UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

Leonardo Schlüter leite

Recuperação de informação em documentos baseada em consultas de linguagem natural utilizando *Machine Learning*

Florianópolis - SC
2023

Leite, Leonardo Schlüter
Recuperação de informação em documentos baseada em consultas de linguagem natural utilizando Machine Learning / Leonardo Schlüter Leite ; orientador, Elder Rizzon Santos, 2023.
68 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Ciências da Computação, Florianópolis, 2023.

Inclui referências.

1. Ciências da Computação. 2. Inteligência Artificial. 3. Aprendizagem de Máquina. 4. Processamento de Linguagem Natural. 5. Extração de informação. I. Santos, Elder Rizzon. II. Universidade Federal de Santa Catarina. Graduação em Ciências da Computação. III. Título.

Leonardo Schlüter Leite

Recuperação de informação em documentos baseada em consultas de linguagem natural utilizando *Machine Learning*

Trabalho Conclusão do Curso de Graduação em Ciência da Computação do Departamento de Informática e Estatística da Universidade Federal de Santa Catarina, como requisito para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Elder Rizzon Santos, Dr.

Florianópolis - SC

2023

Leonardo Schlüter Leite

Recuperação de informação em documentos baseada em consultas de linguagem natural utilizando *Machine Learning*

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do título de Bacharel e aprovado em sua forma final pelo Curso de Ciência da Computação.

Local [inserir local da defesa], [dia] de [mês] de [ano].

_____

Coordenação do Curso
Banca examinadora

_____

Prof. Elder Rizzon Santos, Dr.
Orientador

_____

Prof. Ronaldo dos Santos Mello, Dr.
Universidade Federal de Santa Catarina

_____

Prof. Thiago Angelo Gelaim, Dr.
Universidade Federal de Santa Catarina

Florianópolis - SC, 2023.

Dedico este trabalho a minha família, sem eles nada disso teria sequer começado!

AGRADECIMENTOS

Primeiramente agradeço a minha mãe, Magali Schlüter Leite, que foi minha maior incentivadora e foi quem nunca me deixou desistir deste curso, por mais difícil que tenha sido em diversos momentos.

Em segundo, agradeço ao ensino público, gratuito e de qualidade que recebi durante toda a minha vida, desde o ensino básico até o ensino superior. Sem as oportunidades dadas por instituições públicas nada disso seria possível.

Por fim, mas não menos importante, agradeço a minha esposa, Angella Monteiro Santiago, que foi quem me apoiou e me fez enxergar a luz no fim do túnel, quando acabar este curso parecia impossível.

RESUMO

O projeto desenvolvido trata da recuperação de arquivos e da extração de texto baseado em uma pergunta de texto natural. Para realizar tal desafio, conceitos e técnicas da área de Processamento de Linguagem Natural (PLN) foram empregados. O projeto surgiu para resolver problemas de estudantes com dificuldade de lidar com vários textos de diversas disciplinas ao mesmo tempo, mas as aplicações são as mais diversas. O ajuste fino, essencial para melhorar o desempenho, foi uma tarefa complexa devido às peculiaridades de cada conjunto de dados. A abordagem escolhida, embora desafiadora devido a recursos computacionais limitados, resultou em modelos mais generalistas. Ao concluir o projeto, foi adquirido um entendimento abrangente do processamento de linguagem natural, bem como fundamentos teóricos e conceitos relevantes em áreas correlatas. Apesar dos desafios, este trabalho demonstrou que o ajuste fino em múltiplos conjuntos de dados pode levar a melhorias significativas no desempenho dos modelos de PLN em tarefas de questionamento, proporcionando resultados superiores a várias abordagens relacionadas.

Palavras-chaves: Recuperação de documentos; extração textual; processamento de linguagem natural; aprendizagem de máquina; inteligência artificial.

ABSTRACT

The developed project focuses on file retrieval and text extraction based on natural language queries, utilizing concepts and techniques from the field of Natural Language Processing (NLP). Its primary aim was to assist students in handling multiple texts from diverse subjects simultaneously, although its applications extend beyond education. Fine-tuning, crucial for enhancing performance, proved to be a complex task due to the unique aspects of each *dataset*. The selected approach, despite challenges arising from limited computational resources, resulted in the creation of more generalized models. Upon completion, the project yielded a comprehensive understanding of natural language processing, encompassing theoretical foundations and pertinent concepts from related areas. In spite of the encountered difficulties, this work showcased that fine-tuning across multiple *dataset*s could substantially enhance the performance of NLP models in question-answering tasks, surpassing several related approaches.

Keywords: Information Retrieval; Text Extraction; Natural Language Processing (NLP); Machine Learning; Artificial Intelligence.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIAÇÕES E REDUÇÕES

ANNs	*Artificial Neural Networks*
API	*Application Programming Interface*
BERT	*Bidirectional Encoder Representations from Transformers*
CNNs	*Convolutional Neural Networks*
DL	*Deep Learning*
GB	Gigabytes
GPUs	*Graphical Processing Units*
AI	*Artificial Intelligence*
ICT	*Inverse Cloaz Task*
IDF	*Inverse Document Frequency*
IR	*Information Retrieval*
KAE	*Knowledge-Augmented Encoder*
MLM	*Masked Language Model*
NKR	*Neural Knowledge Retriever*
NLP	*Natural Language Processing*
NQ	*NaturalQuestions*
NSP	*Next Sentence Prediction*
OpenQA	*Open Domain Question Answering*
ORQA	*Open-Retrieval Question Answering*
PLN	Processamento em Linguagem Natural
QA	*Question Answering*
RAM	*Random Access Memory*
RC	*Reading Comprehension*
REALM	*Retrieval-Augmented Language Model Pre-Training*
ANNs	*Artificial Neural Networks*
RNNs	*Recurrent Neural Networks*
TF-IDF	*Term Frequency - Inverse Document Frequency*

SUMÁRIO

1 INTRODUÇÃO

Nos dias de hoje, a maioria de nós tem acesso a diversos tipos de tecnologias. Muitas vezes, não percebemos a quantidade de ferramentas que utilizamos para agilizar tarefas, facilitar rotinas ou até mesmo resolver problemas complexos com apenas alguns cliques. Esse avanço tecnológico só foi possível graças ao progresso de várias áreas da Ciências da Computação, entre elas a Inteligência Artificial (IA). Embora a ideia de IA não seja recente, sua atenção e evolução ganharam destaque especialmente no último século.

Uma área em crescimento nos últimos anos é a de Processamento de Linguagem Natural (PLN, ou *Natural Language Processing* – NLP). O PLN abrange uma variedade de tarefas relacionadas à compreensão da linguagem humana (Otter; Medina; Karina, 2021). Muitas das aplicações do PLN têm impacto direto em problemas cotidianos, como o uso da arquitetura *Bidirectional Encoder Representations from Transformers* (BERT) em uma variedade de tarefas de NLP (Devlin *et al.*, 2019), incluindo dúvidas na hora de estudar e tradução de textos.

De acordo com Otter, Medina e Karina (2021), o crescimento do PLN está diretamente relacionado à evolução da área de *Deep Learning* (DL). O DL refere-se à definição de modelos de redes neurais artificiais (ou *Artificial Neural Networks* - ANNs) com centenas de milhões, e até mesmo bilhões, de parâmetros treináveis. Esse tipo de rede neural tornou-se possível devido ao aumento do poder de processamento dos computadores em geral.

Uma das aplicações do PLN é a recuperação de informações em um conjunto de documentos, também conhecida como *Information Retrieval* (IR). Resumidamente, essa aplicação consiste em encontrar, em um conjunto de documentos, aqueles que melhor se adequam a uma consulta feita em linguagem natural (Kenter *et al.*, 2017). Muitas abordagens de IR atribuem uma classificação numérica a cada documento com base na consulta em linguagem natural.

