UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

GABRIEL PAKULSKI DA SILVA

Comparativo de Técnicas de Geração de Linguagem Natural Para a Tarefa de *Long-Form Question Answering*

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof. Dr. Dante Augusto Barone
Co-orientador: Prof. Ms. Eduardo Gabriel Cortes

Porto Alegre
2022

“A man is but the product of his thoughts.
What he thinks, he becomes.”
— Mahatma Gandhi

AGRADECIMENTOS

Agradeço aos meus pais, Maria Aparecida Pakulski e Ronaldo Santos da Silva por todo apoio prestado, possibilitando a mim o privilégio de estudar numa universidade federal de excelência.

A minha companheira, Larissa de Lima por todo amor incondicional e apoio emocional que muito contribuíram para a realização deste trabalho.

Aos meus queridos amigos, que diante de todas as intempéries da vida sempre se mantiveram ao meu lado, promovendo apoio para que concluísse minha graduação.

Aos professores Dante Augusto Barone e Eduardo Gabriel Cortes, por terem sido meus orientadores e desempenhado tal função com empatia e amizade.

Aos professores do Instituto de Informática da UFRGS, por seu empenho na educação pública e pelos ensinamentos que me permitiram apresentar um melhor desempenho no meu processo de formação profissional ao longo do curso.

Agradeço, por fim, a todos aqueles que contribuíram, de alguma forma, para a realização deste trabalho.

RESUMO

A área de processamento de linguagem natural vem ganhando destaque principalmente pelos avanços em técnicas baseadas em aprendizado de máquina e *Transformers*, que conseguem lidar facilmente com dependências textuais de longo alcance. Entretanto, a área ainda apresenta diversos desafios relacionados à compreensão e geração de linguagem natural. No contexto de *Long-Form Question Answering* (LFQA) temos o desafio em que as perguntas exigem respostas não factuais e complexas, sendo necessário recorrer a outros recursos para que sistemas autônomos de QA sejam capazes de providenciar uma resposta precisa e completa. Na tarefa de geração de linguagem natural, existem diversos avanços, principalmente com modelos de linguagem autorregressiva que usam aprendizagem profunda para produzir texto semelhante ao humano. O presente estudo busca aplicar modelos de geração de linguagem natural para a tarefa de LFQA, apoiando-se em extensas bases de dados de perguntas e respostas não-factuais. Experimentos empíricos comparando algumas das abordagens consideradas estado da arte para a tarefa foram realizados. Para realizar o comparativo, este trabalho faz uso do conjunto de dados ELI5 e simultaneamente criando documentos de suporte através de trechos recuperados de artigos retirados da Wikipedia. Reportamos as métricas de performance dos modelos BART e T5-Small para a tarefa de LFQA. Concluímos que a exploração de modelos mais robustos do tipo T5 pode levar a resultados mais promissores que o atual estado da arte para a base de dados citada. Os resultados dessa pesquisa irão contribuir para outros projetos relacionados à tarefa de LFQA.

Palavras-chave: Transformers, processamento de linguagem natural, NLP, LFQA, geração de linguagem natural, NLG, ELI5, BART, T5-Small.

Comparison of Natural Language Generation Techniques for the LFQA Task

ABSTRACT

The field of Natural Language Processing has been gaining prominence mainly due to advances in techniques based on machine learning and Transformers, which can easily manage long-range text dependencies. However, the field still presents several challenges related to the understanding and generation of natural language. In the context of Long-Form Question Answering (LFQA) we have the challenge where the questions require non-factoid and complex answers, requiring other resources so that autonomous QA systems are able to provide an accurate and complete answer. In the natural language generation task there has been several advances, particularly with auto-regressive language models that use deep learning to produce human-like text. This study seeks to apply natural language generation models to the LFQA task, relying on extensive non-factoid LFQA data sets. Empirical experiments comparing some of the state-of-the-art approaches to the task were held. To compare the models, this work makes use of the ELI5 data set, simultaneously creating support documents through snippets gathered from Wikipedia articles. We report the performance metrics of the BART and T5-Small models for the LFQA task. We conclude that the use of more robust models of the T5 family may lead to more promising results than the current state-of-the-art for the aforementioned dataset. The results of this research will contribute to other projects related to the LFQA task.

Keywords: Transformers, natural language processing, NLP, LFQA, natural language generation, NLG, ELI5, BART, T5-Small.

LISTA DE FIGURAS

Figura 2.1 Busca realizada na plataforma Google, evidenciando o resultado obtido 15
Figura 2.2 Arquitetura de um Sistema de *Question Answering* 16
Figura 2.3 Arquitetura de um Modelo baseado em *Transformers*. 20

Figura 3.1 Etapas da pesquisa 26
Figura 3.2 Visão geral de execução para *Fine-tuning* 27
Figura 3.3 Disposição da frequência das palavras iniciais das perguntas. O tamanho da caixa indica frequência. 28
Figura 3.4 Exemplo de post no subreddit *ELI5* e a resposta mais votada 29
Figura 3.5 Exemplo de registro da base de dados *wiki_snippets* 31
Figura 3.6 Exemplo de inserção de ruído na entrada para o modelo BART 32
Figura 3.7 Funcionamento do modelo T5, todas as tarefas são convertidas em processar texto e gerar texto, inclusive problemas de classificação. 33

LISTA DE TABELAS

Tabela 1.1 Perguntas Não Factuais 12
Tabela 1.2 Perguntas Factuais 12

Tabela 2.1 Tabela comparativa de diferentes técnicas no *benchmark* ANTIQUE 23
Tabela 2.2 Tabela comparativa de diferentes modelos na base de dados MASH-QA 23
Tabela 2.3 Tabela comparativa de diferentes técnicas no *benchmark* WikiPassageQA.. 24
Tabela 2.4 Comparação de Trabalhos Relacionados 25

Tabela 3.1 Distribuição das questões na base de dados ELI5 30

Tabela 4.1 Avaliação quantitativa da performance do modelo BART 40
Tabela 4.2 Avaliação quantitativa da performance do modelo T5-Small 40
Tabela 4.3 Estado da arte para o *dataset* ELI5 41
Tabela 4.4 Comparativo do estado da arte com resultados desta pesquisa 41
Tabela 4.5 Tempos de treinamento dos modelos BART e T5-Small 42

LISTA DE ABREVIATURAS E SIGLAS

NLP	Natural Language Processing
IA	Inteligência Artificial
ML	Machine Learning
NLG	Natural Language Generation
QA	Question Answering
LFQA	Long-Form Question Answering
IR	Information Retrieval
RNN	Recurrent Neural Network
CNN	Convolutional Neural Network
ELI5	Explain Like I'm Five

SUMÁRIO

1 INTRODUÇÃO 11
2 REFERENCIAL TEÓRICO E TRABALHOS RELACIONADOS 14
2.1 *Question Answering* 14
2.1.1 QA: Como funciona 15
2.1.1.1 *Question Processing* 16
2.1.1.2 *Document Processing* 16
2.1.1.3 *Answer Processing* 17
2.1.2 QA Factual x Não Factual 17
2.1.2.1 Perguntas Factuais 17
2.1.2.2 Perguntas Não Factuais 18
2.1.3 *Long-form Question Answering* 19
2.2 *Transformers e Geração de Linguagem Natural* 19
2.3 Modelos Pré-treinados 21
2.4 Trabalhos Relacionados 22
2.4.1 ANTIQUE: *A Non-Factoid Question Answering Benchmark* 22
2.4.2 *Question Answering with Long Multiple-Span Answers* 23
2.4.3 WikiPassageQA: *A Benchmark Collection for Research on Non-factoid Answer Passage Retrieval* 24
2.4.4 Resumo dos Trabalhos Relacionados 24
3 METODOLOGIA 26
3.1 Visão Geral 26
3.2 ELI5: *Explain Like I'm Five* 28
3.3 *Support Documents* 30
3.3.1 *Wikipedia* 30
3.3.2 Criação dos *Support Documents* 31
3.4 BART 32
3.5 T5 33
3.6 Hugging Face ® 34
3.7 Métricas 35
3.8 Treinamento 36
3.8.1 BART 37
3.8.1.1 Detalhes da máquina 37
3.8.1.2 Hiperparâmetros 37
3.8.2 T5-Small 37
3.8.2.1 Detalhes da máquina 38
3.8.2.2 Hiperparâmetros 38
3.9 Avaliação dos modelos 38
3.9.1 Parâmetros de Geração de Respostas 38
4 RESULTADOS 40
4.1 Avaliação Quantitativa dos Modelos 40
4.1.1 Tempos de treinamento 41
4.2 Discussão 42
4.3 Limitações 43
5 CONCLUSÃO 44
REFERÊNCIAS 45
APÊNDICE A - EXEMPLO DE SUPPORT DOCUMENT 49
APÊNDICE B - RESPOSTAS GERADAS PELOS MODELOS 52
APÊNDICE C - AMBIENTE COMPUTACIONAL 53

1 INTRODUÇÃO

O Processamento de Linguagem Natural (do inglês *Natural Language Processing*, ou NLP) é uma das grandes áreas da Inteligência Artificial (IA). Ela contempla, majoritariamente, dois grandes problemas: geração e compreensão automática de línguas humanas naturais. Este trabalho, de certa forma, contempla esses dois campos, pois foram aplicados modelos de aprendizado de máquina (do inglês *Machine Learning*, ou ML) que necessitam compreender perguntas e documentos escritos por outros seres humanos para gerar respostas em uma linguagem compreensível pelos mesmos.

Recentemente, a área de NLP vem ganhando destaque em conferências, principalmente pelos avanços que fazem uso de técnicas baseadas em ML e *Transformers*. Tecnologias essas que conseguem lidar com dependências textuais de longo alcance com facilidade, em termos de custo computacional. Um dos grandes fatores para tanto é a ascensão do mercado de GPUs e TPUs, que possibilitou um enorme salto na questão custo-benefício para trabalhadores e pesquisadores da área (MADIAJAGAN; RAJ, 2019).

Além disso, a área ainda apresenta diversos desafios relacionados à compreensão e geração de linguagem natural (do inglês *Natural Language Generation*, ou NLG). Uma das suas sub-áreas mais famosas é a de *Question Answering* (QA), que engloba a construção de plataformas para responder (utilizando alguma forma de contexto) de forma automática questões feitas por humanos em sua linguagem natural (Calijorne Soares; PARREIRAS, 2020).

Um dos problemas mais desafiadores deste campo é o que chamamos de *Long-Form Question Answering* (LFQA), que consiste em fazer a máquina ser capaz de responder perguntas não factuais, ou seja, questionamentos que demandam respostas complexas, as quais necessitam - muitas vezes - de conhecimentos das mais variadas áreas de estudo. Exemplos podem ser encontrados na tabela 1.1

Por conta dessa dificuldade elevada de resolver o problema, a maioria das abordagens conta com a NLG, utilizando-se principalmente ou de técnicas de aprendizagem profunda ou de tecnologias derivadas das Redes Neurais, como modelos de linguagem autorregressiva, que resultam em texto semelhante àquele produzido por seres humanos.

Segundo McDonald (2010), NLG é o ato no qual o pensamento é processado em linguagem. Para a computação isso implica em tornar as máquinas capazes de transformar em linguagem humana conhecimentos de diferentes fontes, como: bases de dados, tabelas, relatórios médicos, *etc*.

Tabela 1.1: Perguntas Não Factuais

Índice da pergunta	**Conteúdo**	**Tradução**
Pergunta 1	*Why is it hard to breathe with a strong air gust blowing straight at your face?*	Por que é difícil respirar com uma forte rajada de vento soprando no seu rosto?
Pergunta 2	*Why do we use a different letter "a" when typing as opposed to when writing?*	Por que utilizamos uma letra "a" diferente para digitação e escrita?
Pergunta 3	*What's the difference between a bush, a shrub, and a tree?*	Qual a diferença entre um arbusto, uma moita e uma árvore?

Fonte: O Autor, retirado do *dataset* ELI5

Os avanços na área de LFQA são, de certa forma, recentes. A maioria das pesquisas que se têm nesse ramo concentram-se em QA extrativo, que atende melhor perguntas factuais, ou seja, aquelas onde a resposta consiste de um trecho curto, como um nome próprio ou uma data (exemplos podem ser vistos na tabela 1.2). Deste modo, a área de LFQA ainda possui muito potencial, o que torna-se muito atrativo para a pesquisa.

Tabela 1.2: Perguntas Factuais

Índice da pergunta	**Conteúdo**
Pergunta 1	Qual a cor do oceano?
Pergunta 2	Quantos continentes existem no planeta?
Pergunta 3	Qual a temperatura de ebulição da água?

