados os resultados de cada um dos oito modelos treinados. Cada modelo, está apresentado e brevemente discutido em uma subseção. Os resultados comparados de cada modelo estão em uma nona seção.

7.1 Resultados dos Treinamento

Os resultados de treinamento de cada modelo e o *benchmark* das métricas de cada base utilizada para treino são apresentados a seguir.

Cada modelo é identificado segundo sua metodologia, divisão da base de dados e pré-treino, tal como está descrito na Tabela 2.

Dessa maneira, quanto a metodologia os nomes 'Metodologia IBM' e 'BERT Puro' se referem, respectivamente, às metodologias propostas no artigo e à metodologia proposta neste projeto.

Os termos 'IBM' e 'Aleatório' se referem à divisão entre treino e teste proposta na base de dados e a uma divisão aleatória entre treino e teste.

Já o termo 'Base' se refere a *bert base uncased*, enquanto o temo 'Large' se refere a *bert large uncased*.

A apresentação dos resultados foi feita em uma subseção para cada treinamento, que contém uma tabela com os resultados e discussão do modelo frente ao seu *benchmark*.

7.1.1 Metodologia IBM com Divisão Original da Base de Dados e Pré-treino *Base*

O resultado das métricas de avaliação do modelo treinado segundo a Metodologia IBM, a divisão entre treino e teste da base original e com pré-treino *bert base uncased* pode ser observado na tabela 7.1.1, bem como o *benchmark* dessa divisão.

O tempo de treinamento para esse modelo foi de 57 minutos e 39 segundos, obtido pela soma do tempo de treinamento dos modelos intermediários.

Métrica	**Metodologia IBM - IBM - *Base***	***Benchmark***
Acurácia	**0.5352**	**0.6457**
Precisão	**0.5982**	**0.7434**
Recall	**0.7040**	**0.6309**
Especificidade	**0.2797**	**0.6683**
F1-Score	**0.6468**	**0.6825**

Tabela 3: Resultado do Modelo Metodologia IBM - IBM - base.

Observa-se que esse modelo tem acurácia, precisão, especificidade e F1-Score menores que o *benchmark* dessa base de dados e que o recall é maior.

Quanto a comparação do par acurácia e precisão, observa-se que o modelo está tanto com uma variação maior de resultados quanto está com mais resultados errados do que o *benchmark*.

Considerando a precisão e o recall como métricas igualmente importantes para o caso analisado, e que o modelo superou o *benchmark* em recall, a análise da métrica F1-Score se torna melhor para comparação desse par. Nesse sentido, observa–se que a precisão do modelo é tão menor que não compensa o valor maior de recall desse, de maneira que o F1-Score fica também menor.

Um destaque negativo pode ser dado à especificidade do modelo, que ficou inferior em comparação às demais métricas, indicando que o modelo não é eficiente em identificar casos negativos, o que nesse caso representam argumentos contra o tópico.

Com essa informação, pode-se dizer que a a deficiência do modelo em predizer argumentos contrários faz com que ele tenha um desempenho pior que o *benchmark*.

Para entender melhor se o modelo pode ser aprimorado, é interessante observar o treinamento de cada um dos três modelos que compõem a Metodologia IBM: o sentimento da frase argumentativa, o sentimento do tópico e o sentimento da frase argumentativa em relação ao tópico.

O resultado de cada um desses modelos pode ser encontrado na Tabela 7.1.1, na qual o sentimento da frase argumentativa foi abreviado para 'SA', o sentimento do tópico para 'ST' e o sentimento da frase argumentativa em relação ao tópico para 'SA-T'.

Modelo	**Acurácia**	**Precisão**	**Recall**	**Especificidade**	**F1-Score**	***Train Loss***
SA	**0.5913**	**0.5051**	**0.7351**	**0.5988**	**0.7227**	***0.005***
ST	**0.5133**	**0.6479**	**0.5879**	**0.6164**	**0.3082**	***0.012***
SA-T	**0.5597**	**1.0000**	**0.5597**	**0.0000**	**0.0000**	***0.496***

Tabela 4: Resultados Parciais do Modelo Metodologia IBM - IBM - base.

Nessa tabela, pode-se observar que a análise de sentimento da frase argumentativa teve um desempenho melhor que a análise de sentimento do tópico, o que é considerado um resultado surpreendente, por demonstrar que a maneira como o tópico é escrito influencia diretamente este.