Outra aplicação do PLN é a construção de respostas textuais para perguntas semânticas, conhecida como *Question Answering* (QA). Essa subárea utiliza-se de métodos de sumarização e extração de informações (Otter, Medina e Karina, 2021), que também são aplicações do PLN. A sumarização busca resumir o conteúdo de um documento textual sem perder a expressividade semântica.

No entanto, uma área que vem chamando atenção nas aplicações do PLN é aquela capaz de "interpretar" o conteúdo de documentos para responder às consultas do usuário final. Na literatura, já existem tecnologias capazes de ler, interpretar, criar sentenças e até mesmo gerar perguntas sobre arquivos de texto (exemplos: Otter; Marina; Karina, 2021; Burke *et al.*, 1997; Devlin *et al.*, 2019; Nishida *et al.*, 2018). Assim, surge a pergunta que direciona esta pesquisa: será possível criar um sistema que una e coordene as mais recentes tecnologias para recuperar documentos de um repositório com base em consultas em linguagem natural para um usuário comum?

Dessa forma, esta pesquisa tem como objetivo desenvolver um protótipo de um sistema que utilize o PLN como ferramenta principal. Para isso, são empregadas técnicas de PLN em diferentes processos. Todos os textos considerados são escritos na língua inglesa. O primeiro é a capacidade de ler documentos de um repositório e construir representações com informações complementares de seus conteúdos, utilizando extração de informações como base de dados para consultas em linguagem natural. O segundo processo é a interpretação de uma consulta em linguagem natural, enquadrando-se na aplicação de QA. Por fim, há o mapeamento entre a consulta e o conteúdo representado, com o objetivo de criar uma sentença de resposta ou retornar documentos de possível interesse, resolvido por meio da aplicação de QA. Portanto, o problema consiste em utilizar diversos tipos de aplicações de PLN de forma coordenada para construir uma resposta ou retornar um documento, com base em um conjunto de documentos e uma consulta em linguagem natural.

Posteriormente, os algoritmos selecionados são avaliados qualitativa e quantitativamente. Com os resultados em mãos, são realizadas iterações adicionais para aperfeiçoar os algoritmos ou avaliar novas abordagens. Por fim, são analisados os resultados obtidos, apontando possíveis deficiências nos métodos empregados e sugerindo possíveis caminhos futuros a serem explorados.

1.1 OBJETIVO GERAL

Propor um sistema com a capacidade de recuperação de documentos baseado em consultas formuladas em linguagem natural escrita.

1.2 OBJETIVOS ESPECÍFICOS

- Analisar o estado da arte em processamento de linguagem natural com foco em extração de texto e recuperação de informações;
- Analisar e selecionar algum sistema de repositório de arquivos para ser utilizado como base para o modelo que é proposto;
- Desenvolver um modelo visando a utilização de um ou mais algoritmo(s) de processamento de linguagem para a recuperação de informação de documentos contidos no repositório citado;
- Propor critérios quantitativos e qualitativos de validação do modelo desenvolvido e algoritmos utilizados;
- Realizar testes com o modelo e analisar os resultados de acordo com os critérios propostos.

1.3 METODOLOGIA

De forma a atingir o objetivo geral várias etapas precisam ser definidas e executadas. O ponto inicial é realizar uma pesquisa bibliográfica e análise do estado da arte sobre determinados temas, sendo eles o processamento de linguagem, a recuperação de informação e repositórios de documentos. Esta revisão proporcionará a capacidade de:

- Descrever o estado da arte de algoritmos de PLN;
- Identificar e documentar trabalhos com objetivos correlatos; e
- Identificar e descrever repositórios de arquivos que facilitem a integração com os algoritmos de PLN e o modelo que é proposto.

Para desenvolver um modelo visando a utilização de um ou mais algoritmo(s) de PLN para a recuperação de informação de documentos contidos num repositório; Para analisar e selecionar algum sistema de repositório de arquivos para ser utilizado como base para o modelo que é proposto, executar-se-ão as seguintes tarefas:

1. Esboçar uma arquitetura a partir do conhecimento adquirido, contendo as diferentes partes necessárias à realização do objetivo. Neste esboço deve-se definir possíveis requisitos para o sistema de arquivos.
2. Selecionar um repositório de arquivos que atenda aos requisitos contidos na arquitetura proposta.

3. Selecionar algoritmos de PLN que contemplem a arquitetura esquematizada.
4. Implementar algoritmos, bem como a arquitetura como um todo, utilizando tecnologias apresentadas nos trabalhos estudados.
5. Revisar o sistema e a arquitetura para evitar possíveis falhas e pontos fracos.

A fim de propor critérios quantitativos e qualitativos de validação do modelo desenvolvido e algoritmos utilizados, é realizada uma investigação dos métodos de validação desses algoritmos. Então a seleção de critérios é baseada na eficácia, tamanho do conjunto de dados necessários e facilidade de aplicação. Por fim, aplicam-se os critérios selecionados.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados conceitos utilizados ao longo do trabalho para conseguir realizar os objetivos descritos. Entende-se que os conceitos podem ter significados diferentes dependendo do seu contexto, por isso às vezes um contexto é apresentado, para evitar ambiguidade.

2.1 INTRODUÇÃO

Neste capítulo, é apresentada a fundamentação teórica para o desenvolvimento de um sistema de busca de arquivos em um repositório, capaz de responder perguntas factuais sobre os assuntos presentes nos arquivos. São discutidos os principais conceitos relacionados ao processamento de linguagem natural (PLN), modelos de linguagem pré-treinados, compreensão de perguntas e respostas, bem como técnicas de busca e recuperação de informações. Esses conceitos fornecerão a base teórica necessária para compreender e implementar o sistema proposto.

2.2 MÉTRICAS

Nesta seção são apresentadas as métricas utilizadas neste trabalho. Essas métricas tem o objetivo de permitir uma análise probabilística dos resultados e avaliar a performance do modelo em comparação com os trabalhos correlatos.

2.2.1 EM

Segundo Rajpurkar *et al.* (2016), o objetivo da métrica Exact Match (EM) é quantificar a porcentagem de predições que são exatamente iguais às respostas contidas nos conjuntos de dados. No contexto deste trabalho utilizamos a média do EM em todos os resultados dos testes.

2.4.3 **F1**

Essa métrica é utilizada para estimar a acurácia de um teste, baseando-se nas métricas de *recall* e precisão. *Recall* é o número de positivos verdadeiros divididos pela quantidade total de resultados positivos que existem nos dados de teste. Precisão é o número de positivos verdadeiros dividido pelo total de positivos que um dado teste retornou. Exemplo: um conjunto de dados com 10 exemplos, sendo 3 positivos e 7 falsos. O número de positivos retornados pelo teste é 2, sendo 1 verdadeiro positivo. A precisão seria $\frac{1}{2}$, enquanto o *recall* seria $\frac{1}{3}$.

A métrica F1 é a média harmônica de precisão e *recall*. Como explicado por Rajpurkar *et al.* (2016), no contexto de Question Answering, F1 é usada como a média da sobreposição entre os resultados do teste e a resposta correta, isso é possível, pois o texto da resposta original pode não ser exatamente igual ao texto retornado pelo teste. Então podemos tratar as respostas como um conjunto de *tokens* e calcular a semelhança usando a fórmula 2/(recall^-1)+(precisão^-1). Fazemos isso para cada exemplo do teste e depois calculamos a média do valor em relação à quantidade total de amostras no teste.

2.3 RANQUEAMENTO

O processo de ranqueamento de textos é um tópico muito bem desenvolvido já na área de Ciência da Computação, existindo diversos algoritmos para realizar tal tarefa. Focaremos aqui em alguns desses algoritmos como *Term Frequency – Inverse Document Frequency* (TF-IDF) e algumas de suas variações, descrevendo-os numa ordem de evolução.