Fonte: O Autor

Por este motivo, esta pesquisa busca aplicar modelos de ML, especializados em NLG, para a tarefa de LFQA, fazendo uso de extensos conjuntos de dados de perguntas e respostas não factuais, de modo a contribuir para futuros projetos na área. Este tipo de abordagem já é amplamente utilizada, como nos trabalhos: Krishna, Roy and Iyyer (2021), Bui et al. (2020) e Fan et al. (2019).

O objetivo desta pesquisa é fazer um comparativo aplicando modelos de NLG para a tarefa de LFQA. Serão realizados experimentos empíricos coletando métricas automatizadas através da aplicação de modelos pré-treinados considerados o estado da arte para esta tarefa.

Como um diferencial, este trabalho utiliza a *Wikipedia*[1] como base de dados para construir o documento passado aos modelos junto à pergunta (documento este em que se baseiam para construir uma saída), de modo a obter uma resposta satisfatória. O processo

[1]https://en.wikipedia.org/

de ilustração do funcionamento desses modelos será explorado nas próximas seções. Esta plataforma é um projeto de enciclopédia online de licença livre, escrito de maneira colaborativa. A mesma já foi utilizada em estudos semelhantes (LIU et al., 2020).

Algumas das contribuições presentes neste trabalho são: levantamento bibliográfico da área LFQA, bem como acerca de modelos baseados na arquitetura de *Transformers*; apontamento de trabalhos relacionados; e resultados dos experimentos realizados, indicando que performances melhores ainda podem ser obtidas explorando extensões de soluções propostas por este estudo.

Na próxima seção serão discutidas as bases teóricas da tarefa de QA, assim como as diferenças nas categorias da área, de modo a fornecer o referencial necessário para compreender a comparação e implementação dos modelos utilizados para esta pesquisa. Por fim, este trabalho possui a seguinte estrutura: o Capítulo 2 consiste em uma revisão de literatura para melhor apresentar os fundamentos da área, bem como alguns desafios enfrentados por pesquisadores do ramo, ao final do mesmo serão levantados trabalhos relacionados que, de certa forma, fundamentaram esta pesquisa; no Capítulo 3 será descrito o procedimento adotado por este estudo, e também serão detalhadas as tecnologias e recursos utilizados; o Capítulo 4 consiste da apresentação dos resultados obtidos, bem como uma discussão acerca dos mesmos; e, finalmente, no Capítulo 5 serão serão manifestadas as conclusões, e também os possíveis trabalhos futuros.

2 REFERENCIAL TEÓRICO E TRABALHOS RELACIONADOS

Conforme Kumar (2013), Linguagens Naturais (do inglês, *Natural Languages*) são linguagens que naturalmente foram evoluídas e utilizadas por seres humanos para comunicação (por exemplo: Português, Inglês e Espanhol são linguagens naturais). NLP é o estudo científico de linguagens da perspectiva computacional, portanto, NLP é um campo da Ciência da Computação e Linguística que abrange as interações entre computadores e linguagem humanas (KUMAR, 2013).

KUMAR ainda define NLP como um uma área significativa dentro da IA, porque um computador pode ser considerado inteligente se o mesmo puder compreender comandos dados em linguagem natural ao invés de linguagens de programação como C[1], FORTRAN[2] ou PASCAL[3]. Um conceito semelhante foi proposto por Alan Turing em 1950, em um famoso teste conhecido como Teste de Turing (*Turing Test*). Neste experimento, ocorria um diálogo entre um avaliador humano e dois outros interlocutores, onde um desses era uma pessoa e o outro uma máquina, e o computador era considerado inteligente se o avaliador não conseguisse distinguir a máquina do ser humano (TURING, 1950).

Dos campos de estudo dentro da NLP, se destacam: Tradução Automática (*Machine Translation*), Extração de Informação (*Information Extraction*), Sumarização Automática de Texto (*Text Summarization*), *Question Answering*, Recuperação de Informação (*Information Retrieval*), Modelagem de Tópicos (*Topic Modeling*) e Mineração de Opiniões (*Opinion Mining*) (CAMBRIA; WHITE, 2014). O presente estudo será focado no tópico de QA, mas encoraja a leitura acerca dos demais.

2.1 *Question Answering*

A área de QA nasceu de uma necessidade advinda da disseminação da Rede Mundial de Computadores (*WWW, World Wide Web*). Cada vez mais informação se fazia presente ao alcance das mãos dos usuários, portanto era necessário que se implementasse uma maneira eficiente de se navegar por esses dados. Pode-se dizer que são um dos conceitos fundamentais dos motores de busca (*Search Engines*), como o Google[4], visto que estes surgiram da demanda por sistemas que permitem aos usuários fazerem perguntas na

[1] https://www.cprogramming.com/
[2] https://fortran-lang.org/en/
[3] https://www.gnu-pascal.de/gpc/h-index.html
[4] https://google.com

sua linguagem cotidiana e receber respostas de maneira rápida e sucinta, com um contexto suficiente para validá-las (HIRSCHMAN; GAIZAUSKAS, 2001). Estes sistemas conseguem retornar uma lista ranqueada de documentos (ou, também, *websites*), mas necessitam de uma implementação de QA para extrair uma resposta (ou até mesmo gerar uma completamente nova).

Na figura 2.1 podemos observar os resultados obtidos ao pesquisar "Quantos planetas existem no sistema solar?". A plataforma apresenta um trecho de texto retirado de um dos documentos obtidos como resultado, evidenciando o uso de metodologias de QA. A resposta da pergunta se apresenta em negrito, enquanto o restante da frase em que foi retirada é apresentado junto para fornecer contexto.

Figura 2.1: Busca realizada na plataforma Google, evidenciando o resultado obtido

Fonte: O Autor, imagem retirada do Google

Uma área emergente onde utilizam-se sistemas de QA são os chamados assistentes virtuais, como: Alexa, Siri, Microsoft Cortana e Google Assistente. Uma das principais capacidades desses assistentes é a capacidade de responder à perguntas e comandos de seus usuários expressados no formato de voz.

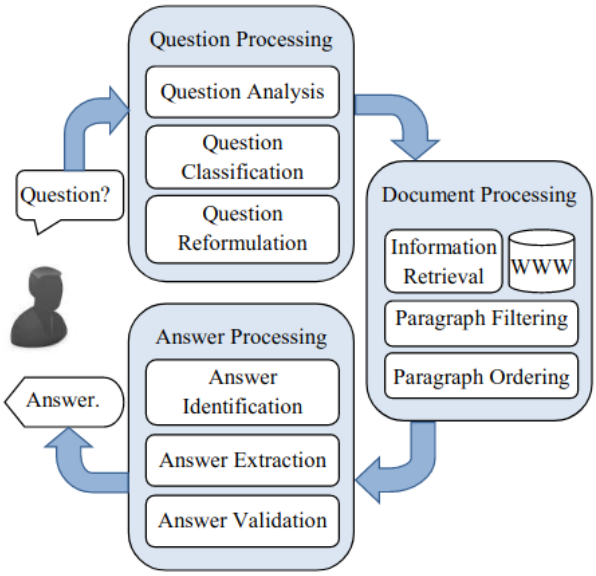
2.1.1 QA: Como funciona

Para ser capaz de responder uma pergunta, o sistema de QA deve primeiro analisar a questão (por vezes, com algum contexto de interação), depois encontrar uma ou mais respostas consultando algum recurso online (como bases de dados) e, por fim, ser capaz de apresentar a(s) resposta(s) de maneira apropriada, na linguagem do usuário e, talvez,

associado com uma justificação ou materiais de suporte (HIRSCHMAN; GAIZAUSKAS, 2001).

De modo geral, um sistema de QA é composto por 3 módulos: Processamento da Questão, Processamento do Documento e Processamento de Resposta. Estes estão ilustrados na figura 2.2. O funcionamento detalhado de cada módulo será descrito nos próximos parágrafos.

Figura 2.2: Arquitetura de um Sistema de *Question Answering*



Fonte: ALLAM; HAGGAG, Figura 1 (2012)

2.1.1.1 Question Processing

O módulo de Processamento de Questão é responsável pela classificação da mesma. Usualmente ele classifica o tipo de questão, deriva o tipo esperado da resposta e reformula a questão em múltiplas outras semanticamente equivalentes. É sabido que reformular a questão aumenta o *recall* do sub-módulo *Information Retrieval* (IR) do próximo componente (ALLAM; HAGGAG, 2012). Sem documentos que possam embasar a resposta do modelo, o mesmo não consegue chegar em uma solução correta.

2.1.1.2 Document Processing

Este módulo é responsável por recuperar informações pertinentes para responder a questão do usuário através de determinadas fontes (no exemplo, o sistema utiliza-se da *web* para recuperar documentos). O componente de IR recebe as questões reformuladas e retorna uma lista ranqueada de documentos, que ainda deverão ser filtrados e ordenados

de modo a auxiliar no funcionamento do próximo módulo.

2.1.1.3 Answer Processing

Por fim, o módulo de Processamento de Respostas é responsável por:

1. Identificar as respostas dentro dos documentos ordenados da etapa anterior;
2. Extrair a resposta escolhendo apenas a palavra ou sentença que melhor se aplica como solução para a pergunta;
3. Validar a resposta, provendo confiança na solução proposta. Isto pode ser atingido de diferentes maneiras, como: fontes de conhecimento específicas do problema, comparando com respostas redundantes de documentos da *web* ou utilizando recursos léxicos como o WordNet[5] para validar que a resposta proposta é do tipo esperado.

2.1.2 QA Factual x Não Factual

Sistemas de QA podem ser classificados quanto ao tipo de pergunta que respondem, o que resulta numa divisão entre perguntas factuais e não factuais. Embora haja esta distinção, conforme mostrado em trabalhos recentes (Noraset, Lowphansirikul and Tuarob (2021); Calijorne Soares and Parreiras (2020); Yogish, Manjunath and Hegadi (2017)) - independentemente do tipo de questão - sistemas de QA possuem uma arquitetura similar (explorada na seção 2.1.1 - QA: Como funciona).

2.1.2.1 Perguntas Factuais

Um pergunta é dita factual (ou factóide) quando a mesma necessita de um fato como resposta, por exemplo: um nome próprio, um lugar ou uma data (YANG et al., 2019). Alguns exemplos casos ser encontrados na tabela 1.2. Normalmente para este tipo de questão, o problema de QA se resume em extrair um pequeno trecho (1 a 3 palavras) de um determinado contexto, como um artigo, ou até mesmo um parágrafo.

[5] https://wordnet.princeton.edu/

2.1.2.2 Perguntas Não Factuais

Ao contrário das anteriores, perguntas não factuais são aquelas que não podem ser respondidas com fatos simples. Via de regra, estas questões demandam respostas mais elaboradas que podem necessitar de informações de mais de uma fonte.

Quando uma pergunta é considerada não factual, ela pode ser caracterizada em diferentes tipos (DIMITRAKIS; SGONTZOS; TZITZIKAS, 2020):

- *Confirmation (yes/no)* - [Confirmação (sim/não)]: Perguntas que podem ser respondidas com sim ou não.
    - e.g. "Porto Alegre é a capital do Rio Grande do Sul?"
- *Definition* - [Definição]: Perguntas cuja resposta é a definição ou descrição de termos
    - e.g. "O que é um dilema?"
- *Causal (how/why/what)* - [Causal(como, por quê, o quê)]: Perguntas que sugerem que a resposta seja uma ou mais consequências de um fato.
    - e.g. "Quais são as consequências da guerra na Ucrânia?"
- *Procedural* - [Procedural]: Perguntas nas quais a resposta requere uma série de ações para concluir algo.
    - e.g. "Quais são os passos para se fazer um bolo?"
- *Comparative* - [Comparativa]: Perguntas que necessitam como respostas um conglomerado de diferenças (ou comparações) entre dois ou mais sujeitos.
    - e.g. "Qual é a diferença entre Ciência da Computação e Engenharia da Computação?"
- *With Examples* - [Com Exemplos]: Perguntas que podem ser respondidas com uma série de exemplos que referenciam o ponto central da questão.
    - e.g. "Quais são os fones de ouvido mais similares aos da marca X?"
- *Opinionated* - [De Opinião]: Perguntas onde se é necessário descobrir a opinião de alguém acerca de um tema ou fato.
    - e.g. "Qual é a opinião do brasileiro sobre a confiabilidade da urna eletrônica?"

2.1.3 *Long-form Question Answering*

LFQA é a tarefa de responder perguntas *open-ended* (que podem abranger qualquer tópico) complexas que demandam respostas que variam entre uma sentença e alguns parágrafos. O termo foi inicialmente introduzido por FAN et al., no trabalho que desenvolveu a - até então - maior base de dados voltada para LFQA, o ELI5 (do inglês, *Explain Like I'm Five*) (2019). Este foi o conjunto de dados no qual o presente estudo se baseia. Maiores detalhes serão explorados na seção de Metodologia.