Quanto ao sentimento da frase argumentativa em relação ao tópico, observa-se que o modelo tem a mesma precisão, acurácia e valores de 1 para recall e 0 para especificidade.

Isso indica que o modelo não foi treinado corretamente. De fato, ao se observar as respostas desse, percebe-se que o modelo retornou apenas respostas positivas.

Um erro nessa escala indica que a divisão entre treino e teste original da base dados não é balanceada segundo o sentimento da frase argumentativa em relação ao tópico. O que é confirmado com o conhecimento sobre o não balanceamento dessa base de dado.

Isso faz com que o modelo tenha muito mais dados positivos em relação a negativos, o que o leva a ser um modelo viciado.

Quanto a análise do *Train Loss* dos modelos, observa-se que esse é baixo para as análises de sentimento da frase argumentativa e tópico, indicando que mais épocas não melhorariam o modelo.

Para a análise de sentimento da frase argumentativa em relação ao tópico, observa-se que esse é um valor alto, o que indica que o modelo poderia melhorar com mais épocas. Contudo, como observa-se um vício no modelo, pode-se concluir que mesmo com mais épocas esse não melhoraria.

Dessa maneira, esse modelo não supera nas métricas de avaliação estabelecidas o *benchmark*. Uma quantidade maior de relações negativas entre frases na base de dados para treino e de teste poderiam ajudar esse modelo a ser melhor treinado quanto a análise de sentimento da frase argumentativa em relação ao tópico.

7.1.2 Metodologia IBM com Divisão Aleatória e Pré-treino *Base*

Apresenta-se a seguir o resultado da avaliação do modelo treinado segundo a Metodologia IBM, a divisão entre treino, teste aleatória e com pré-treino *bert base uncased*. O tempo de treinamento desse caso, esse foi de 70 minutos e 26 segundos.

Observa-se que esse modelo tem acurácia, precisão, recall e F1-Score menores que o *benchmark* dessa base de dados e a especificidade é maior.

Métrica	**Metodologia IBM - Random - *Base***	***Benchmark***
Acurácia	**0.5221**	**0.5796**
Precisão	**0.6094**	**0.6387**
Recall	**0.4071**	**0.5731**
Especificidade	**0.6638**	**0.5879**
F1-Score	**0.4881**	**0.6041**

Tabela 5: Resultado do Modelo Metodologia IBM - Aleatório - base.

Quanto a comparação do par acurácia e precisão, o modelo apresenta um comportamento semelhante com o modelo anterior.

Já quanto ao F1-Score, observa-se que tanto a precisão do modelo quanto o recall desse são menores que o *benchmark*, de maneira que o F1-Score fica também menor.

Para analisar melhor a influência de cada etapa parcial no modelo, o resultado dessas podem ser encontrados na Tabela 7.1.4. Nessa, o sentimento da frase argumentativa foi abreviado para 'SA', o sentimento do tópico para ST e o sentimento da frase argumentativa em relação ao tópico para 'SA-T'.

Modelo	**Acurácia**	**Precisão**	**Recall**	**Especificidade**	**F1-Score**	***Train Loss***
SA	**0.5331**	**0.5614**	**0.7588**	**0.2462**	**0.5521**	***0.056***
ST	**0.4867**	**0.5396**	**0.5652**	**0.3869**	**0.5598**	***0.000***
SA-T	**0.5420**	**0.5598**	**0.8489**	**0.1507**	**0.6747**	***0.362***

Tabela 6: Resultados Parciais do Modelo Metodologia IBM - Aleatório - base.

Nessa tabela observa-se que a análise de sentimento da frase argumentativa e a análise de sentimento da frase argumentativa em relação ao tópico tiveram um desempenho melhor que a análise de sentimento do tópico.

Ainda, a especificidade dos modelos individuais é destacadamente baixa nos três modelos, estando abaixo dos 0.35 em todos eles, enquanto no Modelo IBM é de 0.6638. Isso indica que o Modelo IBM é capaz de minimizar efeitos negativos individuais dos modelos parciais.

Quanto à análise do *Train Loss* dos modelos, este se apresentou baixo para as análises de sentimento da frase argumentativa e tópico, indicando que mais épocas não aprimorariam o modelo.

Quanto a análise de sentimento da frase argumentativa em relação ao tópico, observa-se que esse é um valor alto, o que indica que o modelo poderia melhorar com mais épocas.