2.4.3 **TF-IDF**

A TF-IDF, de acordo com Jurafsky e Martin (2023), é o produto da frequência de um termo e da frequência inversa de um documento. A frequência do termo é auto explicável, pois mede a frequência com que um termo aparece, no nosso contexto uma palavra. Agora a frequência de documento é a medida de em quantos

documentos um dado termo/palavra aparece. E a inversa dessa frequência basicamente mede o quão raro uma palavra é, aumentando o valor da medida.

A fórmula é IDF = log(N/df) , onde N é o número total de documentos e df é a frequência de documento. Por fim, o TF-IDF é o produto das duas medidas apresentadas, sendo que as medidas dependem de uma dada palavra em um documento escolhido. Este valor dado uma palavra ou palavras e um documento é utilizado para ranquear documentos de forma probabilística dado uma lista de documentos.

2.3.2 BM-25

Esse algoritmo é uma variação do TF-IDF, adicionando dois parâmetros. O primeiro parâmetro é k de *knob*, mas que funciona como um balanceador entre TF e IDF. Quando k é muito próximo ou igual a 0, o TF é ignorado. Quando o valor de k é muito alto, o parâmetro IDF é ignorado. O segundo parâmetro é b, que controla a importância da normalização do tamanho dos documentos. A implementação utilizada do BM-25 é a implementação Okapi proposta em 1995 no TREC-3 por Robertson *et al.* (1995). Os valores sugeridos para k são uma variação entre [1.2 a 2] e para o parâmetros b é sugerido 0.75 (Manning; Raghavan; Schutze, 2008).

2.3.3 BM-25F

O algoritmo BM25 não considera a diferença entre diferentes partes do texto, por exemplo um título de um corpo ou o corpo de um resumo. Entretanto, um título pode fornecer mais assertividade sobre uma consulta do que uma parte qualquer do texto. Para tentar resolver isso, a variante do BM25-F foi apresentada, onde o valor total de frequência de um termo num documento é balanceada, sendo que se o termo aparecer em determinada parte do texto, terá mais peso do que em outra (Robertson; Zaragoza, 2009).

2.4 PROCESSAMENTO DE LINGUAGEM NATURAL (PLN)

Nesta seção são descritos diversos conceitos do processamento de linguagem natural, muitos deles foram utilizados ao longo do desenvolvimento, outros apenas servem de base e/ou contexto para conceitos mais complexos.

2.4.1 *Corpus*

Um *corpus* é uma coleção de pedaços de linguagem em texto, selecionados de acordo com critérios externos para representar, na medida do possível, uma linguagem ou uma variação de linguagem como fonte de dados para pesquisas linguísticas (Sinclair, 2005). Neste trabalho, utilizamos um *corpus* composto por um conjunto de textos selecionados que servirá como base para a pesquisa em PLN realizada.

2.4.2 *Open Domain Question Answering* (OpenQA)

A tarefa de QA tem como objetivo responder de forma precisa perguntas feitas em linguagem natural por usuários humanos. No entanto, a abordagem convencional do QA é focada em domínios específicos, como medicina ou matemática. No contexto deste trabalho, estamos interessados em abordar a tarefa de OpenQA, que amplia o domínio das perguntas para várias áreas do conhecimento, tornando o *corpus* aberto e diverso. Dessa forma, o sistema de OpenQA não recebe um contexto específico, mas utiliza todo o *corpus* como fonte de entrada (Zhu *et al.*, 2021).

2.4.3 PLN simbólico

A abordagem do PLN simbólico fundamenta-se em técnicas clássicas e baseadas em regras para a análise e processamento da linguagem natural. Nessa abordagem, a linguagem é tratada como um sistema simbólico, onde as palavras e estruturas linguísticas são representadas por símbolos e manipuladas por meio de operações simbólicas.

2.4.3.1 Tokenização

A tokenização é uma etapa fundamental no PLN simbólico, que consiste em dividir um texto em unidades linguísticas menores, chamadas *tokens*. Esses *tokens* podem ser palavras individuais, sentenças ou outras unidades linguísticas pré-definidas. A tokenização é realizada com base em delimitadores, como espaços em branco ou pontuações, que separam os *tokens* no texto.

A escolha adequada dos delimitadores e a definição precisa dos *tokens* são essenciais para o processamento eficiente da linguagem. No entanto, a tokenização no PLN simbólico pode apresentar desafios, especialmente em casos em que há ambiguidade ou unidades linguísticas compostas, como frases ou termos específicos. Nesses casos, técnicas avançadas de tokenização podem ser empregadas, como o uso de expressões regulares para identificar padrões específicos ou a combinação de regras gramaticais para segmentar corretamente o texto (Trim, 2013).

2.4.3.2 Expressões regulares

Expressões regulares são sequências de caracteres que definem um padrão de busca em texto. Elas são utilizadas para realizar operações de correspondência e extração de informações em um texto. As expressões regulares permitem definir regras de busca com base em padrões específicos de caracteres, como letras, dígitos, símbolos especiais, entre outros (Aho; Ullman, 1992).

No contexto deste trabalho, as expressões regulares podem ser úteis para realizar a tokenização de um texto, ou seja, dividir o texto em unidades linguísticas menores, como palavras ou sentenças. Ao definir um padrão de busca correspondente a espaços em branco, pontuações e outros delimitadores, é possível separar o texto em *tokens* de forma eficiente.

Além disso, as expressões regulares podem ser empregadas na busca e recuperação de informações específicas em um conjunto de documentos. Por exemplo, é possível definir um padrão de busca que corresponda a datas, permitindo a extração de informações relacionadas a eventos temporais. Essa funcionalidade é relevante para a compreensão de perguntas e respostas, pois possibilita a identificação de informações relevantes nos documentos.

2.4.3.3 Gramáticas

Gramáticas são formalismos utilizados para descrever a estrutura e a organização de uma linguagem. Elas são compostas por um conjunto de regras que definem as possíveis combinações de símbolos em uma linguagem. As gramáticas podem ser classificadas em diferentes tipos, como gramáticas regulares, gramáticas livres de contexto, gramáticas sensíveis ao contexto e gramáticas irrestritas (Sipser, 1998).

No âmbito do processamento de linguagem natural, as gramáticas livres de contexto são amplamente utilizadas. Elas permitem descrever a estrutura gramatical de uma linguagem natural de forma abstrata, definindo regras para a formação de frases e a combinação de elementos linguísticos, como sujeitos, verbos e objetos.

As gramáticas podem ser aplicadas em tarefas como a análise sintática e a geração de sentenças. Na análise sintática, as regras gramaticais são utilizadas para determinar a estrutura gramatical de uma sentença, identificando os constituintes sintáticos e as relações entre eles. Já na geração de sentenças, as regras gramaticais são empregadas para construir frases válidas de acordo com a gramática especificada.

No contexto deste trabalho, o uso de gramáticas pode ser explorado para auxiliar na compreensão de perguntas e na geração de respostas gramaticalmente corretas. Por meio da definição de regras gramaticais adequadas, é possível garantir que as respostas geradas sigam as estruturas gramaticais da linguagem natural.

2.4.3.4 Ontologia

Ontologia é uma disciplina que estuda a natureza da existência e as categorias fundamentais da realidade. No contexto do processamento de linguagem natural, uma ontologia refere-se a uma representação formal e estruturada do conhecimento em uma determinada área de domínio. Ela define os conceitos, as relações e as propriedades relevantes para esse domínio, permitindo a organização e a categorização do conhecimento de forma semântica (Powers, 1991).

As ontologias são utilizadas para capturar informações e conhecimentos específicos de um domínio, o que facilita a compreensão e o processamento de linguagem natural. Elas fornecem uma estrutura hierárquica para representar os termos e conceitos relevantes, bem como as relações entre eles. Dessa forma, é

possível estabelecer uma base de conhecimento que pode ser consultada para responder perguntas e fornecer informações precisas.