2.2 *Transformers e Geração de Linguagem Natural*

A tarefa de NLG consiste em gerar automaticamente linguagem natural compreensível para outros seres humanos, no geral, em formato de texto utilizando-se de algum contexto de entrada que tipicamente é textual, mas pode ser não linguístico (REITER; DALE, 1997). Autores da área costumam atribuir à tarefa de NLG apenas os casos em que a entrada dos algoritmos constitui-se de dados em forma não-textual, mas segundo Gatt and Krahmer (2018) os dois tipos de geração (*data-to-text* e *text-to-text*) apresentam diversas semelhanças, apontando a convergência das tarefas para a NLG.

Além de um texto compreensível, os modelos ainda se preocupam em gerar linguagem natural com o máximo de semelhança em relação à produzida por seres humanos, no sentido de que além de questões sintáticas e semânticas, estes ainda buscam eliminar ambiguidades e realizam decisões para transpassar seu objetivo comunicativo.

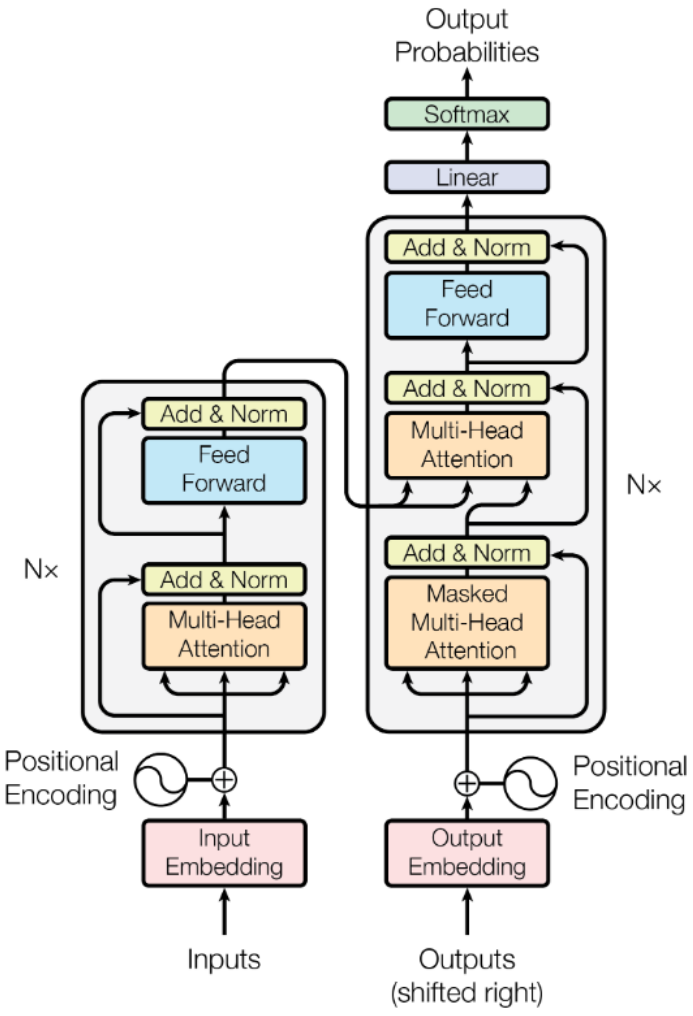
Os primeiros sistemas de NLG foram construídos com técnicas baseadas em *templates*, onde são gerados modelos de *template* com algumas lacunas que são dinamicamente preenchidas com informações (REITER; DALE, 1997). As pesquisas mais recentes convergem para sistemas mais complexos, que normalmente treinam modelos estatísticos recorrendo a *Machine Learning*, que fazem uso de bases de dados extensas (GEHRMANN et al., 2021).

Em 2017, VASWANI et al. introduziram o modelo chamado de *Transformer*, cuja principal contribuição foi um mecanismo de atenção (*self-attention*) que permite aos modelos manter referências de todas as palavras da sentença de entrada e saída. Este trabalho, de certa forma, revolucionou a área que até então se baseava na utilização de Redes Neurais Recorrentes (RNN, do inglês *Recurrent Neural Networks*) ou Redes Neurais Convolucionais (CNN, do inglês *Convolutional Neural Networks*).

As RNNs funcionam consumindo a entrada e produzindo a saída de maneira sequencial. Durante o processo é gerada uma sequência de estados ocultos (*Hidden States*) h_t como uma função do estado anterior h_{t-1} e a posição t da entrada (onde $t = 0$ representa a primeira palavra). Essa maneira sequencial não faz um uso otimizado da paralelização, um dos pontos mais fortes das GPUs e TPUs. Dessa maneira, mecanismos de atenção foram propostos como parte fundamental destes modelos, pois estes possibilitam que dependências maiores do que a palavra imediatamente anterior possam ser levadas em consideração.

Por este motivo, o modelo de *Transformers* foi proposto por VASWANI et al. O modelo se baseia completamente no mecanismo de atenção para traçar dependências globais entre a entrada e a saída (em oposição ao modelo sequencial, que era estado da arte até então). A arquitetura geral do *Transformer* pode ser observada na figura 2.3.

Figura 2.3: Arquitetura de um Modelo baseado em *Transformers*.



Fonte: VASWANI et al., Figura 1 (2017)

A arquitetura de um *Transformer* consiste em um modelo com estrutura *encoder-decoder*, onde o primeiro mapeia a entrada para uma sequência de símbolos $x = (x_1, ..., x_n)$, e posteriormente para uma sequência de representações contínuas $z = (z_1, ..., z_n)$. Partindo de z, o *decoder* gera uma sequência de saída $y = (y_1, ..., y_n)$ de símbolos, um elemento por vez. A cada passo, o modelo é auto-regressivo, isto é, ele consome os símbolos que foram gerados previamente como uma entrada adicional para gerar o próximo (VASWANI et al., 2017). Acima disso, o modelo ainda usa camadas de atenção, conectando completamente as camadas do *encoder* e do *decoder*.

Desde a sua publicação, *Transformers* têm ganhado uma visibilidade e importância muito grandes na área. Recentemente, muitos trabalhos foram publicados baseados em *Transformers*, como BERT (DEVLIN et al., 2018) e GPT-3 (BROWN et al., 2020). Estes modelos demonstraram a capacidade de gerar textos fluentes e coesos, semelhante aos produzidos por outros seres humanos (CHEN et al., 2020). Em especial o GPT-3 é utilizado na tarefa de NLG, recebendo uma passagem de texto e produzindo algo inteiramente novo.

2.3 Modelos Pré-treinados

Uma das características mais marcantes de modelos como BERT e GPT-3 é que foram idealizados para serem pré-treinados. Essa característica já se provou muito efetiva quando o assunto é melhorar o desempenho nas mais variadas tarefas de NLP (Dai and Le (2015); Peters et al. (2018)). Os modelos são treinados em bases de dados extensivas na tarefa de *Language model pre-training*, fazendo com que possam ser facilmente adaptados para tarefas mais específicas como: *Next Sequence Prediction*, *Sequence Classification*, *Multiple-Choice Classification*, *Token Classification*, *Summarization* e *Question Answering*. O processo de adaptação desse tipo de modelo para uma outra tarefa é chamado de *fine-tuning*.

Outros modelos notáveis e um pouco mais recentes são RoBERTa (LIU et al., 2019) e T5 (RAFFEL et al., 2019), sendo o primeiro proposto como uma otimização do pré-treinamento do BERT e o segundo como uma maneira de transformar todas as tarefas de NLP em tarefas do tipo *text-to-text*. T5 foi o nome dado ao *Text-To-Text Transfer Transformer*, proposto por pesquisadores da empresa Google em 2019.

Finalmente, ambos os campos de QA e NLG se beneficiam do avanço destes modelos baseados em *Transformers*. Em ambos os casos, quando a tarefa final envolve

responder questões (seja via NLG, isto é, gerando um texto novo ou por meio de extração de informação) os modelos recebem uma entrada similar. Ela constitui-se da questão e o contexto, sendo estes delimitados por algum *token* específico e concatenados numa só sentença. Além disto, o modelo também recebe a resposta que deveria ser gerada a partir desses dados. Um exemplo será explorado com mais detalhes no Capítulo 3, Metodologia.

2.4 Trabalhos Relacionados

2.4.1 ANTIQUE: *A Non-Factoid Question Answering Benchmark*

De maneira similar à proposta nesta pesquisa, Hashemi et al. (2020) propuseram um estudo abordando QA de maneira não-factual e também de domínio aberto (*Open-Domain*), isto é, questões que abordam os mais variados tópicos. Os autores se basearam na extinta[6] plataforma Yahoo! Answers, promovendo através do método *crowdsourcing* uma classificação feita por pessoas acerca da qualidade das respostas. A plataforma era uma conhecida por seu sistema de perguntas e respostas, onde todo o conteúdo era gerado pela comunidade. Este contexto é semelhante ao da plataforma utilizada nesta pesquisa, como será discutido na seção 3.2.

O resultado foi uma base de dados composta por 2626 perguntas e 34011 respostas, destas 11668 foram categorizadas como não suficientes para responder a sua pergunta (categorias 1 e 2 do estudo) (HASHEMI et al., 2020).

Neste *benchmark* proposto pelos autores, como no presente estudo, foi utilizado um modelo baseado na arquitetura de *Transformers* (BERT). O mesmo mostrou performance superior quando comparado a outros sistemas que não tinham como base a mesma arquitetura, mostrando que essa categoria de modelos trouxe um novo estado da arte (com base nas métricas utilizadas pelos autores) e que o problema de QA não factual ainda estava em aberto.

Como a tarefa do ANTIQUE se baseava em classificação de respostas, as métricas coletadas se diferem das utilizadas no presente estudo, mas os resultados obtidos podem ser observados na tabela 2.1.

[6] https://help.yahoo.com/kb/SLN35642.html

Tabela 2.1: Tabela comparativa de diferentes técnicas no *benchmark* ANTIQUE

Método	MAP	MRR	P@1	P@3	P@10	nDCG@1	nDCG@3	nDCG@10
BM25	0.1977	0.4885	0.3333	0.2929	0.2485	0.4411	0.4237	0.4334
DRMM-TKS	0.2315	0.5774	0.4337	0.3827	0.3005	0.4949	0.4626	0.4531
aNMM	0.2563	0.6250	0.4847	0.4388	0.3306	0.5289	0.5127	0.4904
BERT	**0.3771**	**0.7968**	**0.7092**	**0.6071**	**0.4791**	**0.7126**	**0.6570**	**0.6423**

Fonte: Adaptado de Hashemi et al. (2020)

2.4.2 *Question Answering with Long Multiple-Span Answers*

Zhu et al. (2020) propuseram um estudo gerando a base de dados MASH-QA, baseada em perguntas não-factuais e conhecimentos retirados de artigos do domínio da saúde. Os autores reportam contextos (ou *Support Documents*, como serão explorados na seção 3.3) extensos, construídos exclusivamente de trechos de texto, da mesma maneira que o presente estudo.

De maneira similar à este trabalho, Zhu et al. (2020) realizaram treinamento em Redes Neurais baseadas em *Transformers*, mais especificamente no modelo *Transformer-XL*, proposto por Dai et al. (2019). O modelo em questão possui um mecanismo de atenção mais extenso, conseguindo lidar com contexto bem maiores, como ocorre nos documentos médicos utilizados na pesquisa *Question Answering with Long Multiple-Span Answers*.

Da mesma maneira que o ANTIQUE, a base de dados proposta por Zhu et al. (2020) possui como tarefa um problema de classificação. No caso deles, era necessário que os modelos classificassem cada uma das sentenças no documento de suporte como pertencendo ou não à resposta para a pergunta em questão. Os resultados encontrados pelos autores avaliando os modelos em seu conjunto de dados proposto são evidenciados na tabela 2.2. Todos os modelos presentes na tabela possuem como base a arquitetura *Transformers*, e o melhor resultado obtido foi através do modelo proposto pelos autores, o MultiCo.

Tabela 2.2: Tabela comparativa de diferentes modelos na base de dados MASH-QA

Modelo	P	R	F1
TANDA	56.48	16.42	25.44
BERT	56.18	16.25	25.21
RoBERTa	57.70	19.06	28.65
XLNet	56.05	19.73	29.19
MultiCo	**58.16**	**55.90**	**57.00**

Fonte: Adaptado de Zhu et al. (2020)

2.4.3 WikiPassageQA: *A Benchmark Collection for Research on Non-factoid Answer Passage Retrieval*

O estudo proposto por Cohen, Yang and Croft (2018) - assim como o ANTIQUE - estabeleceu um *benchmark*, que pode ser visto como uma maneira de fazer um comparativo da performance de diferentes técnicas para realizar a tarefa proposta pelo *dataset*. Como destacado no título do trabalho, o foco é sobre *Passage Retrieval*, isto é, a capacidade de retirar informações importantes (ou trechos) de extensos textos.