Dessa maneira, esse modelo não supera nas métricas de avaliação estabelecidas o *benchmark*. Um treinamento com mais épocas do modelo parcial de classificação de sentimento de frases argumentativas em relação ao tópico poderia ser testado para melhorar o desempenho desse modelo, embora isso não seja verdade para os outros modelos parciais.

7.1.3 Metodologia IBM com Divisão do Artigo da IBM e Pré-treino *Large*

A seguir, tem-se o resultado da avaliação do modelo treinado segundo a Metodologia IBM, a divisão entre treino e teste do artigo da IBM e com pré-treino *bert large uncased*. O tempo de treinamento desse caso, esse foi de 173 minutos e 10 segundos.

Observa-se que esse modelo tem acurácia, precisão e recall menores que o *benchmark* dessa base de dados e a especificidade e F1-Score é maior.

Dessa maneira, estabelece-se uma relação de escolha entre um modelo que tenha uma acurácia maior e um que tenha um F1-Score a depender do objetivo do modelo, o que só poderá ser avaliado em uma aplicação prática. De todo modo, o modelo proposto não superou totalmente o *benchmark*.

Métrica	**Nome do Modelo**	**Benchmark**
Acurácia	**0.6396**	**0.6457**
Precisão	**0.4914**	**0.7434**
Recall	**0.4914**	**0.6309**
Especificidade	**0.8652**	**0.6683**
F1-Score	**0.6221**	**0.6041**

Tabela 7: Resultado do Modelo Metodologia IBM - IBM - large.

Para analisar melhor a influência de cada etapa parcial nesse modelo, o resultado dessas podem ser encontrados na Tabela 7.1.3. Nessa, o sentimento da frase argumentativa foi abreviado para 'SA', o sentimento do tópico para ST e o sentimento da frase argumentativa em relação ao tópico para 'SA-T'.

Modelo	**Acurácia**	**Precisão**	**Recall**	**Especificidade**	**F1-Score**	***Train Loss***
SA	**0.6262**	**0.7604**	**0.5561**	**0.7331**	**0.6424**	*0.000*
ST	**0.5174**	**0.5766**	**0.7551**	**0.1554**	**0.6539**	*0.094*
SA-T	**0.5564**	**0.5935**	**0.8418**	**0.1217**	**0.6962**	*0.273*

Tabela 8: Resultados Parciais do Modelo Metodologia IBM - IBM - large.

Nessa, observa-se que os três treinamentos se deram de forma bem distribuída, com resultados parecidos em cada modelo. Há destaque para o F1-Score que é alto para 'ST' e 'AS-T', que se reflete no resultado final do modelo.

Contudo, esse valor não é obtido por conta de uma precisão alta, mas sim por conta de um recall alto. Isso leva com que a escolha entre esse modelo e o *benchmark* tenda para esse segundo.

7.1.4 Metodologia BERT Puro com Divisão Aleatória e Pré-treino *Large*

A seguir, tem-se o resultado da avaliação do modelo treinado segundo a Metodologia IBM, a divisão entre treino, teste aleatória e com pré-treino *bert large uncased*. O tempo de treinamento desse caso, esse foi de 246 minutos e 13 segundos. Também apresenta-se a tabela com o resultado dos treinamentos parciais dos modelos.

Métrica	**Nome do Modelo**	**Benchmark**
Acurácia	**0.3963**	**0.6457**
Precisão	**0.0000**	**0.7434**
Recall	**0.0000**	**0.6369**
Especificidade	**1.0000**	**0.6683**
F1-Score	**-**	**0.6825**

Tabela 9: Resultado do Modelo Metodologia IBM - Aleatório - large.

Modelo	Acurácia	Precisão	Recall	Especificidade	F1-Score	*Train Loss*
SA	**0.6036**	**0.6036**	**1**	**0**	**0.7528**	***0.676***
ST	**0.3963**	**0.0000**	**0.0000**	**1.0000**	**-**	***0.094***
SA-T	**0.6037**	**0.6036**	**1.0000**	**0.0000**	**0.7528**	***0.000***

Tabela 10: Resultados Parciais do Modelo Metodologia IBM - Aleatório - large.

Dessa maneira, observa-se que esse modelo não deu certo em seu treinamento, nem no resultado final nem em nenhum dos treinamentos parciais. Isso pode ter se devido a uma divisão ruim dos dados feita por conta da divisão aleatória dos dados.