No âmbito deste trabalho, o uso de ontologias pode contribuir para a compreensão de perguntas e respostas, permitindo o acesso a informações mais estruturadas e contextualizadas. Por exemplo, uma ontologia sobre medicina pode incluir conceitos relacionados a doenças, tratamentos, sintomas e medicamentos, o que possibilita uma melhor compreensão de perguntas nesse domínio e uma resposta mais precisa.

Além disso, as ontologias podem ser utilizadas em conjunto com técnicas de recuperação de informações para enriquecer a busca e a recuperação de documentos. Por meio da associação de termos e conceitos presentes nos documentos com os conceitos definidos na ontologia, é possível melhorar a precisão e a relevância dos resultados retornados em uma busca.

2.4.4 PLN neural

A abordagem do PLN neural baseia-se no uso de modelos de redes neurais para o processamento e compreensão da linguagem natural. Essa abordagem tem ganhado destaque nos últimos anos devido aos avanços na área de aprendizagem profunda e ao desenvolvimento de técnicas que permitem capturar informações contextuais e semânticas mais complexas.

Nesta seção são apresentados os conceitos e as técnicas essenciais do PLN neural, que incluem a representação vetorial de palavras, redes neurais, aprendizagem profunda, atenção, autoatenção e transferência de aprendizagem. Esses conceitos são fundamentais para compreender a forma como os modelos de PLN neural abordam a tarefa de compreensão de perguntas e respostas, bem como para o desenvolvimento de sistemas capazes de responder de maneira precisa e eficiente às perguntas feitas em linguagem natural.

2.4.4.1 Representação vetorial de palavras

A representação vetorial de palavras é uma técnica que mapeia palavras em vetores numéricos. Inicialmente, essa técnica foi utilizada para criar representações reduzidas de um texto contínuo. No entanto, à medida que as redes neurais

avançaram, novas abordagens surgiram, permitindo a incorporação de informações semânticas e contextuais nos vetores de palavras.

Dentre as técnicas de representação vetorial de palavras, destaca-se o conceito de "saco de palavras" (*bag-of-words*), onde a ordem das palavras não é levada em consideração, mas sim sua presença e frequência. Além disso, o modelo *Skip-gram* propõe a representação vetorial de palavras com base na previsão das palavras próximas a uma determinada palavra, levando em conta o contexto em que ela aparece.

Outro modelo de representação vetorial de palavras amplamente utilizado é o BERT. O BERT utiliza uma abordagem de aprendizagem prévia para capturar informações mais complexas e contextuais nas representações vetoriais, possibilitando uma melhor compreensão das nuances e ambiguidades presentes na linguagem natural (Devlin *et al.*, 2019).

2.4.4.2 Redes neurais artificiais

As redes neurais artificiais (RNAs) são sistemas computacionais inspirados no funcionamento dos neurônios e do cérebro humano. Elas são compostas por camadas de neurônios interconectados, onde cada neurônio recebe sinais de entrada, realiza operações matemáticas e produz um sinal de saída.

No contexto do PLN neural, as redes neurais desempenham um papel fundamental no processamento e compreensão da linguagem natural. Diferentes tipos de redes neurais são empregados, dependendo da tarefa específica em questão. Entre eles, destacam-se as redes neurais convolucionais (CNNs), que são eficazes na extração de características e padrões em dados sequenciais, e as redes neurais recorrentes (RNNs), que são adequadas para lidar com a natureza sequencial das sentenças (Bet, 2005).

2.4.4.3 Aprendizagem profunda

A aprendizagem profunda, também conhecida como DL, busca representar abstrações complexas, como a linguagem e a visão, por meio de múltiplas camadas de operações não lineares. Essa abordagem permite que os modelos de PLN neural

aprendam representações hierárquicas e de maior nível de abstração dos dados de entrada.

Por meio da aprendizagem profunda, é possível capturar informações contextuais e semânticas mais profundas, melhorando a capacidade de compreensão e geração de respostas precisas. Redes neurais profundas, como os modelos baseados em *Transformers*, têm alcançado resultados impressionantes em diversas tarefas de PLN, devido à sua capacidade de processar informações em paralelo e capturar dependências de longo alcance. Essa abordagem tem sido aplicada com sucesso em várias tarefas de PLN (Bengio, 2009).

2.4.4.4 Atenção e autoatenção

A atenção é um mecanismo fundamental no PLN neural, que permite aos modelos focarem em partes específicas da entrada, destacando informações relevantes e contextos importantes para a tarefa em questão. A atenção é amplamente utilizada para melhorar a representação de contextos mais amplos e para capturar relações complexas entre palavras e sentenças.

Um exemplo de mecanismo de atenção é a autoatenção (*self-attention*), que permite relacionar diferentes posições de uma sequência e criar uma representação contextual para cada elemento da sequência. Por meio da autoatenção, é possível levar em consideração a importância relativa de cada palavra ou elemento, permitindo uma compreensão mais precisa das relações e do contexto global (Vaswani *et al.,* 2017).

2.4.4.5 Transferência de aprendizagem

A transferência de aprendizagem é uma técnica que permite treinar modelos em um conjunto de dados de treinamento com características diferentes do conjunto de dados de teste. Essa abordagem tem sido aplicada em PLN para aproveitar o conhecimento prévio aprendido em tarefas relacionadas, melhorando o desempenho em novas tarefas. Modelos como BERT utilizam a transferência de aprendizagem para pré-treinar as representações de linguagem e adaptá-las a tarefas específicas por meio do ajuste fino (Devlin *et al.*, 2019).

2.4.5 Compreensão de perguntas e respostas

A compreensão de perguntas e respostas é uma área de pesquisa que busca desenvolver sistemas capazes de entender perguntas feitas por humanos e fornecer respostas adequadas e precisas. Essa tarefa envolve técnicas de sumarização, extração de informações e modelos de linguagem pré-treinados. No contexto deste trabalho, a compreensão de perguntas e respostas é aplicada para responder perguntas factuais sobre os assuntos presentes nos documentos do repositório de arquivos fornecidos pelo usuário (Zhu *et al.*, 2021).

2.4.5.1 Análise sintática e semântica

Para compreender perguntas, é essencial analisar a estrutura sintática e a semântica das sentenças. A análise sintática envolve a identificação da estrutura gramatical da pergunta, como o reconhecimento de sujeito, verbo e objetos. Por sua vez, a análise semântica busca atribuir significado às palavras e identificar as relações entre elas. Isso permite determinar a intenção do usuário e extrair informações relevantes para gerar uma resposta apropriada (Zhu *et al.*, 2021).

2.4.5.2 Classificação de perguntas

Um aspecto importante na compreensão de perguntas é a classificação delas em categorias específicas. Isso ajuda a direcionar o processamento adequado para diferentes tipos de perguntas. Por exemplo, as perguntas podem ser classificadas em categorias como perguntas factuais, perguntas opinativas, perguntas de localização, entre outras. A classificação de perguntas pode ser realizada com base em padrões linguísticos, algoritmos de aprendizado de máquina ou combinação de ambos.

2.4.5.3 Extração de informações

A extração de informações é um processo essencial na compreensão de perguntas. Envolve identificar entidades mencionadas na pergunta, como pessoas, lugares ou datas, e recuperar informações relevantes relacionadas a essas entidades. Isso pode ser feito por meio do uso de bases de conhecimento ou recursos externos,

como bancos de dados ou ontologias, para obter informações precisas que possam contribuir para a geração de uma resposta adequada (Zhu *et al.*, 2021).

2.4.5.4 Modelos de linguagem pré-treinados

Os modelos de linguagem pré-treinados, como o BERT, têm desempenhado um papel significativo na compreensão de perguntas e respostas. Esses modelos são treinados em grandes quantidades de texto para capturar relações contextuais e aprendizados linguísticos gerais. Eles podem ser adaptados e ajustados em conjuntos de dados específicos para melhorar seu desempenho em tarefas de compreensão de perguntas e geração de respostas (Devlin *et al.*, 2019).