O tipo citado de tarefa favorece, normalmente, modelos extrativos. Esses modelos são treinados para extrair passagens do texto como estão escritas, ao contrário do presente estudo que treinou modelos para geração de um conteúdo inteiramente novo através da pergunta, contexto e conhecimento prévio do modelo.

Assim como o atual trabalho, o *WikiPassageQA* lida com problemas de domínio aberto, mas por ter a característica de ser um problema extrativo, favorece à performance de métodos tradicionais como BM25 e QL. Foram utilizados como comparação modelos com arquitetura mais clássica de CNNs, os autores do *WikiPassageQA* não reportaram o uso de *Transformers*. Ainda assim, a melhor performance obtida foi atingida utilizando CNN, como pode ser reparado na tabela 2.3.

Tabela 2.3: Tabela comparativa de diferentes técnicas no *benchmark* WikiPassageQA

Type	Method	MAP	MRR	P@5	P@10	nDCG	Recall@5	Recall@10	Recall@20
Base	WC	0.3456	0.4004	0.1370	0.0923	0.5096	0.4618	0.6079	0.7615
	WC.IDF	0.3417	0.3898	0.1351	0.0928	0.5049	0.4518	0.6129	0.7526
Traditional IR	VSM	0.3970	0.4588	0.1476	0.0921	0.5490	0.4837	0.5979	0.7464
	BM25	0.5373	0.6258	0.1947	0.1151	0.6659	0.6334	0.7311	0.8309
	QL	0.5436	0.6338	0.1947	0.1151	0.6715	0.6353	0.7275	0.8426
Neural IR	LSTM	0.3352	0.3947	0.1197	0.0780	0.4912	0.3915	0.5894	0.7169
	CNN+TF	0.4009	0.4581	0.1572	0.1099	0.5577	0.5212	0.7024	0.8412
	LSTM-CNN+TF	0.3577	0.4156	0.1351	0.0942	0.5196	0.4538	0.6187	0.7608
	Char+WordCNN-LSTM	0.4385	0.5534	0.1728	0.1104	0.5837	0.5709	0.6931	0.8326
	Memory-CNN-LSTM+TF	**0.5608**	**0.6792**	**0.2083**	**0.1228**	**0.6791**	**0.6522**	**0.7329**	**0.8592**

Fonte: Adaptado de Cohen, Yang and Croft (2018)

2.4.4 Resumo dos Trabalhos Relacionados

Todos os trabalhos citados foram estudos realizados acerca do tema de QA não-factual, com algumas semelhanças e diferenças do presente trabalho. Na tabela 2.4 são explicitadas algumas dessas características.

Nota-se que por se tratar de um problema de classificação de corretude e com-

Tabela 2.4: Comparação de Trabalhos Relacionados

Pesquisa	**Domínio aberto**	**Tipo de QA**	**Utiliza *Transformers***
ANTIQUE	Sim	Não se aplica	Sim
MASH-QA	Não	Extrativo	Sim
WikiPassageQA	Sim	Extrativo	Não
Presente estudo	Sim	Abstrativo	Sim

Fonte: O Autor

pletude de respostas, o tipo de QA abordado pelo *benchmark* ANTIQUE não se aplica. Ademais, com exceção do WikiPassageQA, todos utilizaram alguma implementação de modelo baseada em *Transformers* para fins de comparação. Por fim, apenas o *dataset* MASH-QA abordou QA de domínio fechado, concentrando-se em perguntas e respostas do domínio de saúde.

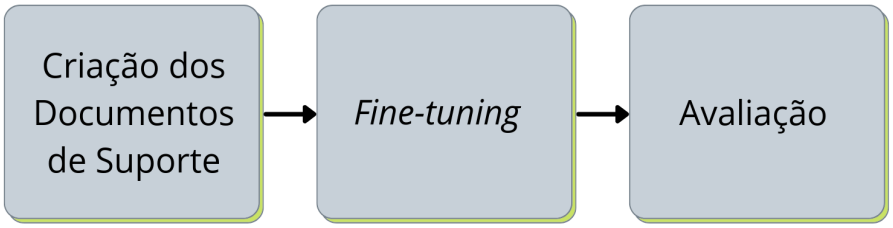
3 METODOLOGIA

Como explicitado anteriormente, o objetivo deste trabalho é fazer um comparativo da performance de diferentes modelos pré-treinados baseados na arquitetura de *Transformers*. Para tanto, nesta seção será explorada a metodologia utilizada no processo, partindo de uma visão geral das etapas na seção 3.1. Nas seções seguintes: em 3.2 a base de dados será detalhada; em 3.3 será explicado o *Support Document*, bem como o *dataset* auxiliar; em 3.4 e 3.5 serão vistos em detalhes os modelos comparados neste projeto; na seção 3.6 será discutida a plataforma utilizada para obter os modelos pré-treinados; em 3.7 serão evidenciadas as métricas utilizadas para esta pesquisa; seção 3.8 detalha o processo de treinamento, bem como hiperparâmetros utilizados; e, por fim, a seção 3.9 descreve a metodologia empregada para comparar os modelos.

Os detalhes sobre o ambiente computacional utilizado nesta pesquisa, bem como ferramentas, pacotes e configurações de máquina serão explorados no Apêndice C - Ambiente Computacional.

3.1 Visão Geral

Figura 3.1: Etapas da pesquisa



Fonte: O Autor

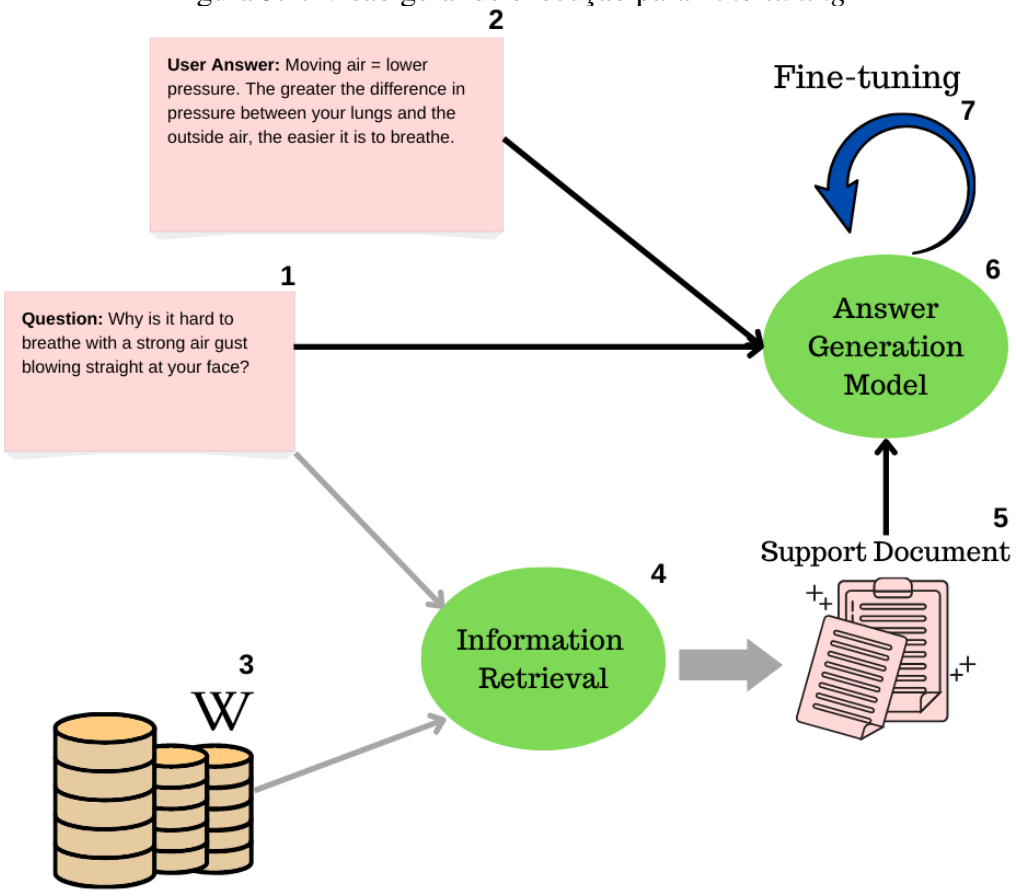
O presente estudo foi dividido em 3 etapas: Criação dos Documentos de Suporte, *Fine-tuning* e Avaliação. Antes de entrar em detalhes sobre as etapas realizadas, serão definidos alguns conceitos chave utilizados nesta seção:

1. *Question*: Uma pergunta retirada diretamente do *dataset* ELI5;
2. *User Answer*: A resposta considerada como correta, feita por uma pessoa (também retirada do *dataset* ELI5);
3. *wiki_snippets*: A base de dados auxiliar utilizada neste trabalho;
4. *Information Retrieval*: O módulo responsável por reunir informações pertinentes

acerca de cada pergunta na base de dados *wiki_snippets*;

5. *Support Document*: A saída do módulo de IR, um texto feito através da concatenação de trechos obtidos da *Wikipedia*;
6. *Answer Generation Model*: Módulo onde os diferentes modelos pré-treinados são carregados. É onde ocorre o processamento central do programa, isto é, onde é feito o treinamento (*fine-tuning*) e posterior processamento das perguntas e documentos de suporte para respostas;
7. *Fine-tuning*: Etapa do processo onde o modelo é refinado sobre o modelo pré-treinado, sendo exposto à perguntas, documentos de suporte e respostas de usuários. A figura 3.2 mostra o processo em mais detalhes;
8. *Model input*: Entrada dos modelos, composta por: *Question*, *User Answer* e *Support Document*;
9. *Model Answer*: Resposta gerada pelo modelo. Exemplos gerados nesta pesquisa podem ser observados no Apêndice B - Respostas Geradas Pelos Modelos.

Figura 3.2: Visão geral de execução para *Fine-tuning*



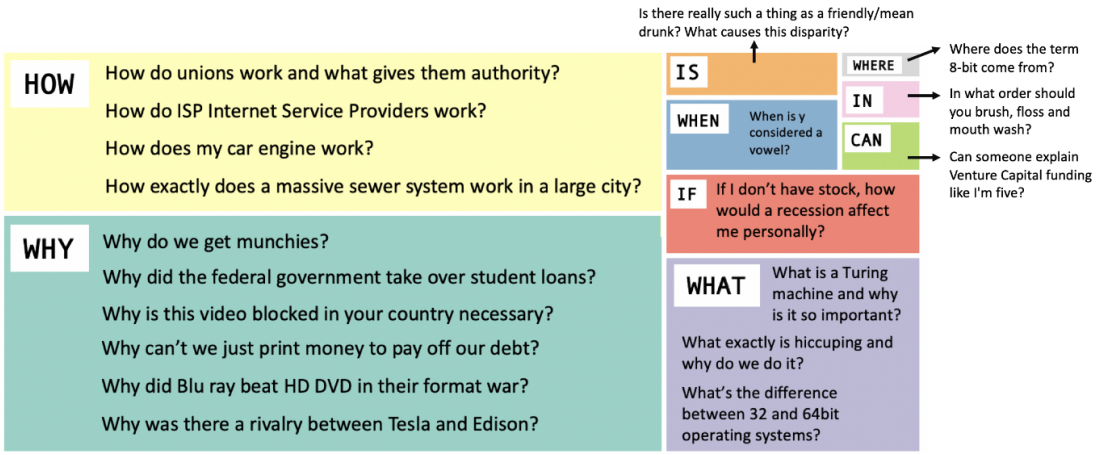
Fonte: O Autor

3.2 ELI5: *Explain Like I'm Five*

Esta é a base de dados principal desta pesquisa, foi introduzido em 2019 por FAN et al.. Foi o primeiro estudo a explorar exclusivamente problemas envolvendo perguntas não factuais utilizando informações de diferentes fontes. Este *dataset* é composto por mais de 270 mil pares de perguntas e respostas, sendo elas consideradas não factuais. Exemplos de perguntas, bem como a distribuição geral dos pronomes que iniciam cada uma, podem ser observados na figura 3.3.

KHUSHHAL et al. exploram em seu trabalho o que chamam de *community question answering*, ou seja, sistemas de QA baseados em plataformas onde pessoas perguntam e respondem outras pessoas. É o caso do *dataset* ELI5, pois o mesmo é baseado na plataforma *Reddit*[1].

Figura 3.3: Disposição da frequência das palavras iniciais das perguntas. O tamanho da caixa indica frequência.



Fonte: FAN et al., Figura 2 (2019)

A plataforma *Reddit* se define como uma rede onde seus usuários podem compartilhar seus interesses, *hobbies* e paixões. Ela consiste de um agregador social de notícias, conteúdo geral e discussões. A mesma se divide em várias comunidades chamadas de *subreddits*, que são nichos onde pessoas discutem conteúdos específicos, cada um com seus assuntos e regras (ANDERSON, 2015).