7.1.5 Metodologia BERT Puro com Divisão Original da Base de Dados e Pré-treino *Base*

O resultado das métricas de avaliação do modelo treinado segundo a Metodologia BERT Puro, a divisão entre treino e teste do artigo da IBM e com pré-treino *bert base uncased* podem ser observado na Tabela 7.1.5, bem como o *benchmark* desta divisão. O tempo de treinamento para esse modelo foi de 18 minutos e 19 segundos.

Percebe-se que esse modelo tem acurácia, precisão, recall, especificidade e F1-Score melhores que o *benchmark* dessa base de dados.

Métrica	Metodologia BERT Puro - IBM - *Base*	*Benchmark*
Acurácia	**0.8677**	**0.6457**
Precisão	**0.9335**	**0.7434**
Recall	**0.8199**	**0.6309**
Especificidade	**0.8865**	**0.6683**
F1-Score	**0.8730**	**0.6825**

Tabela 11: Resultado do Modelo BERT Puro - IBM - base.

Quanto a comparação do par acurácia e precisão, o modelo apresenta valores altos e próximos, o que o um modelo que tanto prevê com consistência quanto corretamente, uma característica positiva.

Quanto ao F1-Score, observa-se que tanto a precisão do modelo quanto o recall desse são maiores que o *benchmark*, de maneira que o F1-Score fica também maior.

Além desses pontos, o *Train Loss* do modelo é de 0.56, o que indica que, somado com

dados positivos anteriores, que se esse modelo fosse treinado com mais épocas, teria um desempenho melhor.

Dessa maneira, o modelo apresentado supera o *benchmark*. Um ponto de melhoria seria testar seu treinamento para mais de quatro épocas, uma vez que o bom desempenho dele somado a um *Train Loss* alto, indica que o modelo ainda pode ser aprimorado.

7.1.6 Metodologia BERT Puro com Divisão Aleatória e Pré-treino *Base*

O resultado das métricas de avaliação do modelo treinado segundo a Metodologia BERT Puro, a divisão entre treino e teste aleatória e com pré-treino *bert base uncased* podem ser observado na tabela 7.1.6, bem como o *benchmark* desta divisão. O tempo de treinamento para esse modelo foi de 28 minutos e 33 segundos.

Percebe-se que esse modelo tem acurácia, precisão, recall, especificidade e F1-Score melhores que o *benchmark* dessa base de dados.

Métrica	**Metodologia BERT Puro - IBM - *Base***	***Benchmark***
Acurácia	**0.7608**	**0.6457**
Precisão	**0.8423**	**0.6681**
Recall	**0.8087**	**0.5856**
Especificidade	**0.8109**	**0.6368**
F1-Score	**0.8251**	**0.6241**

Tabela 12: Resultado do Modelo BERT Puro - Aleatório - base.

Além desses pontos, o *Train Loss* do modelo é de 0.2139, o que indica que se esse modelo fosse treinado com mais épocas, teria um desempenho melhor.

Dessa maneira, o modelo apresentado supera o *benchmark*. Um ponto de melhoria seria testar o treinamento desse para mais de quatro épocas, uma vez que o bom desempenho dele somado a um *Train Loss* alto, indica que o modelo ainda pode ser aprimorado.

7.1.7 Metodologia BERT Puro com Divisão Original da Base de Dados e Pré-treino *Large*

O resultado das métricas de avaliação do modelo treinado segundo a Metodologia BERT Puro, a divisão entre treino e teste do artigo da IBM e com pré-treino *bert large*

uncased podem ser observado na tabela 7.1.7, bem como o *benchmark* desta divisão. O tempo de treinamento para esse modelo foi de 97 minutos e 23 segundos.

Percebe-se que esse modelo tem acurácia, precisão, recall, especificidade e F1-Score melhores que o *benchmark* dessa base de dados.

Métrica	**Metodologia BERT Puro - IBM - *Base***	***benchmark***
Acurácia	**0.9568**	**0.6457**
Precisão	**0.9475**	**0.7434**
Recall	**0.9829**	**0.6309**
Especificidade	**0.8865**	**0.6683**
F1-Score	**0.9648**	**0.6825**

Tabela 13: Resultado do Modelo BERT Puro - IBM - large.

Além desses pontos, o *Train Loss* do modelo é de 0.6892, o que indica que se esse modelo fosse treinado com mais épocas, teria um desempenho melhor.