2.4.5.5 Avaliação da qualidade das respostas

Avaliar a qualidade das respostas geradas é um aspecto crítico da compreensão de perguntas e respostas. Existem várias métricas de avaliação, incluindo precisão, completude, relevância e coerência. Além disso, é importante levar em consideração a adequação e a fluidez da resposta em relação à pergunta feita pelo usuário. A avaliação pode ser realizada manualmente por especialistas ou por meio de técnicas automatizadas, como comparação com respostas de referência ou uso de modelos de avaliação específicos (Rajpurkar *et al.*, 2016).

2.4.5.6 Desafios e avanços

A compreensão de perguntas e respostas apresenta desafios significativos devido à diversidade linguística, ambiguidades, perguntas mal formuladas e necessidade de conhecimento contextual. No entanto, avanços recentes em modelos de linguagem pré-treinados, algoritmos de aprendizado de máquina e técnicas de processamento de linguagem natural têm impulsionado melhorias nessa área. Abordagens baseadas em redes neurais profundas, como modelos de atenção e *transformers*, têm alcançado resultados impressionantes na compreensão de perguntas e respostas.

3 TRABALHOS RELACIONADOS

Neste capítulo são abordadas arquiteturas de sistemas avaliados na tarefa de encontrar respostas para perguntas de domínio aberto (*OpenQA*) que contenham aspectos utilizados neste trabalho. Um dos critérios utilizados é o desempenho nos *datasets*[1] *SQuAD, NaturalQuestions, CuratedTrec e TriviaQA*. Outro critério é o tipo de treinamento utilizado na arquitetura, vários sistemas menores que realizam subtarefas em sequência (*pipeline*) ou ponta a ponta (*end-to-end*). Um terceiro critério importante para este trabalho é se o modelo utiliza transferência de aprendizagem (*transfer learning)*, pois modelos pré-treinamos são menos custosos computacionalmente para implementar. O último critério é se o modelo é reproduzível, se contém documentação ou algum tipo de artefato que permita ser reproduzido.

Primeiro é apresentada a arquitetura *Retrieve and Read* que incorpora a tarefa de compreensão de leitura com a tarefa de recuperar arquivos para que o recuperador seja capaz de entender o contexto da pergunta. Então falamos de BERT e algumas arquiteturas que o utilizam para desempenhar a tarefa de *OpenQA* com um treinamento ponta a ponta. Em sequência, são apresentados outros trabalhos que alcançaram bons resultados, mas que tem uma arquitetura baseada em métodos estáticos e/ou que não compartilham treinamento entre as diferentes etapas. Por fim, uma comparação entre os trabalhos mencionados é apresentada.

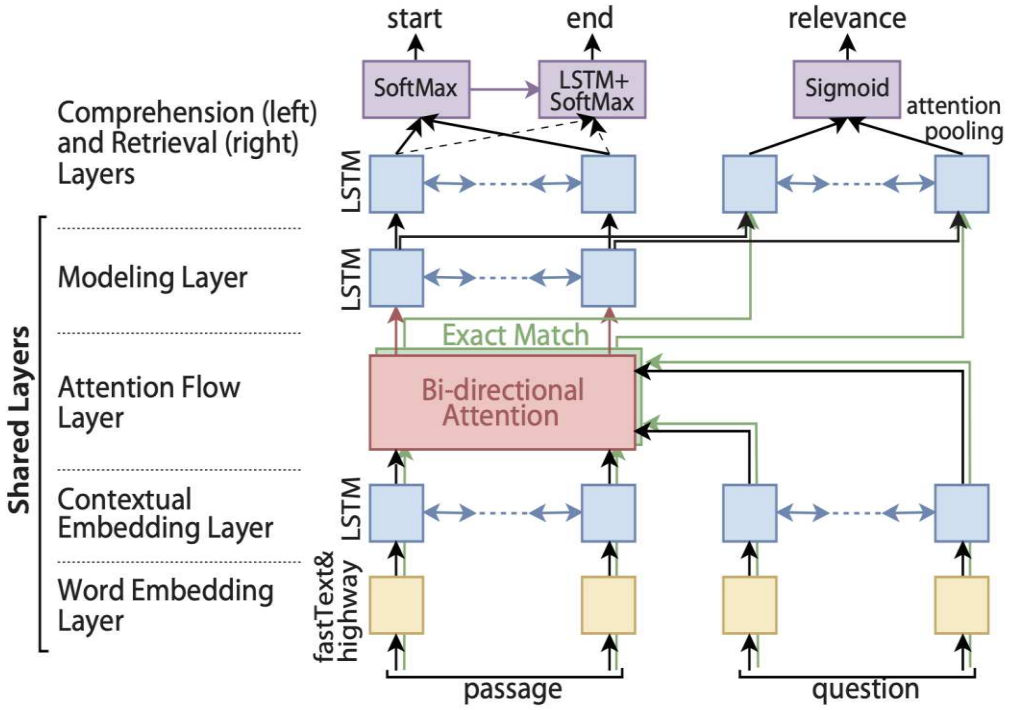
3.1 RETRIEVE AND READ

Uma arquitetura relevante para este trabalho é a proposta por Nishida *et al.* (2018), baseando-se em dois passos: Recuperar e Ler (*Retrieve and Read*). Nishida *et al.* (2018) argumentam que a tarefa de compreensão da leitura (*Reading Comprehension* – RC) melhora o processo de recuperação de informação (*Information Retrieval* – IR), pois o sistema conseguirá distinguir de uma forma mais precisa quais partes do texto são realmente relevantes para a pergunta em questão. A proposta deles consiste em utilizar um modelo supervisionado multitarefas de aprendizagem de máquina.

[1] *Datasets* são um conjunto de dados estruturados.

Como demonstrado na Figura 1 a seguir, a arquitetura de Nishida *et al.* (2019) compartilha entre as duas tarefas, RC e IR, as camadas que executam as mesmas subtarefas. Para construir a saída, cada tarefa tem uma camada específica que utiliza de entrada a saída das camadas compartilhadas, transferindo o treinamento de uma tarefa para outra e realizando apenas o ajuste fino para a tarefa em questão, IR ou RC. A camada de RC deriva a frase de resposta para uma pergunta, prevendo o índice do começo e do fim em uma frase de algum corpo textual utilizando o conceito de *Pointer Networks*[2]. Enquanto a camada de IR utiliza um comparador binário (*binary exact-matching*) e um mecanismo de autoatenção para calcular a relevância de uma dada passagem para uma pergunta. Os testes executados demonstram que o sistema de IR conseguiu alcançar o estado da arte em *benchmarks* como SQuAD v1.1. A principal diferença dessa arquitetura para a que está sendo proposta neste trabalho é que não é baseada em aprendizagem de máquina supervisionado, mas sim em *deep learning*, entretanto essa visão de *retrieve and read* ainda é utilizada para este trabalho.

Figura 1 - Arquitetura proposta.



[2] Uma forma de lidar com o problema de representar dicionários de tamanho variáveis utilizando uma distribuição probabilística *softmax* como "ponteiro" para um dado específico dos dicionários.

Fonte: Nishida *et al.* (2019, p. 2).

3.2 BERT

O BERT é um método de pré-treinamento das representações de linguagem. O pré-treinamento significa que primeiro o BERT é treinado em um grande corpo textual e então os parâmetros do modelo recebem um ajuste fino (*fine-tuning*) para uma tarefa específica de NPL, como por exemplo QA. Outros modelos utilizam uma arquitetura unidirecional, na qual o *token* atual conhece apenas os *tokens* anteriores.