Uma dessas comunidades (ou *subreddit*) é a chamada *Explain Like I'm Five* (ELI5), cujo propósito geral é simplificar conceitos complexos de maneira que possam ser entendidos por leigos[2]. A tradução do acrônimo é: explique como se eu fosse uma criança de

[1] https://www.reddit.com/
[2] https://www.reddit.com/r/explainlikeimfive/wiki/detailed_rules/

5 anos, evidenciando o nível de simplificação desejado nas respostas.

Nessas comunidades os usuários postam perguntas e outros usuários respondem. Na imagem 3.4 é evidenciado um exemplo de pergunta junto da resposta mais votada. A plataforma conta com um sistema de ranking baseado em *Upvotes*, e segundo Fan et al. (2019) a resposta mais votada deve ser utilizada como referência. Algo similar ao que ocorre na maioria de sistemas baseados em plataformas de perguntas e respostas (KHUSHHAL et al., 2020).

Figura 3.4: Exemplo de post no subreddit *ELI5* e a resposta mais votada

Fonte: O Autor, retirado de Reddit, usuários omitidos

Diferentemente do trabalho proposto por Fan et al. (2019), onde foram coletados pares de perguntas e respostas datando até Julho de 2018, esta pesquisa utilizou uma versão um pouco mais recente da base de dados. A versão utilizada pode ser encontrada na plataforma *Huggingface*[3], e coleta dados até Agosto de 2019. A versão da plataforma possui 272 mil pares de perguntas e respostas no conjunto de treino, um adicional de 35 mil perguntas comparado com o anterior (o original possui 237 mil pares de perguntas e respostas no conjunto de treino).

A base de dados foi dividida em 3 secções (também chamados de *splits*): treino, validação e teste. Os subconjuntos de validação e teste foram criados comparando todas as questões do *dataset* através do método TF-IDF, e aquelas com menor semelhança foram utilizadas nestes conjuntos (evitando que questões muito semelhantes aparecessem simultaneamente no conjunto de treino e nos demais, o que poderia fazer com que os modelos memorizassem respostas). Na tabela 3.1 são mostrados os tamanhos exatos dos conjuntos resultantes.

Na sua pesquisa, FAN et al. utilizaram dados coletados na *web* através da plata-

[3]https://huggingface.co/

Tabela 3.1: Distribuição das questões na base de dados ELI5

Split	**Número de Registros**
Treino	272634
Validação	24512
Teste	9812

Fonte: O Autor

forma *Common Crawl*[4], resultando em documentos de tamanho médio de 857.6 palavras. Este trabalho se diferencia utilizando-se de outra fonte de conhecimento: a *Wikipedia*.

3.3 *Support Documents*

Depois de recuperar as perguntas e respostas do *dataset* ELI5, o próximo passo necessário é construir documentos de suporte, também chamados de contexto, para alimentar os modelos. Para cada uma das perguntas do conjunto de treino foram criados documentos de suporte extraídos de dados coletados da *Wikipedia*. Essa escolha foi necessária, pois os modelos selecionados para esta pesquisa não conseguem lidar com entradas de tamanho tão grande, sendo necessário reduzir a entrada para um tamanho máximo de 510 ou 1024 palavras (conforme o modelo, detalhes serão explorados ao final desta seção).

3.3.1 *Wikipedia*

A base de dados auxiliar utilizada neste trabalho foi a *wiki_snippets*[5]. É um *dataset* com pouco mais de 10 GB de tamanho, composto por páginas da *Wikipedia* filtradas para remover páginas ambíguas, de redirecionamento ou deletadas. Ao fim da filtragem, todos os textos restantes são divididos em trechos (*snippets*) de até 100 palavras. Um exemplo de registro desta base de dados pode ser visto na figura 3.5. Os campos de interesse são: *article_title* e *passage_text*, que representam: o título da página da *Wikipedia* correspondente e a passagem de texto (no caso, com exatas 100 palavras).

[4]http://commoncrawl.org
[5]https://huggingface.co/datasets/wiki40b

Figura 3.5: Exemplo de registro da base de dados *wiki_snippets*


```
{
  "_id": "{\"nlp_id\": 1665408, \"wiki_id\": \"Q1069369\", \"sp\": 10, \"sc\": 720, \"ep\": 14, \"ec\"
    : 70}",
  "article_title": "Hazardous waste",
  "end_character": 70,
  "end_paragraph": 14,
  "nlp_id": 1665408,
  "passage_text": "EPA determined that some specific wastes are hazardous. These wastes are
    incorporated into lists published by the Agency. These lists are organized into three categories:
    F-list (non-specific source wastes) found in the regulations at 40 CFR 261.31, K-list (source
    -specific wastes) found in the regulations at 40 CFR 261.32, and P-list and the U-list (discarded
    commercial chemical products) found in the regulations at 40 CFR 261.33.\nRCRA's record keeping
    system helps to track the life cycle of hazardous waste and reduces the amount of hazardous waste
    illegally disposed. Comprehensive Environmental Response, Compensation, and Liability Act The
    [Comprehensive Environmental Response, Compensation, and Liability",
  "section_title": "Resource Conservation and Recovery Act (RCRA) & Comprehensive Environmental
    Response, Compensation, and Liability Act",
  "start_character": 720,
  "start_paragraph": 10,
  "wiki_id": "Q1069369"
}
```


Fonte: O Autor

3.3.2 Criação dos *Support Documents*

Para criar o contexto, foi utilizada a ferramenta *Elasticsearch*. Segundo a definição do *website* próprio da ferramenta[6]: "o Elasticsearch é um mecanismo de busca e análise de dados distribuído, gratuito e aberto para todos os tipos de dados, incluindo textuais, numéricos, geoespaciais, estruturados e não estruturados". Neste trabalho foi utilizada a versão 7.7.0 do *Elasticsearch*[7].

A ferramenta realiza uma indexação dos dados para possibilitar uma computação mais rápida da busca. Para esta pesquisa, a *query* consiste da pergunta em questão do ELI5. Especificamente, o *Elasticsearch* categoriza os vizinhos mais próximos de cada documento utilizando a função de similaridade BM25, que baseia-se na utilização de TF-IDF para atribuir valores à palavras. Deste modo, em última instância, o motor central da busca utiliza-se de um método de *bag-of-words* junto da frequência das palavras.

Então, para cada questão do *split* de treino do ELI5 o *Elasticsearch* será executado em busca dos top 10 *snippets* que possuem mais similaridade com o texto em questão. Para auxiliar na busca, são consideradas apenas as palavras fora da lista de *stop-words* (preposições, artigos, conjunções e outros, como o próprio nome do *subreddit*). A ferramenta retorna uma lista de tamanho k arbitrário, no caso deste estudo $k = 10$. Um

[6]https://www.elastic.co/pt/what-is/elasticsearch
[7]https://www.elastic.co/downloads/past-releases/elasticsearch-7-7-0

exemplo de retorno do *Elasticsearch* será fornecido no Apêndice A - Exemplo de Support Document.

No caso do modelo BART, é possível passar entradas de tamanho até 1024 palavras (utilizando-se, portanto, de todos os 10 trechos selecionados). Já no caso do modelo T5, as opções pré-treinadas reconhecem entradas de tamanho até 510 palavras (512 contando os tokens especiais de início e fim), portanto utilizando-se dos top 5 trechos apenas.

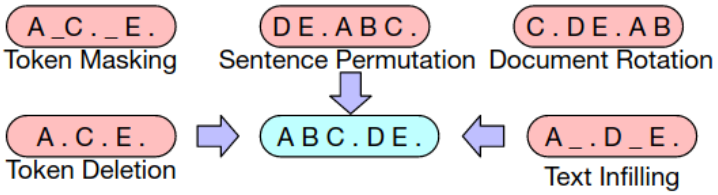
Por fim, os trechos são concatenados em um grande parágrafo e fornecidos aos modelos junto com as perguntas e respostas para treinamento e posterior predição.

3.4 BART

O primeiro dos modelos pré-treinados utilizados nesta pesquisa. A definição dos criadores do modelo é a seguinte: *"BART is a denoising autoencoder for pretraining sequence-to-sequence models"* (LEWIS et al., 2019), que - em tradução livre - significa que é um modelo que faz uma codificação automática de eliminação de ruído, feito para pré-treinamento em modelos *sequence-to-sequence*, ou seja, aqueles que recebem como entrada texto e geram texto como saída.

De modo geral, um certo ruído é introduzido de maneira arbitrária na entrada e o modelo é treinado para reconstruir o texto original. As funções de inserção de ruído utilizadas pelos autores podem ser conferidas na figura 3.6. Os autores relatam que o modelo funciona melhor para tarefas de NLG, mas que também pode ser treinado para tarefas de compreensão de texto. O BART consegue atingir resultados semelhantes à modelos que já foram consolidados, como RoBERTa (LIU et al., 2019), e também atinge o estado da arte em tarefas que envolvem abstração, como QA abstrativo, geração de diálogo e sumarização abstrativa.

Figura 3.6: Exemplo de inserção de ruído na entrada para o modelo BART



Fonte: LEWIS et al., Figura 2 (2019)

LEWIS et al. desenvolveram duas versões do modelo BART: BART-Small com 6

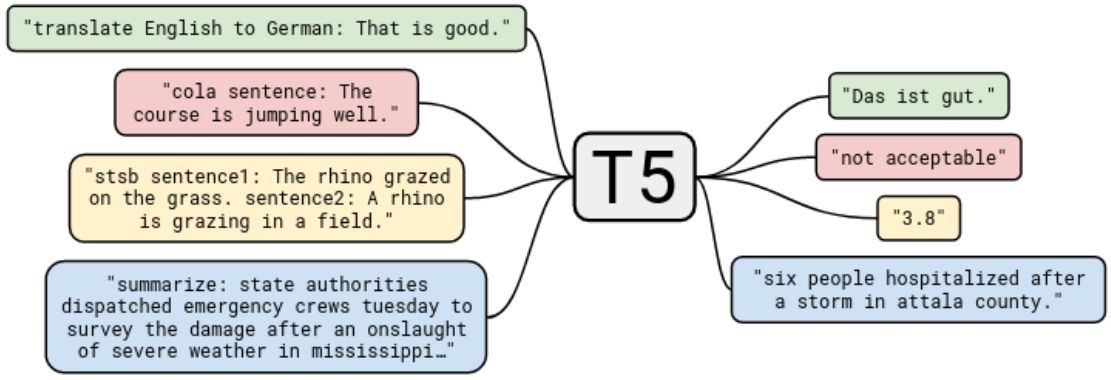
camadas (*layers*) de *encoders* e *decoders*, totalizando 140 milhões de parâmetros internos; e BART-Large com 12 camadas (*layers*) de *encoders* e *decoders*, totalizando 400 milhões de parâmetros internos. Por já possuir um modelo treinado sobre o *dataset* ELI5, esta pesquisa utilizará o modelo BART-Large.

3.5 T5

Introduzido por Raffel et al. (2019), o modelo **T**ext-**t**o-**T**ext **T**ransfer **T**ransformer (**T5**) foi inicialmente apresentado no *Journal of Machine Learning Research* 21 (2020). Desenvolvido por pesquisadores da empresa Google[8], o modelo possui como diferencial unificar todos os problemas baseados em texto em um formato *text-to-text*. Isto possibilita a utilização do mesmo modelo, objetivo, procedimento de treinamento, *loss function*, hiperparâmetros e procedimento de decodificação para todas as tarefas (RAFFEL et al., 2019).

Na figura 3.7 observa-se que diferentes tarefas: tradução, verificar se uma sentença possui a gramática correta (introduzido por Warstadt, Singh and Bowman (2018), o Corpus of Linguistic Acceptability, CoLA), similaridade semântica entre textos (do inglês Semantic Textual Similarity Benchmark, STSB) e sumarização.

Figura 3.7: Funcionamento do modelo T5, todas as tarefas são convertidas em processar texto e gerar texto, inclusive problemas de classificação.



Fonte: RAFFEL et al., Figura 1 (2019)

O modelo foi pré-treinado extensivamente em dados provenientes da plataforma Common Crawl, sendo exposto a mais de 750 GB de dados em formato textual na língua inglesa.

[8] https://www.google.com

Existem 5 variações do modelo T5: T5-Small, T5-Base, T5-Large, T5-3B, T5-11B. Cada uma dessa variações possui uma quantidade diferente de parâmetros internos (o tamanho do modelo), em ordem de aparição os tamanhos são: 60 milhões, 220 milhões, 770 milhões, 3 bilhões e 11 bilhões. Nas tarefas testadas por RAFFEL et al., o modelo que atinge melhores resultados é o T5-11B, mas é necessário levar em consideração o custo computacional também. Em razão dos recursos disponíveis, o modelo explorado neste trabalho será o T5-Small, que possui um desempenho ligeiramente pior que o T5 base, mas aproximadamente 1/4 do seu tamanho.