Dessa maneira, o modelo apresentado supera o benchmark. Um ponto de melhoria seria testar o treinamento desse para mais de quatro épocas, uma vez que o bom desempenho dele somado a um *Train Loss* alto, indica que o modelo ainda pode ser aprimorado.

7.1.8 Metodologia BERT Puro com Divisão Aleatória e Pré-treino *Large*

O resultado das métricas de avaliação do modelo treinado segundo a Metodologia BERT Puro, a divisão entre treino e teste aleatória e com pré-treino *bert large uncased* podem ser observado na tabela 7.1.8, bem como o *benchmark* desta divisão. O tempo de treinamento para esse modelo foi de 95 minutos e 35 segundos.

Percebe-se que esse modelo tem acurácia, precisão, recall, especificidade e F1-Score melhores que o *benchmark* dessa base de dados.

Métrica	**Metodologia BERT Puro - IBM - *Base***	***benchmark***
Acurácia	**0.8097**	**0.6084**
Precisão	**0.8423**	**0.6681**
Recall	**0.8087**	**0.5856**
Especificidade	**0.8109**	**0.6368**
F1-Score	**0.8251**	**0.6241**

Tabela 14: Resultado do Modelo BERT Puro - Aleatório - large.

Além desses pontos, o *Train Loss* do modelo é de 0.3184, o que indica que se esse modelo fosse treinado com mais épocas, teria um desempenho melhor.

Dessa maneira, o modelo apresentado supera o benchmark. Um ponto de melhoria seria testar o treinamento desse para mais de quatro épocas, uma vez que o bom desempenho dele somado a um *Train Loss* alto, indica que o modelo ainda pode ser aprimorado.

7.2 Comparação do Resultado Entre Modelos

Uma maneira mais organizada de se comparar o desempenho de cada modelo em relação às métricas avaliadas pode ser vista na Tabela 15.

Nessa tabela, 'PB' representa 'BERT Puro', 'IBM' na coluna 'Mét.' representa 'Metodologia IBM' e na coluna 'Split' representa a divisão original. 'base' representa bert-base-uncased e 'large' representa bert-large-uncased. Por fim, 'rand' representa a divisão aleatória.

Mét.	**Split**	**Pre-Treino**	**Accuracia**	**Recall**	**Precisão**	**F1-Score**	**Especificidade**
PB	**IBM**	**base**	**0.8677**	**0.8199**	**0.9335**	**0.8730**	**0.8865**
PB	**IBM**	**large**	**0.9568**	**0.9829**	**0.9475**	**0.9648**	**0.9170**
PB	**Rand**	**base**	**0.7608**	**0.8245**	**0.7608**	**0.7913**	**0.6727**
PB	**Rand**	**large**	**0.8097**	**0.8087**	**0.8423**	**0.8251**	**0.8109**
IBM	**IBM**	**base**	**0.5359**	**0.7040**	**0.5982**	**0.6468**	**0.2797**
IBM	**IBM**	**large**	**0.6396**	**0.4914**	**0.8475**	**0.6221**	**0.8652**
IBM	**Rand**	**base**	**0.5221**	**0.4071**	**0.6094**	**0.4881**	**0.6638**
IBM	**Rand**	**large**	**0.3963**	**0**	**0**	**0**	**1**

Tabela 15: Resultado de todos os modelos treinados.

Analisando os resultados da etapa anterior, fica clara uma divisão entre os modelos Metodologia IBM e os modelos BERT Puro. Os modelos BERT Puro tiveram um desempenho significativamente superior aos modelos da Metodologia IBM em todas as métricas avaliadas.

Na comparação, a diferença entre os dois modelos testados chega a ser o dobro entre o melhor modelo BERT Puro e o pior modelo Metodologia IBM para a métrica F1-Score, sendo que mesmo para o melhor modelo dessa metodologia a diferença continua alta.

Além disso, todos os quatro modelos BERT Puro tiveram desempenho superior ao *benchmark* estabelecido para cada uma das divisões de dados de teste feitas.

Enquanto as métricas retornadas pela API da IBM tiveram valores em torno de 0.6 e 0.65 para todas as métricas, grande parte dos valores dos modelos BERT Puro estão com valores acima de 0.75, chegando a 0.98 para o recall do modelo BERT Puro treinado com a base da IBM e com o maior pré-treino.