Dessa forma, modelos como *OpenAI GPT* só conhecem o contexto dos *tokens* anteriores e, segundo Devlin *et al.* (2019), isso acaba comprometendo a performance de modelos pré-treinados com ajuste fino. Para resolver tal problema, BERT utiliza um modelo de linguagem mascarado (MLM) para prever o identificador no vocabulário da palavra mascarada apenas pelo contexto. Com isso, o BERT é capaz de ser pré-treinado de forma bidirecional (Devlin *et al.*, 2019).

A arquitetura do BERT baseia-se na arquitetura de Transformadores (*Transformers*) proposta por Vaswani *et al.* (2017), que elimina a dependência de redes neurais recorrentes e utiliza um mecanismo de atenção para criar as relações entre entrada e saída. A motivação da arquitetura *Transformers* é permitir uma maior paralelização do treinamento removendo computações sequências exigidas pelas redes neurais recorrentes e suas camadas ocultas (*hidden layers)* (Vaswani *et al.*, 2017).

O mecanismo de atenção pode ser descrito como o mapeamento entre uma consulta e um conjunto de chave/valor com uma saída, onde todas as informações são vetores. No trabalho de Vaswani *et al.* (2017), o mecanismo de atenção utilizado é chamado de autoatenção, caracterizado por levar em consideração diferentes posições de uma entrada para criar a representação de tal entrada. A saída é computada pela soma ponderada dos valores baseada em pesos e os pesos por sua vez são computados pela função de compatibilidade da consulta com a chave correspondente.

A representação dos dados de entrada utilizado no BERT permite representar de forma desambigua uma única sentença ou um par de sentenças (como é o caso deste trabalho de pergunta/resposta). O vocabulário de *tokens* utilizado é o *WordPiece* (Wu *et al.*, 2016). O funcionamento do BERT envolve dois passos:

- Pré-treinamento, que pode ser dividido em duas subtarefas:

    - Criar um modelo de linguagem mascarado, também conhecido como tarefa Cloze (Taylor, 1953);
    - Predição da Próxima Sentença (*Next Sentence Prediction - NSP*), que utiliza um par de sentenças A e B. A tarefa consiste em classificar B como próxima sentença de A ou não. Para construir os exemplos de treinamento, 50% das vezes B é a sentença esperada e 50% das vezes B é uma sentença aleatória do corpo textual. Para o pré-treinamento, os corpos textuais usados são *BookCorpus* (Devlin, 2018), com 800 milhões de palavras, e a Wikipedia em inglês, com 2,5 bilhões de palavras.
- Ajuste fino: basta alterar a ordem das entradas e saídas do modelo BERT. Para a tarefa deste trabalho que é *Question Answering* pode-se entender o par de sentença A e B do passo de pré-treinamento como um par de perguntas/respostas dos conjuntos de dados mais comuns como o *SquaD* ou *SimpleQuestions*.

Os experimentos na tarefa de *Question Answering* mostram que o BERT superou outros trabalhos publicados na época. Os experimentos foram realizados sobre o *dataset SQuAD v1.1* e o *SQuAD v2.0*. Para *SQuaD v1.1*, primeiro realizaram o ajuste fino utilizando o *dataset TriviaQA* e então reaplicaram o ajuste fino com o *SQuAD v1.1*, desta forma alcançando uma pontuação no F1 de 93.2 (Devlin *et al.*, 2019), sendo F1 a média harmônica entre precisão (*precision*) e *recall*, também conhecido como sensibilidade. A única diferença para o *SQuAD v2.0* é que o ajuste fino com o *TriviaQA* não é realizado, apenas com o próprio *SQuAD v2.0*, alcançando 83.1 na medida F1. Portanto, supera os artigos previamente publicados que tinham o melhor desempenho (Devlin *et al.*, 2019).

3.2.1 ORQA

Outros trabalhos se utilizam da arquitetura proposta por BERT para desempenhar a tarefa de *Question Answering* (QA). Um exemplo é o trabalho de Lee *et al.* (2019), que constrói um sistema ponta a ponta para QA unificando as tarefas de recuperação e leitura (*retrieve and read*). Esse artigo traz uma proposta parecida com a desenvolvida neste trabalho, pois utiliza o BERT para executar a tarefa de QA de ponta a ponta, excluindo sistemas intermediários de recuperação de arquivos. Lee *et*

al. (2019) define um novo conceito de sistema de recuperação aberta para respostas de perguntas, *Open-Retrieval Question Answering* (ORQA).

A principal diferença com sistemas de domínio aberto para respostas de pergunta é que o corpo textual de busca não é primeiro reduzido por um recuperador de arquivos, ao invés disso o corpo textual é tratado como uma variável latente para o modelo de ponta a ponta. Lee *et al.* (2019) argumentam que o aprendizado de ponta a ponta para essa tarefa só é possível graças a uma tarefa inversa de Cloze, a *Inverse Cloaz Task* (ICT).

A tarefa ICT consegue ensinar o modelo reconhecer contexto escolhendo sentenças do corpo textual de treino como uma pseudopergunta e seu contexto como uma pseudoevidência, então o modelo recebe várias evidências embaralhadas com a pseudoevidência para discernir qual a sentença correta para a pseudopergunta de entrada. Após pré-treinar o modelo com ICT, é realizado um ajuste fino com os *datasets* escolhidos, pois a ICT já permite que o modelo tenha um desempenho não trivial em uma situação de *zero-shot*.

Os *datasets* utilizados foram: *Natural Questions* (33.3)*, WebQuestions* (36.4)*, CuratedTrec* (30.1)*, TriviaQA* (45.0) e *SQuAD* (20.2). O modelo utilizado para comparação foi o BM25 e a diferença nos 3 primeiros *datasets* varia de 6 a 19 em favor do modelo ORQA. Nos dois últimos, *TriviaQA* e *SQuAD*, o modelo ORQA acabou perdendo por uma margem de 2 a 13. Lee *et al.* (2019) argumentam que o desempenho abaixo do rival é causado pelo enviesamento dos *datasets* SQuAD e TriviaQA na sua natureza em relação a tarefa de IR porque os escritores das perguntas já sabiam as respostas e sabiam que estavam presentes no corpo textual.

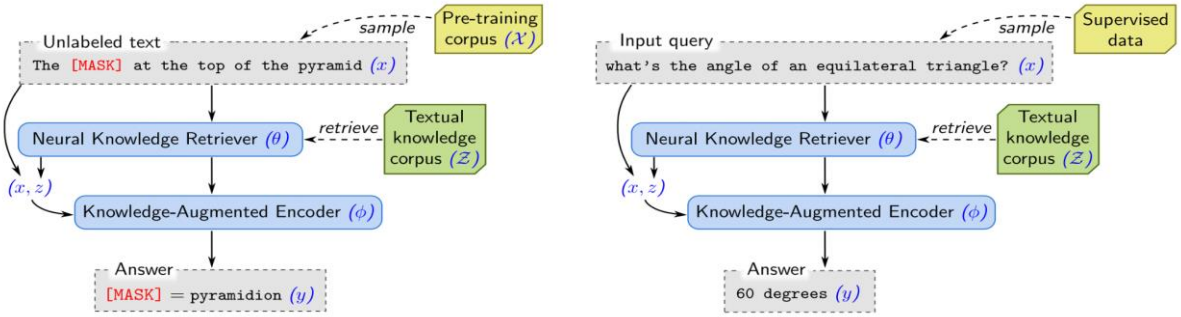
Além disso, é ressaltado que o BM25 tem uma capacidade muito maior de representar os documentos e é especializado na tarefa de IR. Entretanto, no presente trabalho o objetivo é justamente lidar com cenários mais reais, onde o usuário do sistema não sabe exatamente se a resposta está presente nos documentos, se assemelhando mais ao objetivo do trabalho de Lee *et al.* (2019).