3.6 Hugging Face ®

Hugging Face[9] é uma startup focada em NLP com uma grande comunidade *open-source*. Em particular, a plataforma ficou famosa por sua implementação da biblioteca *Transformers* (WOLF et al., 2020), que permite utilizar em Python[10] uma API para fazer uso de vários modelos baseados na arquitetura de *Transformers*, como: BERT, RoBERTa, GPT-2, DistilBERT, BART e T5.

A plataforma ainda oferece, em seu HUB[11] mais de 60 mil modelos pré-treinados e mais de 6 mil bases de dados para explorar os desafios de NLP. Ambos as bases de dados utilizadas neste trabalho, ELI5[12] e *wiki_snippets*[13] são provenientes dela. Além de oferecer modelos e *datasets*, a plataforma ainda possui uma série de tutoriais e *notebooks* (arquivos no formato *.ipynb* que podem ser executados através do software Jupyter Notebook[14]) para auxiliar no desenvolvimento de aplicações para NLP.

Por esses motivos, Hugging Face se diz uma plataforma com o objetivo de democratizar a NLP, tornando acessível o acesso a modelos treinados em larga escala para pesquisadores e usuários (WOLF et al., 2020).

[9] https://huggingface.co/
[10] https://www.python.org/
[11] https://huggingface.co/docs/hub/index
[12] https://huggingface.co/datasets/eli5
[13] https://huggingface.co/datasets/wiki_snippets
[14] https://jupyter.org/)

3.7 Métricas

A avaliação da saída de aplicações *text-to-text* ainda é um tópico em aberto para pesquisadores da área, visto que é um problema em diversas tarefas, como: *machine translation, text simplification, text summarization, grammatical error correction, and natural language generation* (SPECIA; SCARTON; PAETZOLD, 2018).

Seguindo a métrica utilizada pelos próprios autores do *dataset*, este trabalho utilizará o método ROUGE, proposto por LIN em 2004. A métrica é utilizada para avaliar a similaridade entre a saída de um modelo e uma (ou mais) referências. No caso do presente estudo, a saída dos modelos será avaliada de acordo com a resposta de maior *score* entre as possíveis.

Formalmente, $\mathrm{ROUGE\text{-}N}$ é calculado através da seguinte fórmula (LIN, 2004):

$$\mathrm{ROUGE\text{-}N} = \frac{\sum_{S \in \{ReferenceSummaries\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{ReferenceSummaries\}} \sum_{gram_n \in S} Count(gram_n)} \tag{3.1}$$

Onde n significa o tamanho do n-grama $gram_n$, e $Count_{match}(gram_n)$ é o número máximo de n-gramas co-ocorrendo na saída do modelo e no conjunto de referências. Para fins desta pesquisa, um n-grama (ou *n-gram*) é uma sequência contínua de n palavras.

Evidentemente o ROUGE-N é uma medida orientada a *recall*, visto que o denominador da equação é o somatório da quantidade de n-gramas ocorrendo no texto de referência. Para este trabalho serão apresentados ROUGE-1 e ROUGE-2 e ROUGE-L (uma medida semelhante ao ROUGE-N, porém usa maior subsequência comum entre a saída predita pelo modelo e a resposta de referência), indo de acordo com as métricas utilizadas para avaliar a performance de modelos no mesmo *dataset*.

A maneira clássica do ROUGE-N avalia apenas o *recall*, como dito anteriormente, mas é possível calcular *precision e f1-score* de maneiras semelhantes:

$$ROUGE_{Recall} = \frac{\#overlaps}{\#\mathrm{reference\_N\text{-}grams}} \tag{3.2}$$

Onde $\#\mathrm{overlaps}$ é o somatório de n-gramas comuns entre a saída predita pelo modelo (*Model Answer*) e a resposta correta (*User Answer*), e $\#\mathrm{reference\_N\text{-}grams}$ é a

quantidade total de n-gramas na frase de referência (*User Answer*).

$$ROUGE_{Precision} = \frac{\#overlaps}{\#\text{predicted\_N-grams}} \tag{3.3}$$

Onde $\#\text{predicted\_N-grams}$ é a quantidade total de n-gramas na saída predita pelo modelo (*Model Answer*).

$$ROUGE_{F1} = \frac{2 * (ROUGE_{Precision} * ROUGE_{Recall})}{ROUGE_{Precision} + ROUGE_{Recall}} \tag{3.4}$$

A implementação do método ROUGE utilizada nesta pesquisa corresponde à utilizada por Fan et al. (2019). Ela pode ser encontrada no repositório GitHub dos autores[15].

3.8 Treinamento

Para investigar o processo de *fine-tuning*, são detalhados os métodos de treinamento e hiperparâmetro utilizados para ambos os modelos. De maneira geral, o modelo recebe como *Model Input* três componentes: *Question*, *User Answer* e *Support Document*; de posse da entrada, o processo de *fine-tuning* consiste em fazer o modelo ser capaz de chegar em resultados próximos da *User Answer*, utilizando a pergunta original, o documento de suporte e seu conhecimento prévio. Em relação aos parâmetros, serão apenas elucidados aqueles que diferem dos *default* implementados e que impactam no resultado do treinamento.

Visto que o componente de *hardware* que possui mais impacto no treinamento de *Transformers* são as GPUs, e ambos os modelos utilizados nesta pesquisa foram treinados sobre GPUs, as CPUs serão desconsideradas.

As entradas para os tipos de modelos explorados nesta pesquisa precisam de um pré-processamento especial chamado de *padding* e *truncation*. O *padding* se encarrega que todas as entradas possuam um mesmo tamanho, a estratégia utilizada nesta pesquisa é fazê-lo baseado na maior entrada passada pro modelo, de modo a não perder informação nesta etapa. O processo de *truncation* garante que o modelo não receba entradas maiores do que consegue lidar, o parâmetro max_input_length nos modelos abaixo pode ser entendido como o tamanho máximo da entrada, em *tokens* (ou palavras). A entrada é considerada até o tamanho máximo, qualquer conteúdo que vier depois do max_input_length é desconsiderado.

[15] https://github.com/facebookresearch/ELI5/tree/main/model_code

3.8.1 BART

O modelo utilizado nesta pesquisa foi treinado por Jernite (2020). Ele se baseia na implementação original do BART-Large, que possui 400 milhões de parâmetros internos (LEWIS et al., 2019). O autor relatou que realizou o treinamento com 3 *epochs* (épocas). O número de épocas é um hiperparâmetro que define o número de vezes que o algoritmo de aprendizado processará o *dataset* completamente (BROWNLEE, 2018).

É importante salientar que JERNITE utilizou uma abordagem diferente para construir os documentos, mas para fins de comparação ambos os modelos serão expostos ao mesmo conjunto de testes (com *support documents* criados utilizando o Elasticsearch).

3.8.1.1 Detalhes da máquina

- **GPU**: JERNITE não especificou o modelo, apenas que foram utilizadas 4 GPUs de 16 GB de memória;

3.8.1.2 Hiperparâmetros

- $\mathrm{batch\_size} = 8$; corresponde a quantos *batches* serão processados por vez na GPU. Este parâmetro possui impacto apenas no tempo necessário para realizar o treinamento;
- $\mathrm{max\_input\_length} = 1024$; equivale ao número máximo de palavras consideradas pelo modelo no processamento da entrada;
- $\mathrm{learning\_rate} = 2e - 4$; parâmetro utilizado pelo algoritmo de otimização do treinamento;
- $\mathrm{num\_epochs} = 3$; como elucidado anteriormente, equivale ao número de vezes que o modelo processará todo o *dataset* de entrada completamente.

3.8.2 T5-Small

O modelo treinado nesta pesquisa, como dito anteriormente, baseia-se na implementação do T5-Small (RAFFEL et al., 2019). É uma implementação com apenas 60 milhões de parâmetros internos. Dado o tamanho menor do modelo, optou-se por realizar o treinamento por um número maior de *epochs*: 10 épocas.

3.8.2.1 Detalhes da máquina

- **GPU**: 1x NVIDIA Tesla P100 16GB;

3.8.2.2 Hiperparâmetros

- $\text{batch\_size} = 4$
- $\text{max\_input\_length} = 512$
- $\text{learning\_rate} = 1e - 4$
- $\text{num\_epochs} = 5$

3.9 Avaliação dos modelos

Para conseguir fazer um comparativo das performances dos modelos, ambos foram avaliados conforme a qualidade de suas respostas dadas as perguntas do *split* de validação da base de dados ELI5. Ou seja, para cada registro do subconjunto mencionado foram geradas respostas pelos dois modelos. Foram fornecidos os mesmos parâmetros para o método $.\text{generate}()$ das implementações do BART e T5-Small. Tal método corresponde à implementação do algoritmo de geração da *Model Answer* de cada *Answer Generation Model*. Constatam-se os parâmetros utilizados na subseção 3.9.1.

3.9.1 Parâmetros de Geração de Respostas

- $\text{num\_answers} = 1$; corresponde ao número de respostas que o modelo deve gerar;
- $\text{num\_beams} = 1$; é um parâmetro para realização de *beam search*, para fins deste estudo não foi utilizado tal método;
- $\text{min\_len} = 96$; corresponde ao tamanho mínimo da *Model Answer*;
- $\text{max\_len} = 256$; corresponde ao tamanho máximo da *Model Answer*;

Para realizar a avaliação são consideradas as 24512 perguntas do conjunto de validação da base de dados ELI5, seguindo as diretrizes de pesquisas sobre o mesmo *dataset*. Os modelos foram avaliados utilizando a métrica ROUGE, mais especificamente ROUGE-1, ROUGE-2 e ROUGE-L. Para cada uma delas, foram avaliados *precision*, *recall* e *F1 measure*.

A avaliação dos modelos tomou como base a resposta com maior *score* para cada pergunta (*User Answer*), para fins deste estudo esta foi considerada correta e passada como referência para os modelos. Os dois modelos foram avaliados utilizando o ambiente Google Colab Pro[16], tendo disponíveis os seguintes recursos:

- **GPU**: 1x NVIDIA Tesla P100 16GB;
- **RAM**: 27 GB;
- **HD**: 167 GB;
- **CPU**: 4x Intel(®) Xeon(®) CPU @ 2.20GHz.

O tempo de processamento necessário para gerar as métricas para cada modelo, considerando todo o conjunto de validação do ELI5 - e que todas as respostas já estejam geradas - é de aproximadamente 8 minutos.

[16]https://colab.research.google.com/

4 RESULTADOS

Para apresentação e análise dos resultados desta pesquisa, primeiramente serão apresentadas as métricas coletadas através da avaliação dos modelos BART e T5-Small, seguindo de uma posterior comparação entre os resultados obtidos e os considerados como estado da arte para o *dataset*, e consecutivamente uma discussão dos possíveis motivos para a performance observada. Por fim, serão abordadas algumas limitações do presente estudo.

Retomando a base de dados utilizada nesta pesquisa, foi escolhido como principal *dataset* o ELI5, composto por perguntas não factuais e de domínio aberto retiradas da plataforma Reddit, mais especificamente do *subreddit Explain Like I'm Five*. Esse *dataset* foi dividido em 3 subconjuntos: treino, utilizado na etapa de *fine-tuning* dos modelos; teste, utilizado para coletar métricas internas durante a fase de treinamento; e validação, utilizado para avaliação dos modelos e coleta das métricas automáticas baseadas no método ROUGE.

4.1 Avaliação Quantitativa dos Modelos

Seguindo a metodologia proposta, foram geradas respostas para todo o *split* de validação do ELI5. Partindo delas, foram coletadas as 3 métricas utilizadas neste estudo: ROUGE-1, ROUGE-2 e ROUGE-L. Os resultados podem ser conferidos nas tabelas 4.1 e 4.2.

Tabela 4.1: Avaliação quantitativa da performance do modelo BART

	ROUGE-1	**ROUGE-2**	**ROUGE-L**
P	0.3635	0.0616	0.3322
R	0.2608	0.0519	0.2373
F1	0.2743	0.0478	0.2497

Fonte: O Autor

Tabela 4.2: Avaliação quantitativa da performance do modelo T5-Small

	ROUGE-1	**ROUGE-2**	**ROUGE-L**
P	0.3345	0.0575	0.3038
R	0.2483	0.0507	0.2248
F1	0.2572	0.0456	0.2329

Fonte: O Autor

Observa-se, ao analisar as tabelas 4.1 e 4.2, que o modelo BART mostrou resultados superiores em todas as métricas analisadas quando comparado ao T5-Small. Con-

siderando modelos da literatura, ambos tiveram resultados piores do que o estado da arte reportado para o *dataset*. O modelo que melhor performou até hoje foi uma implementação do BART, que atingiu ROUGE-1 F1: 30.5, ROUGE-2 F1: 6.2 e ROUGE-L F1: 24.3 (LEWIS et al., 2019). O modelo em questão utilizou-se da mesma fonte de conhecimento proposta originalmente por Fan et al. (2019), a plataforma Common Crawl. Os modelos com melhores avaliações até a atualidade no ELI5 podem ser observados na tabela 4.3.