Para além do desempenho dos dois tipos de modelo, pode-se observar também que para os modelos BERT Puro houve um desempenho melhor dos modelos treinados e testados segundo a divisão original da base de dados.

O comportamento observado é diferente do observado para o Modelo IBM, onde não há uma melhora clara dependendo da base de dados, mas sim um balanço entre uma especificidade ou um recall menor.

No quesito de pré-treino, os modelos treinados com *bert large uncased* tiveram desempenho superior aos modelos treinados com *bert base uncased*. Esse comportamento era esperado, uma vez que o léxico e processamento do *bert large uncased* são maiores.

Essa diferença foi significativa para os modelos BERT Puro, representando uma melhora em todos as métricas avaliadas menos para o recall dos BERT Puro treinados com divisão aleatória dos dados, ainda que a diferença não tenha sido grande.

Para as métricas acurácia e F1-score, houve um aumento de 0.04 e 0.03 para os modelos treinados com divisão aleatória dos dados e de 0.09 e 0.09 para os modelos com a divisão original da base de dados.

Frente a essas avaliações, observa-se que um modelo baseado em Transformer como o BERT que tenha como entradas um tópico e uma frase argumentativa, classificadas entre argumentos contra ou a favor do tópico, supera a API da IBM que executa a mesma tarefa e que foi estabelecida com Benchmark.

Contudo, observa-se também que a Metodologia IBM aplicada com Transformer não supera o resultado da API da IBM.

Dentro disso, observa-se ainda que um modelo treinado com a separação de dados proposta originalmente para a base de dados utilizada e que seja pré-treinado com o *bert large uncased* terá o melhor desempenho dentre as alternativas testadas.

Dessa forma, o melhor modelo treinado foi aquele treinado com 4 épocas, segundo a metodologia BERT Puro, com a divisão entre base de treino e teste original da base de daods utilizada e utilizando o *bert large uncased*.

8 CONCLUSÕES

Neste TCC propõe-se um novo método para classificar a relação de suporte ou ataque de uma frase argumentativa associada a um determinado tópico.

Esse novo método proposto foi a utilização da arquitetura Transformer, aplicada por meio do modelo BERT, uma vez que não se encontrou na literatura uma aplicação dessa arquitetura e modelo para o caso proposto.

O modelo foi utilizado tanto com uma entrada direta da frase argumentativa e seu tópico quanto pela Metodologia IBM, que propõe a análise de sentimento da frase arguemntativa, do tópico e da frase argumentativa em relação ao tópico.

Para se estabelecer um *benchmark*, utilizou-se a API do projeto IBMD, executando a tarefa de classificação de argumentos entre contra ou a favor de um tópico que apresentam as mesmas entradas que o presente TCC.

Dessa maneira, tanto a BERT Puro quanto a Metodologia IBM foram comparadas entre si e com o *benchmark* estabelecido para cada treinamento quanto a acurácia, precisão, recall, F1-Score e especificidade Foi registrado também o tempo de treino e as características da máquina que rodou esses treinos.

Além do teste principal de diferenças de metodologia, testou-se também utilizar dois pré-treinos diferentes, os *bert base uncased* e *bert large uncased*, para avaliar como a diferença de tamanho de lexos e cabeças de treinamento afetam os desempenhos dos modelos.

Esses testes foram feitos com a divisão entre treino e teste original da base de dados e com uma divisão aleatórias. Dessa maneira, totalizaram 8 modelos treinados.

Desses modelos, todos aqueles treinados com BERT Puro tiveram desempenho melhor que o *benchmark* e aqueles treinados com a Metodologia IBM foram piores que o *benchmark*.

O desempenho da Metodologia IBM pode ser em parte explicado pelo treinamento ruim observado do sentimento da frase argumentativa em relação ao tópico. Esse desempenho

ruim foi avaliado como sendo resultado do fato da base de dados não estar balanceada nesse rótulo.

Ainda assim, para o caso de metodologia IBM treinada com a divisão original da base de dados e *bert large uncased*, os resultados do modelo exigem que se faça uma avaliação caso a caso, uma vez que teve acurácia pior mas F1-Score melhor que o *benchmark*.

Dentre os modelos BERT Puro, aquele que obteve o melhor desempenho foi o treinado com a divisão original entre treino e teste da base de dados e com o pré-treino *bert large uncased*, tendo esse superado por larga margem o *benchmark*.