3.3 REALM

Outra arquitetura relevante que utiliza BERT como base é o *Retrieval-Augmented Language Model Pre-Training* (REALM). A relevância dessa arquitetura vem da capacidade de recuperar conhecimento de uma forma latente. Os *datasets*

escolhidos para avaliar REALM foram *NaturalQuestion-Open*, *WebQuestions* e *CuratedTrec.* O modelo REALM contém duas peças, o recuperador de conhecimento neural (*Neural Knowledge Retriever* – NKR) e o codificador de conhecimento (*Knowledge-Augmented Encoder* – KAE). O NKR é responsável por modelar a distribuição p(z|x), onde x é a entrada (uma frase com algum *token* mascarado) e z são documentos relevantes para x. KAE é responsável por modelar a distribuição p(y,z|x), onde y é um possível *token* faltante em x como demonstrado na Figura 2.

Figura 2 - Arquitetura proposta por Guu et al. (2020).



Fonte: Guu *et al.* (2020).

Notas: Do lado esquerdo: fase de pré-treino sem supervisão, onde recuperador de conhecimento e o codificador de conhecimento são conjuntamente treinados na tarefa de modelagem de linguagem mascarada. Do lado direito: ajuste fino supervisionado na tarefa desejada, nesse caso QA.

Os resultados de REALM superaram o estado da arte da época por uma margem relevante. Os resultados foram obtidos submetendo REALM aos *datasets* já mencionados, o critério de escolha foi datastes que os escritores das perguntas não sabiam das respostas, pois segundo Guu *et al.* (2020) é mais próximo de um cenário real onde se busca uma informação.

REALM comparou-se com os seguintes modelos: BERT-Baseline, T5 (base, large e 11b), DrQA, HardEM, *GraphRetriever, PathRetriever* e ORQA. T5-11b, com o melhor desempenho desta lista de modelos, pontuou: 34.5 no *dataset NaturalQuestions*, enquanto REALM 40.4; e 37.4 no *dataset WebQuestions*, enquanto REALM 40.7. No terceiro *dataset* o melhor competidor é ORQA com 30.1, enquanto REALM alcançou 46.8. Dessa forma, REALM superou nesses 3 *datasets* os melhores competidores da época, assim se tornando relevante para este trabalho.

3.4 OUTROS TRABALHOS

Nesta seção são apresentados modelos parcialmente/marginalmente relacionados com este trabalho, por isso não são descritos com muitos detalhes. No entanto, são trabalhos utilizados como comparativo dos trabalhos mencionados previamente ou que obtiveram bons resultados. Primeiro é apresentado DrQA, que utiliza uma arquitetura mista de métodos estáticos e de rede neural. Então é apresentado RankQA que reintroduz o conceito de rerranqueamento de respostas ao status quo de QA, que é a pipeline de duas etapas (recuperar arquivos relevantes, selecionar respostas). Por fim, é apresentado o trabalho de Clark e Gardner (2018), pelo seu desempenho e por ser reproduzível.

Uma arquitetura importante de apresentar neste trabalho é DrQA (Chen *et al.*, 2017), pois outros trabalhos mencionam e comparam os seus próprios resultados com o de DrQA. DrQA é composto de: um recuperador de documentos que usa *hashing* de duas palavras e TF-IDF para dada uma pergunta recuperar um conjunto de documentos relevantes; um leitor de documentos que usa uma rede neural recorrente de multicamadas treinada para detectar a resposta dada uma pergunta e os documentos selecionados pelo recuperador de documentos.

A arquitetura foi avaliada de três formas, primeiro apenas o recuperador de documentos, então o leitor de documentos e por fim os dois atuando em conjunto para a tarefa de QA. No recuperador de documentos, o desempenho foi comparado com o motor de busca *Wiki Search*, superando o competidor de 2 a 15 pontos, dependendo do conjunto de dados escolhido (*SQuAD, CuratedTREC, WebQuestions, WikiMovies*).

O leitor de documentos foi avaliado apenas com o conjunto de dados *SQuAD*, ficando atrás apenas de R-net (competidores: BiDAF, R-net, *Multi-Perspective Matching, Dynamic Coattention Networks*) por 1.3 na medida EM e 1.7 na medida F1. Por fim, a arquitetura como um todo foi avaliada na tarefa de QA usando três configurações diferentes.

Entretanto, as três configurações tiveram desempenho inferior ao competidor selecionado, YodaQA, por uma média de 12.5 pontos, a medida utilizada é a acurácia do primeiro documento recuperado em uma comparação exata com o esperado. Uma possível melhoria apontada por Chen *et al.* (2017) é treinar o modelo de ponta a ponta na tarefa de QA, ao invés de usar os componentes treinados independentemente.

A arquitetura RankQA proposta por Kratzwald, Eigenmann e Feuerriegel (2019) tem como diferencial um terceiro passo em contrapartida do *status quo* de QA, em que se utilizam dois passos: recuperar arquivos relevantes e encontrar respostas nos arquivos relevantes. Essa terceira etapa do modelo é realizar um novo ranqueamento das respostas gerando novas características para cada resposta e utilizando essas novas características para ranquear as respostas. Apesar de ser um método simples e leve, os resultados nos *datasets SQuADOpen, CuratedTrec, WebQuestions e WikiMovies* foram superiores ao BERT com ajuste fino para QA. O treinamento do modelo foi feito utilizando os subconjuntos de treino de cada *dataset*.

Outra arquitetura relevante devido ao seu desempenho e por ser reproduzível é a proposta por Clark e Gardner (2017). O seu modelo é complexo e baseado em *pipeline*, utilizando uma combinação de heurística baseada em TF-IDF, redes neurais com autoatenção e rede neural recorrente, totalizando 12 componentes. Esse modelo foi avaliado e treinado em dois *datasets* distintos, *TriviaQA e SQuAD*. Os resultados no conjunto de dados *TriviaQA* foram superiores por 15 pontos ou mais às arquiteturas escolhidas como referência, *Mnemonic Reader* (Hu *et al.*, 2017) e *Reading Twice for NLU* (Weissenborn *et al.*, 2017a). Os resultados no conjunto de SQuAD entretanto colocaram o trabalho de Clark e Gardner (2017) em oitavo lugar quando publicado.

A seguir apresentaremos uma comparação entre os trabalhos apresentados acima, de tal forma a elucidar critérios relevantes para o presente trabalho.

O primeiro critério é se o modelo desempenha a tarefa de perguntas e respostas de domínio aberto, pois é um dos objetivos deste trabalho.

O segundo critério é como que o modelo foi treinado, de forma serializada dividindo a tarefa de OpenQA em diferentes subtarefas.

O terceiro critério é se utiliza transfer learning, pois isso facilita o treinamento se utilizarmos modelos pré-treinados.

O quarto critério é a utilização de autoatenção, pois permite criarmos uma representação considerando várias partes da sentença.

O quinto critério é se o modelo é reproduzível, ou seja, se possui código disponibilizado ou documentação explicando como reproduzir.

Por fim, as métricas e os *datasets* são considerados, pois isso nos permite uma comparação mais fácil entre os modelos e o nosso.

Tabela 1. Trabalhos Relacionados

Arquitetura	***OpenQA***	***Pipeline/ end-to-end***	***Transfer Learning***	**Autoatenção**	**Reproduzível**	**Métricas utilizadas**
Retrieve and Read	Sim	Pipeline	Não	Sim	Não	EM e F1 no *dataset* SQuAD.
BERT	Sim	End-to-end	Sim	Sim	Sim	EM e F1 nos *datasets* GLUE, SQuAD v1.1 e SQuAD v2.0.
ORQA	Sim	End-to-end	Sim	Sim	Sim	EM nos *datasets Natural Questions*, WebQuestions, CuratedTrec, Squad e TriviaQA.
REALM	Sim	End-to-end	Sim	Sim	Sim	EM nos *datasets* NaturalQuestions, WebQuestions e CuratedTrec.
DrQA	Sim	Pipeline	Não	Não	Sim	EM nos *datasets* SQuAD, CuratedTrec, WebQuestions e WikiMovies.
RankQA	Sim	Pipeline	Sim	Sim	Sim	EM nos *datasets* SQuAD, WikiMovies, CuratedTrec e WebQuestions
Reading Comprehension (Clark C., Gardner M., 2017)	Sim	Pipeline	Não	Sim	Sim	EM e F1 nos *datasets* SQuAD e TriviaQA.