Acreditamos que uma das principais razões pelas quais os modelos performam pior que o estado da arte na base de dados ELI5 é o uso da base de dados *wiki_snippets*. Visto que utilizamos um método de IR que se baseia na ocorrência de palavras iguais das perguntas em trechos de artigos da Wikipedia, nem sempre estes trechos terão relação com a pergunta original, como observa-se no Apêndice A - Exemplo de Support Document. Um comparativo das implementações utilizadas nesta pesquisa e o estado da arte pode ser conferido na tabela 4.4

Tabela 4.3: Estado da arte para o *dataset* ELI5

	ROUGE-1	**ROUGE-2**	**ROUGE-L**
Modelo Extrativo	0.235	0.031	0.175
Language Model (LM)	0.278	0.047	0.231
Seq2Seq	0.283	0.051	0.228
Seq2Seq Multi-task	0.289	0.054	0.231
BART	0.306	0.062	0.243

Fonte: Adaptado de Fan et al. (2019)

Tabela 4.4: Comparativo do estado da arte com resultados desta pesquisa

Modelo	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**
BART (LEWIS et al., 2019)	**0.306**	**0.062**	0.243
Seq2Seq Multi-task (FAN et al., 2019)	0.289	0.054	0.231
Seq2Seq (FAN et al., 2019)	0.283	0.051	0.228
Language Modelling (FAN et al., 2019)	0.278	0.047	0.231
BART (esta pesquisa)	0.274	0.048	**0.250**
T5-Small (esta pesquisa)	0.257	0.046	0.233
Modelo Extrativo (FAN et al., 2019)	0.235	0.030	0.170

Fonte: O autor

4.1.1 Tempos de treinamento

Como pode ser observado na tabela 4.5, o modelo T5-Small, apesar de treinado em uma máquina com 25% da capacidade de VRAM em comparação com o BART, consegue processar uma época inteira do *dataset* com um tempo 6.92 vezes menor. Isto ocorre

devido a diferença de tamanhos das redes neurais que compõem estes modelos, mas é uma característica positiva do modelo T5-Small, considerando que o mesmo poderia ser treinado por muito mais épocas durante o mesmo período de tempo.

Tabela 4.5: Tempos de treinamento dos modelos BART e T5-Small

Modelo	**Tempo de treinamento (h)**	**Tempo por época (h)**
BART	54	18
T5-Small	13	2.6

Fonte: O Autor

4.2 Discussão

Em seu trabalho original, Jernite (2020) utilizou uma abordagem diferente para o componente de IR. O autor propôs a utilização de um modelo do estilo BERT para fazer a seleção dos trechos de artigos da Wikipedia. Em comparação com a abordagem utilizada neste estudo, o autor atingiu um ROUGE-1 F1 pior (diferença de 0.0018), porém com melhores ROUGE-2 F1 e ROUGE-L (diferenças de 0.0073 e 0,0086, respectivamente). Nota-se que todas as diferenças foram na ordem de 10^{-3}, um indicativo de que os dois métodos diferentes de IR produziram resultados semelhantes.

Esta pesquisa apresenta, portanto, a primeira avaliação quantitativa de desempenho de um dos modelos da categoria T5 para o *dataset* ELI5, considerando os conjuntos de validação e testes propostos por seus autores. Apesar de ter uma performance inferior ao BART, o modelo T5-Small é o menor modelo disponível da família T5. De acordo com Raffel et al. (2019), os modelos maiores performam melhor, porém possuem um custo computacional bem mais elevado. Portanto, pode-se sugerir que resultados melhores sejam alcançados por variações maiores do T5, como o T5-Large, ou até mesmo o T5-11B. Petroni et al. (2020) testaram a performance do modelo T5-Base no *dataset*, porém como foram utilizados conjuntos diferentes para teste, não cabe a comparação.

Apesar de proposto em 2019, o ELI5 permanece como um dos *datasets* mais desafiadores da área de LFQA. Segundo LEWIS et al., o grande motivo é porque as respostas não são fortemente definidas pela pergunta, dada a natureza abstrata das questões desta base de dados (2019).

4.3 Limitações

Uma das grandes limitações para a avaliação de modelos de NLG é escolher um conjunto de métricas adequado. Apesar da extensa utilização do método ROUGE na literatura, o mesmo não consegue considerar sinônimos, por exemplo. Uma opção para sanar isto é utilizar a avaliação feita por seres humanos, mas conforme explorado por Krishna, Roy and Iyyer (2021), dada a variedade de tópicos presente no *dataset* e a extensão das respostas, avaliadores precisariam consultar outras fontes de conhecimento para averiguar a corretude das respostas, o que dificulta a tarefa. A maioria dos avaliadores sugeriu a utilização de especialistas, o que torna inviável a avaliação de conjuntos de perguntas e respostas muito grandes e diversos.

Outra clara limitação do método utilizado neste trabalho é o módulo de IR, visto que a base de conhecimentos utilizada para gerar material auxiliar aos modelos é a Wikipedia. Não é possível garantir que a resposta para a pergunta esteja no conjunto de trechos recuperados pelo módulo, o que faz com que os modelos baseiem-se apenas em conhecimento prévio para responder perguntas quando isso ocorre. Como trabalho futuro, talvez seja benéfico recuperar informações concomitantemente de outras bases de conhecimento para providenciar mais contexto aos modelos.

5 CONCLUSÃO

Como visto anteriormente, do ponto de vista de NLG, o problema LFQA de tomar como entrada perguntas e documentos de suporte e gerar respostas de maneira abstrativa (ou seja, gerar um texto novo) é considerado uma tarefa do tipo *sequence-to-sequence*. O presente estudo se propôs a realizar experimentos empíricos, coletando métricas automatizadas, através da aplicação de modelos pré-treinados considerados o estado da arte para problemas do tipo *sequence-to-sequence*.

Para atingir esses objetivos, foram utilizados dois modelos diferentes baseados na arquitetura de *Transformers*: BART e T5-Small. Os modelos foram treinados sobre a base de dados ELI5, e comparados utilizando as mesmas métricas propostas por seus autores (FAN et al., 2019). Este trabalho foi pioneiro em reportar as métricas de performance de um dos modelos do tipo T5, considerando as mesmas condições de avaliação propostas pelos autores do *dataset*.

Apesar dos resultados negativos encontrados comparando os modelos, encoraja-se que versões mais robustas do T5 sejam testadas, pois os indicativos são de que tais modelos performariam melhor em tarefas semelhantes comparados à versão utilizada neste trabalho. Entretando, para fins deste estudo, a escolha do modelo T5-Small justifica-se do ponto de vista dos recursos computacionais necessários para o seu treinamento.

Portanto, recomendamos como pesquisas futuras explorar a utilização de diferentes bases de conhecimento para a criação dos documentos de suporte para os modelos, e também a utilização de versões mais robustas do modelo T5.

REFERÊNCIAS

ALLAM, A. M. N.; HAGGAG, M. H. The question answering systems: A survey. **International Journal of Research and Reviews in Information Sciences (IJRRIS)**, v. 2, n. 3, 2012.

ANDERSON, K. E. Ask me anything: what is reddit? **Library Hi Tech News**, Emerald Group Publishing Limited, 2015.

BROWN, T. B. et al. **Language Models are Few-Shot Learners**. arXiv, 2020. Available from Internet: <https://arxiv.org/abs/2005.14165>.

BROWNLEE, J. What is the difference between a batch and an epoch in a neural network. **Machine Learning Mastery**, v. 20, 2018.

BUI, M.-Q. et al. How state-of-the-art models can deal with long-form question answering. In: **Proceedings of the 34th Pacific Asia Conference on Language, Information and Computation**. [S.l.: s.n.], 2020. p. 375–382.

Calijorne Soares, M. A.; PARREIRAS, F. S. A literature review on question answering techniques, paradigms and systems. **Journal of King Saud University - Computer and Information Sciences**, v. 32, n. 6, p. 635–646, 2020. ISSN 1319-1578. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S131915781830082X>.

CAMBRIA, E.; WHITE, B. Jumping nlp curves: A review of natural language processing research [review article]. **IEEE Computational Intelligence Magazine**, v. 9, n. 2, p. 48–57, 2014.

CHEN, W. et al. Logical natural language generation from open-domain tables. In: **Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics**. Online: Association for Computational Linguistics, 2020. p. 7929–7942. Available from Internet: <https://aclanthology.org/2020.acl-main.708>.

COHEN, D.; YANG, L.; CROFT, W. B. Wikipassageqa: A benchmark collection for research on non-factoid answer passage retrieval. In: **The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval**. New York, NY, USA: Association for Computing Machinery, 2018. (SIGIR '18), p. 1165–1168. ISBN 9781450356572. Available from Internet: <https://doi.org/10.1145/3209978.3210118>.

DAI, A. M.; LE, Q. V. Semi-supervised sequence learning. In: CORTES, C. et al. (Ed.). **Advances in Neural Information Processing Systems**. Curran Associates, Inc., 2015. v. 28. Available from Internet: <https://proceedings.neurips.cc/paper/2015/file/7137debd45ae4d0ab9aa953017286b20-Paper.pdf>.

DAI, Z. et al. Transformer-xl: Attentive language models beyond a fixed-length context. **CoRR**, abs/1901.02860, 2019. Available from Internet: <http://arxiv.org/abs/1901.02860>.

DEVLIN, J. et al. **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding**. arXiv, 2018. Available from Internet: <https://arxiv.org/abs/1810.04805>.

DIMITRAKIS, E.; SGONTZOS, K.; TZITZIKAS, Y. A survey on question answering systems over linked data and documents. **Journal of Intelligent Information Systems**, v. 55, n. 2, p. 233–259, Oct 2020. ISSN 1573-7675. Available from Internet: <https://doi.org/10.1007/s10844-019-00584-7>.

FAN, A. et al. ELI5: long form question answering. **CoRR**, abs/1907.09190, 2019. Available from Internet: <http://arxiv.org/abs/1907.09190>.

GATT, A.; KRAHMER, E. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. **Journal of Artificial Intelligence Research**, v. 61, p. 65–170, 2018.

GEHRMANN, S. et al. The GEM benchmark: Natural language generation, its evaluation and metrics. In: **Proceedings of the 1st Workshop on Natural Language Generation, Evaluation, and Metrics (GEM 2021)**. Online: Association for Computational Linguistics, 2021. p. 96–120. Available from Internet: <https://aclanthology.org/2021.gem-1.10>.

HASHEMI, H. et al. Antique: A non-factoid question answering benchmark. In: JOSE, J. M. et al. (Ed.). **Advances in Information Retrieval**. Cham: Springer International Publishing, 2020. p. 166–173. ISBN 978-3-030-45442-5.

HIRSCHMAN, L.; GAIZAUSKAS, R. Natural language question answering: the view from here. **natural language engineering**, Cambridge University Press, v. 7, n. 4, p. 275–300, 2001.

JERNITE, Y. **Explain Anything Like I'm Five: A Model for Open Domain Long Form Question Answering**. 2020. Available from Internet: <https://yjernite.github.io/lfqa.html>.

KHUSHHAL, S. et al. Question retrieval using combined queries in community question answering. **Journal of Intelligent Information Systems**, v. 55, n. 2, p. 307–327, Oct 2020. ISSN 1573-7675. Available from Internet: <https://doi.org/10.1007/s10844-020-00612-x>.

KRISHNA, K.; ROY, A.; IYYER, M. Hurdles to progress in long-form question answering. **CoRR**, abs/2103.06332, 2021. Available from Internet: <https://arxiv.org/abs/2103.06332>.

KUMAR, E. **Natural language processing**. [S.l.]: IK International Pvt Ltd, 2013. 1–23 p.

LEWIS, M. et al. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. **CoRR**, abs/1910.13461, 2019. Available from Internet: <http://arxiv.org/abs/1910.13461>.

LIN, C.-Y. Rouge: a package for automatic evaluation of summaries. In: **Workshop on Text Summarization Branches Out, Post-Conference Workshop of ACL 2004, Barcelona, Spain**. [s.n.], 2004. p. 74–81. Available from Internet: <https://www.microsoft.com/en-us/research/publication/rouge-a-package-for-automatic-evaluation-of-summaries/>.

LIU, D. et al. Rikinet: Reading wikipedia pages for natural question answering. **CoRR**, abs/2004.14560, 2020. Available from Internet: <https://arxiv.org/abs/2004.14560>.