Dessa maneira, o presente TCC cumpriu seu objetivo de propor e testar um novo método para classificar a relação de suporte ou ataque de uma frase argumentativa associada a um determinado tópico.

Além disso, demonstrou-se que a arquitetura Transformers aplicada com o modelo BERT é capaz de ser significativamente melhor que o desempenho da ferramenta comercial de classificação de argumentos do IBMD dentro das métricas avaliadas.

9 TRABALHOS FUTUROS

Espera-se que esse projeto tenha contribuído com os projetos desenvolvidos no C4AI dentro da linha de argumentação computacional. Para melhorar e expandir as possibilidades desse projeto, uma avaliação de novos modelos baseado em Transformers como o RoBERTa [37] é interessante, inclusive porque esse modelo tem reconhecidamente desempenho melhor que o BERT.

Além disso, poderão ser exploradas novas arquiteturas diferentes da Transformers, como as GPT e T5, para avaliar a classificação de argumentos nessas condições.

Por fim, para a atuação específica do C4AI, como a argumentação computacional sendo desenvolvida no laboratório tem o campo específico de AA, pode ser proveitoso dar mais destaque nos pré-treinos para palavras envolvendo esse tema, inclusive tendo a possibilidade de se criar um pré-treino com esse campo específico.

REFERÊNCIAS

1 BAR-HAIM, R. et al. Stance classification of context-dependent claims. In: *In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. [S.l.]: Association for Computational Linguistics.

2 United Nations Office of Legal Affairs. *The Second World Ocean Assessment - Volume 1*. Uninated Nations, 2021. 5–7 p. Disponível em: <https://doi.org/10.18356/9789216040062>.

3 General Assembly Resolution. *Transforming our wolrd: the 2030 Agenda for Sustainable Development, A/RES/70/1*. 21 October 2015. Disponível em: <https://www.un.org/ga/search/view_doc.asp?symbol=A/RES/70/1&Lang=E>.

4 United Nations Office of Legal Affairs. *The Second World Ocean Assessment - Volume 2*. Uninated Nations, 2021. 355–356 p. Disponível em: <https://doi.org/10.18356/9789216040062>.

5 Wikipedia. *Zona econômica exclusiva do Brasil*. Disponível em: <https://pt.wikipedia.org/wiki/Zona_econÃťmica_exclusiva_do_Brasil>. Acesso em: 28/11/2022.

6 Roberto de Guimarães Carvalho. *A outra Amazônia*. 2004. Disponível em: <https://www1.folha.uol.com.br/fsp/opiniao/fz2502200409.htm>. Acesso em: 28/11/2022.

7 LIPPI, M.; TORRONI, P. Argumentation mining: State of the art and emerging trends. *ACM Transactions on Internet Technology*, 2016. Disponível em: <http://dx.doi.org/10.1145/2850417>.

8 LYTOS, A. et al. The evolution of argumentation mining: From models to social media and emerging tools. *Information Processing and Management*, v. 56, 2019. ISSN 03064573.

9 SHIELDS, C. Aristotle. *The Stanford Encyclopedia of Philosophy*, abs/1810.04805, 2022.

10 WHATELY, R. *Elements of Logic*. [S.l.]: Harper Brothers, 1857.

11 TOULMIN, S. E. *The Uses of Argument - Updated Edition*. 2. ed. [S.l.]: Cambrige, 2003. 87–131 p.

12 HABERNAL, I.; ECKLE-KOHLER, J.; GUREVYCH, I. Argumentation mining on the web from information seeking perspective. *CEUR Workshop Proceedings*, v. 1341, 2014.

13 NOBLE, D.; RITTEL, H. W. Issue-based information systems for design.

ACADIA Conference Proceedings '88, p. 275–286, 1988. Disponível em: <https://doi.org/10.52842/conf.acadia.1988.275>.

14 NOBLE, D.; RITTEL, H. W. Defeasible reasoning. *Cognitive Science*, v. 11, p. 481–518, 1987. Disponível em: <https://doi.org/10.1016/S0364-0213(87)80017-4>.

15 DUNG, P. M. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, v. 77, p. 321–357, 1995. Disponível em: <https://doi.org/10.1016/0004-3702(94)00041-X.>

16 KRAUSE, P. et al. A logic of argumentation for reasoning under uncertainty. *Computational Intelligence*, v. 11, p. 113–131, 1995. Disponível em: <https://doi.org/10.1111/j.1467-8640.1995.tb00025.x>.

17 BENTAHAR, J.; MOULIN, B.; B´ELANGER, M. A taxonomy of argumentation models used for knowledge representation. *Artificial Intelligence Review*, v. 33, p. 211–259, 2010. Disponível em: <https://doi.org/10.1007/s10462-010-9154-1>.

18 POLLOCK, J. L. Defeasible reasoning with variable degrees of justification. *Artificial Intelligence*, v. 133, p. 233–282, 1995. Disponível em: <https://doi.org/10.1016/S0004-3702(01)00145-X>.

19 BILU, Y. et al. Argument invention from first principles. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, p. 1013–1026, 2019.

20 WALTON, D.; GORDAN, T. The carneades model of argument invention. *Pragmatics Cognition*, v. 20, 2012. Disponível em: <https://doi.org/10.1075/pc.20.1.01wal>.

21 CABRIO, E.; VILLATA, S. Context dependent claim detection. In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. [s.n.], 2018. p. 5427–5433. Disponível em: <https://doi.org/10.24963/ijcai.2018/766>.

22 HABERNAL, I.; ECKLE-KOHLER, J.; GUREVYCH, I. Argumentation mining on the web from information seeking perspective. *CEUR Workshop Proceedings*, v. 1341, 2014.

23 LEVY, R. et al. Context dependent claim detection. In: *COLING*. [S.l.: s.n.], 2014. p. 1489–1500.

24 MOCHALES, R.; MOENS, M.-F. Argumentation mining. *Artificial Intelligence and Law*, v. 19, p. 1–22, 03 2011.

25 BAR-HAIM LILACH EDELSTEIN, C. J. R.; SLONIM., N. Improving claim stance classification with lexical knowledge expansion and context utilization. In: *In Proceedings of the 4th Workshop on Argument Mining*. [S.l.]: Association for Computational Linguistics.

26 AHARONI, E. et al. A benchmark dataset for automatic detection of claims and evidence in the context of controversial topics. In: *Proceedings of the First Workshop on Argumentation Mining*. Association for Computational Linguistics, 2014. p. 64–68. Disponível em: <https://aclanthology.org/W14-2109>.

27 SONNTAG, J.; STEDE, M. GraPAT: a tool for graph annotations. In: *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. European Language Resources Association (ELRA), 2014. p. 4147–4151. Disponível em: <http://www.lrec-conf.org/proceedings/lrec2014/pdf/824_Paper.pdf>.

28 SLONIM, N. et al. An autonomous debating system. *Nature*, v. 591, p. 379–384, 2021. Disponível em: <https://doi.org/10.1038/s41586-021-03215-w>.

29 BAR-HAIM, R. et al. Project debater apis: Decomposing the ai grand challenge. *IBM Research*, 2021. Disponível em: <https://doi.org/10.48550/arXiv.2110.01029>.

30 TOLEDO-RONEN, O. et al. Multilingual argument mining: Datasets and analysis. In: *FINDINGS*. [S.l.: s.n.], 2020.

31 VASWANI, A. et al. Attention is all you need. In: GUYON, I. et al. (Ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017. v. 30, p. 5998–6008. Disponível em: <https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf>.

32 Nicolo Cosimo Albanese. *Fine-Tuning BERT for Text Classification - A step-by-step tutorial in Python*. 2022. Disponível em: <https://towardsdatascience.com/fine-tuning-bert-for-text-classification-54e7df642894#12df>. Acesso em: 15/05/2022.

33 WOLF, T. et al. Transformers: State-of-the-art natural language processing. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, 2020. p. 38–45. Disponível em: <https://www.aclweb.org/anthology/2020.emnlp-demos.6>.

34 DEVLIN, J. et al. BERT: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805, 2019.

35 WU, Y. et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *ArXiv*, abs/1609.08144, 2016.

36 AHARONI ANATOLY POLNAROV, T. L. D. H. R. L. R. R. D. G. E.; SLONIM, N. Ta benchmark dataset for automatic detection of claims and evidence in the context of controversial topics. In: *In Proceedings of the First Workshop on Argumentation Mining*. [S.l.]: Association for Computational Linguistics, 2014. p. 64—68.

37 LIU, Y. et al. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. Disponível em: <http://arxiv.org/abs/1907.11692>.