LIU, Y. et al. **RoBERTa: A Robustly Optimized BERT Pretraining Approach**. arXiv, 2019. Available from Internet: <https://arxiv.org/abs/1907.11692>.

MADIAJAGAN, M.; RAJ, S. S. Chapter 1 - parallel computing, graphics processing unit (gpu) and new hardware for deep learning in computational intelligence research. In: SANGAIAH, A. K. (Ed.). **Deep Learning and Parallel Computing Environment for Bioengineering Systems**. Academic Press, 2019. p. 1–15. ISBN 978-0-12-816718-2. Available from Internet: <https://www.sciencedirect.com/science/article/pii/B9780128167182000087>.

MCDONALD, D. D. Natural language generation. **Handbook of Natural Language Processing**, v. 2, p. 121–144, 2010.

NORASET, T.; LOWPHANSIRIKUL, L.; TUAROB, S. Wabiqa: A wikipedia-based thai question-answering system. **Information Processing and Management**, v. 58, n. 1, p. 102431, 2021. ISSN 0306-4573. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S0306457320309249>.

PETERS, M. E. et al. Deep contextualized word representations. In: **Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)**. New Orleans, Louisiana: Association for Computational Linguistics, 2018. p. 2227–2237. Available from Internet: <https://aclanthology.org/N18-1202>.

PETRONI, F. et al. KILT: a benchmark for knowledge intensive language tasks. **CoRR**, abs/2009.02252, 2020. Available from Internet: <https://arxiv.org/abs/2009.02252>.

RAFFEL, C. et al. **Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer**. arXiv, 2019. Available from Internet: <https://arxiv.org/abs/1910.10683>.

REITER, E.; DALE, R. Building applied natural language generation systems. **Natural Language Engineering**, Cambridge University Press, v. 3, n. 1, p. 57–87, 1997.

SPECIA, L.; SCARTON, C.; PAETZOLD, G. Quality estimation for machine translation. **Synthesis Lectures on Human Language Technologies**, v. 11, p. 1–162, 09 2018.

TURING, A. M. Computing machinery and intelligence. **Mind**, Oxford University Press on behalf of the Mind Association, v. 59, n. 236, p. 433–460, 1950. ISSN 00264423. Available from Internet: <http://www.jstor.org/stable/2251299>.

VASWANI, A. et al. **Attention Is All You Need**. arXiv, 2017. Available from Internet: <https://arxiv.org/abs/1706.03762>.

WARSTADT, A.; SINGH, A.; BOWMAN, S. R. Neural network acceptability judgments. **CoRR**, abs/1805.12471, 2018. Available from Internet: <http://arxiv.org/abs/1805.12471>.

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. Available from Internet: <https://aclanthology.org/2020.emnlp-demos.6>.

YANG, M. et al. Investigating the transferring capability of capsule networks for text classification. **Neural Networks**, v. 118, p. 247–261, 2019. ISSN 0893-6080. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S089360801930187X>.

YOGISH, D.; MANJUNATH, T. N.; HEGADI, R. S. Survey on trends and methods of an intelligent answering system. In: **2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT)**. [S.l.: s.n.], 2017. p. 346–353.

ZHU, M. et al. Question answering with long multiple-span answers. In: **Findings of the Association for Computational Linguistics: EMNLP 2020**. Online: Association for Computational Linguistics, 2020. p. 3840–3849. Available from Internet: <https://aclanthology.org/2020.findings-emnlp.342>.

APÊNDICE A - EXEMPLO DE SUPPORT DOCUMENT

Questão passada como entrada para o *Elasticsearch*: ***"Why do we have freckles?"***

A saída está ordenada pelo ranqueamento atribuído pelo próprio algoritmo do *Elasticsearch*, e os itens podem ser vistos como tuplas $(\text{article\_title} : \text{passage\_text})$, definidos na seção 3.3. Observa-se que nem todos os trechos possuem o assunto relacionado com a resposta, o que é comum quando utiliza-se o método *bag-of-words*, que não consegue representar a semântica das palavras nas frases.

O *suport document* gerado através do processamento desses registros consiste da concatenação dos passage_text, e pode ser inferido pelo leitor.

1. *Freckles (novel)*: *enthralled with the Limberlost, is impressed by the boy's polite assertiveness and hires him despite his youth and disability. He gives his name only as "Freckles", insisting that he has no name of his own. He claims the name given him in the orphanage (which we never learn) "is no more my name than it is yours". Freckles asks McLean to choose a name for him to put down on the books. McLean gives Freckles the name of his own father, James Ross McLean. Freckles' duty is to twice a day walk the perimeter of the lumber company's land, a*
2. *Freckle*: *and more commonly found on children before puberty. Upon exposure to the sun, freckles will reappear if they have been altered with creams or lasers and not protected from the sun, but do fade with age in some cases. Freckles are not a skin disorder, but people with freckles generally have a lower concentration of photo-protective melanin, and are therefore more susceptible to the harmful effects of UV radiation. It is suggested that people whose skin tends to freckle should avoid overexposure to sun and use sunscreen. Genetics The presence of freckles is related to rare alleles of the MC1R gene, though*
3. *Freckle*: *it does not differentiate whether an individual will have freckles if they have one or even two copies of this gene. Also, individuals with no copies of the MC1R do sometimes display freckles. Even so, individuals with a high number of freckling sites have one or more of variants of the MC1R gene. Of the variants of the MC1R gene Arg151Cys, Arg160Trp, and Asp294His are the most common in the freckled subjects. The MC1R gene is also associated with red hair more strongly than with freckles. Most red-haired individuals have two variants of the MC1R gene and almost all*
4. *Freckle*: *Freckle Biology The formation of freckles is caused by exposure to sunlight.*

The exposure to UV-B radiation activates melanocytes to increase melanin production, which can cause freckles to become darker and more visible. This means that one who has never developed freckles may develop them suddenly following extended exposure to sunlight. Freckles are predominantly found on the face, although they may appear on any skin exposed to the sun, such as arms or shoulders. Heavily distributed concentrations of melanin may cause freckles to multiply and cover an entire area of skin, such as the face. Freckles are rare on infants,

5. *Freckles (novel)*: *the thistle-patch". She promises that she will find his parents and prove that Freckles comes from "a race of men that have been gentlemen for ages, and couldn't be anything else."Her inquiries at his former orphanage lead her to Lord and Lady O'More, Irish nobility who have been searching Chicago for Lord O'More's lost nephew. They prove themselves to be kind and noble, and explain that Freckles' father had been disinherited when he married a clergyman's daughter, and both had perished in the fire that took his hand. Freckles' true name is Terence Maxwell O'More of Dunderry House in*
6. *Freckles Comes Home*: *Freckles Comes Home Plot summary Freckles Winslow (Johnny Downs) is on his way home from college. On the bus he encounters a crook, "Muggsy"Dolan who calls himself Jack Leach (Walter Sande). Jack is on the run from the law, and is looking for a safe place to hide. The two men come to talking and Freckles mentions his serene home town to Jack, having only good things to say about it. Jack decides to tag along and take his refuge in Freckles home town. With Freckles help he gets to stay at the local hotel, owned by Danny Doyle*
7. *Freckles (1935 film)*: *Freckles (1935 film) Plot Freckles, a young man and orphan, shows up at a lumber camp, where the local schoolteacher, Mary Arden, takes a shine to him and convinces the lumber company's owner, McLean, to hire Freckles as a guard. While working there, Freckles begins a relationship with Mary, while Laurie-Lou Duncan, a precocious young girl also befriends Freckles and helps him learn more about the forest and the plants it contains. Laurie-Lou has a pet bear cub, and one day when the cub is in danger of being injured by a tree about to be felled by the lumberjacks,*
8. *Freckle Juice*: *Freckle Juice Summary Nicky Lane at school had freckles; but Andrew Marcus did not. Andrew desperately wants to have freckles just like Nicky, so that his mother will never notice when his neck is dirty. He buys a recipe from a*

classmate named Sharon for a horrible concoction called freckle juice that is supposed to cause a person to get freckles. It just makes him sick, however, causing his mother to think he has appendicitis. He stays home the following day and quickly recovers. But before that, in the following morning, when his mother sings, "Rise and shine!"(and takes

9. *Champagne gene*: *is "pink with abundant dark freckles, except under white markings."The freckles - not mottles, splotches, specks, or blotches - are dark and may have a purple cast, and are small and numerous. By comparison, mottling associated with the Leopard complex is large and blocky alternations between true black or charcoal-colored skin and pigmentless-pink skin. The skin of cremellos and perlinos is pigmented-pink, and exhibit a scant few tiny black specks. Homozygous pearls and pearl-cream combinations also exhibit some freckling of the skin, however this is muted in comparison to the freckles of champagne skin. The skin may exhibit an*
10. *Lentigo*: *Lentigo Differential diagnosis Lentigines are distinguished from freckles (ephelis) based on the proliferation of melanocytes. Freckles have a relatively normal number of melanocytes but an increased amount of melanin. A lentigo has an increased number of melanocytes. Freckles will increase in number and darkness with sunlight exposure, whereas lentigines will stay stable in their color regardless of sunlight exposure. Treatment Lentigines by themselves are benign, however one might desire the removal or treatment of some of them for cosmetic purposes. In this case they can be removed surgically, or lightened with the use of topical depigmentation agents. Some common depigmentation*

APÊNDICE B - RESPOSTAS GERADAS PELOS MODELOS

Neste apêndice serão exploradas as gerações de respostas dos modelos BART e T5 a partir da pergunta: "*How does a flush toilet work?*". Será apresentada a resposta escolhida como correta na base de dados, seguido da produzida pelo modelo BART e, posteriormente, a resposta produzida pelo modelo T5-Small.

- **Resposta de referência**

1. When you depress the handle, it raises a flush lever inside the tank.

2. When flush lever is raised, it allows the tank water to quickly drain through an opened flapper down into the bowl. This causes the flushing of the bowl.

3. There is float (or plastic balloon) inside the tank. The float on the fill valve will also drop with the tank's declining water to allow more water into the tank, arriving from the water supply hose connected beneath the toilet.

4. Step 3 fills the tank again and when the rising water level increases the float rises and closes the valve stopping new water into the tank. This completes the flush cycle.

- **Resposta gerada pelo modelo BART**

A flush toilet uses a combination of gravity and gravity assisted water pressure to push the water through a pipe. The water is then used to flush the toilet. A dry toilet uses gravity assisted gravity to push water through the pipe. This is done by a pump that pushes water through an opening in the wall of the toilet, and then the water is pushed through a valve that opens up to allow the water to flow through the opening. The valve is connected to a pump in the toilet that pushes the water down the pipe, and the water flows out the drain.

- **Resposta gerada pelo modelo T5-Small**

The water in the toilet is poured into the bowl. The water is pumped into the toilet, and the water is then poured through the bowl, and then pumped through the toilet. Then the water goes through the drain, and it is pushed into the water. The cistern is a small cylindre, which is absorbed by the water, and absorbed into the drain. The drain is filtered through the water and poured out. The flushing is essentially a crystal. The pipe is sprayed with a bucket of water, which then spits the water out of the bowl and squeezing it out. This is why the water in a toilet is not filtered out, so it is not the same as the water that is thrown into the bucket.

APÊNDICE C - AMBIENTE COMPUTACIONAL

Todo código desenvolvido para esta pesquisa foi elaborado utilizando a linguagem Python, com versão superior a 3.9.0. Ele se encontra disponível no diretório do GitHub do autor[1]. De modo a facilitar a replicação, a porção de código que roda localmente foi feita utilizando Docker[2] e também será disponibilizada no mesmo repositório.

Os módulos externos utilizados foram: Datasets[3]; Transformers[4]; Elasticsearch[5]; e Rouge[6].

Durante o desenvolvimento do presente estudo foram utilizados dois ambientes computacionais distintos, um para a geração dos documentos de suporte (Máquina 1) e outro para o *fine-tuning* com posterior avaliação dos modelos BART e T5-Small (Máquina 2).

Configurações da Máquina 1

- **GPU**: 1x NVIDIA GTX 1060 6GB;
- **RAM**: 16 GB;
- **HD**: SSD Sata III 512 GB;
- **CPU**: 1x Intel®Core™i5-7400 @ 3.0GHz.

Configurações da Máquina 2

- **GPU**: 1x NVIDIA Tesla P100 16GB;
- **RAM**: 27 GB;
- **HD**: 167 GB;
- **CPU**: 4x Intel®Xeon®CPU @ 2.20GHz.

[1]https://github.com/GabrielPakulski/ELI5-experiment
[2]https://www.docker.com/
[3]https://pypi.org/project/datasets/
[4]https://pypi.org/project/transformers/
[5]https://elasticsearch-py.readthedocs.io/en/v8.4.2/
[6]https://pypi.org/project/rouge/