Fonte: Elaborada pelo autor (2023).

Notas: Demonstra que os trabalhos selecionados contém pelo menos um *dataset* e uma métrica em comum com os outros trabalhos. Algo relevante é que quase todos os trabalhos são de fato reproduzíveis, permitindo uma comparação mais controlada.

Tendo a Tabela 1 e os resumos acima como base, algumas partes de arquitetura e/ou ideias apresentadas são reutilizadas e/ou reaproveitadas. O primeiro critério para reutilizar algo é ser replicável, outro critério é o tipo de treinamento (*end-to-end* ou *pipeline*) e por fim o último critério é se utiliza transfer learning pois poderíamos utilizar modelos pré-treinados, assim não sendo penalizado pelo custo computacional para treinar um modelo de NPL.

O primeiro modelo a ser utilizado é o BERT, como vimos existem várias formas de utilizar o BERT para realizar a tarefa de QA, podendo ser *end-to-end* ou *pipeline*. Além disso, ele é facilmente reproduzível e utiliza transfer learning, desta forma podemos utilizar um modelo pré treinado. Utilizamos BERT de forma pura, construindo um modelo end-to-end nos aproveitando da capacidade de utilizar MLM, modelo de linguagem mascarado.

O segundo conceito que utilizaremos é o apresentado na arquitetura ORQA, onde o recuperador de arquivos e o leitor de arquivos são treinados em conjunto, para que o recuperador de arquivos seja capaz de reconhecer contextos. ORQA é reproduzível, contendo uma página no *github* e documentação. ORQA baseia-se em BERT, consequentemente utiliza *transfer learning*.

Por fim, utilizaremos o conceito de conhecimento como uma variável latente utilizado no REALM. Desta forma, poderemos entender melhor de onde vem a inferência do modelo. Além disso, ganhamos a vantagem de o modelo ser menos custo para treinar, pois não precisamos crescer o modelo conforme a quantidade de arquivos cresce, já que o conhecimento é uma variável latente e não embutida nos parâmetros do modelo. Essa arquitetura é reproduzível, utiliza *transfer learning* e é *end-to-end*, dessa forma se encaixa nos critérios.

4 DESENVOLVIMENTO

Neste capítulo é apresentado o desenvolvimento realizado neste trabalho. O capítulo está dividido em modelos preliminares que foram desenvolvidos e o modelo final. A cada iteração sobre os modelos preliminares, mais complexidade é introduzida ao modelo, bem como pontos fracos são abordados. O objetivo é alcançar um desempenho considerável em relação ao dos trabalhos relacionados nos mesmos *datasets* mencionados, além de permitir um *zero shot* com uma confiança de pelo menos 0.7 nos arquivos enviados pelo usuário.

Cada seção deste capítulo é uma iteração do modelo, sendo que cada iteração é composta por algumas subseções que se repetirão entre cada modelo. Essas subseções descreverão: 1. Parâmetros usados para configurar o modelo e modificações se houver no modelo; 2. Preparação dos *datasets*; 3. Processo de ajuste fino; 4. Avaliação e resultados nos *datasets* de entrada; 5. Avaliação nos conjuntos de arquivos para *zero shot*; 6. Considerações e melhorias para a próxima iteração.

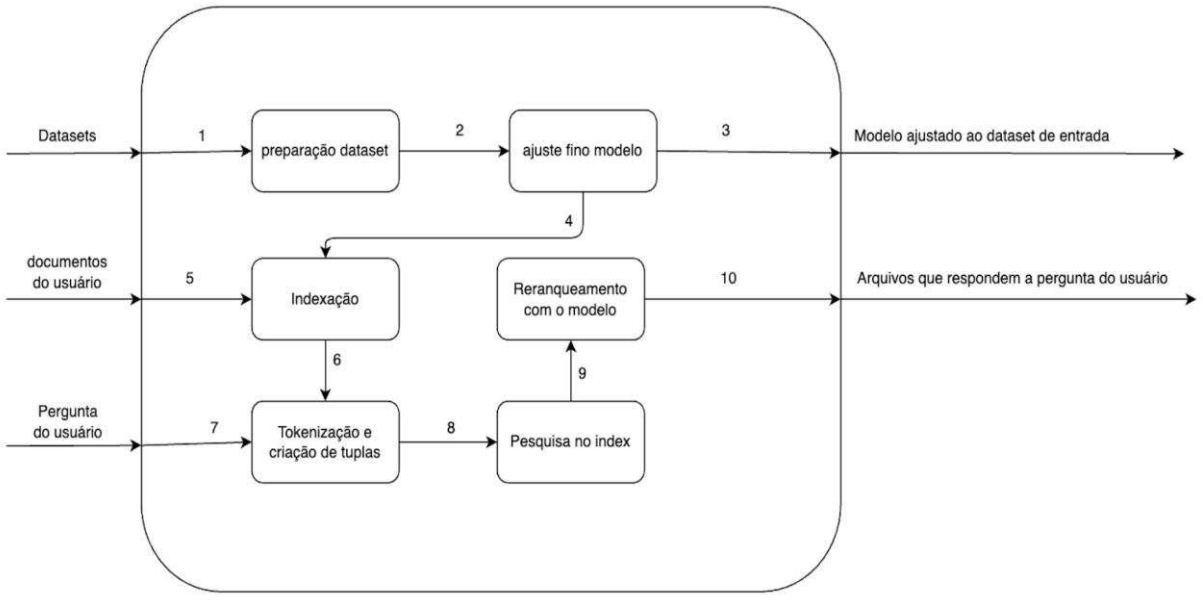
A arquitetura de forma geral é apresentada na Figura 3, onde podemos ver que temos 3 entradas, 10 passos e 6 principais componentes. A entrada *dataset* se refere aos conjuntos de dados utilizados para o treinamento, na nossa arquitetura isso pode ser uma lista de *datasets*, mas por simplicidade utilizamos apenas dois, *NaturalQuestions* e *Squad v1.1*. A entrada de documentos do usuário são conjuntos de arquivos do *dataset* 20newsGroup. Para a pergunta do usuário, diversas perguntas factuais são utilizadas para testar a capacidade de *zero shot* do modelo.

Para cada *dataset* de entrada repetimos um processo como descrito no caso de uso Treinamento da Figura 4. Para isso precisamos de uma implementação que transforma um *dataset* específico para *features* e exemplos, como explicado em mais detalhes na subseção Preparação dos *datasets* na seção Modelo preliminar. Após, enviamos o *dataset* convertido para o processo de ajuste fino (passo 2 da Figura 3) que por sua vez retorna um modelo, esse modelo é salvo (passo 3 da Figura 3) para que futuramente possamos ou ajustá-lo a outro *dataset* ou submetê-lo ao passo 4 da arquitetura.

Seguindo a Figura 3, temos o passo 5, que é receber um conjunto de arquivos do usuário para serem indexados e então recuperados dado uma entrada de busca, passo 6 e 7. Então realizamos um processo de conversão da pergunta e pré-busca

com o algoritmo de indexação, passo 8 e 9. Por fim, criamos tuplas com esses arquivos e a pergunta original e usamos isso de entrada para um novo ranqueamento que é realizado pelo modelo ajustado. Vale ressaltar que o que melhoramos de uma iteração para outra são justamente os componentes responsáveis pelos passos descritos acima. A arquitetura como um todo permanece a mesma de iteração para iteração.

Figura 3 - Arquitetura geral do sistema proposto.

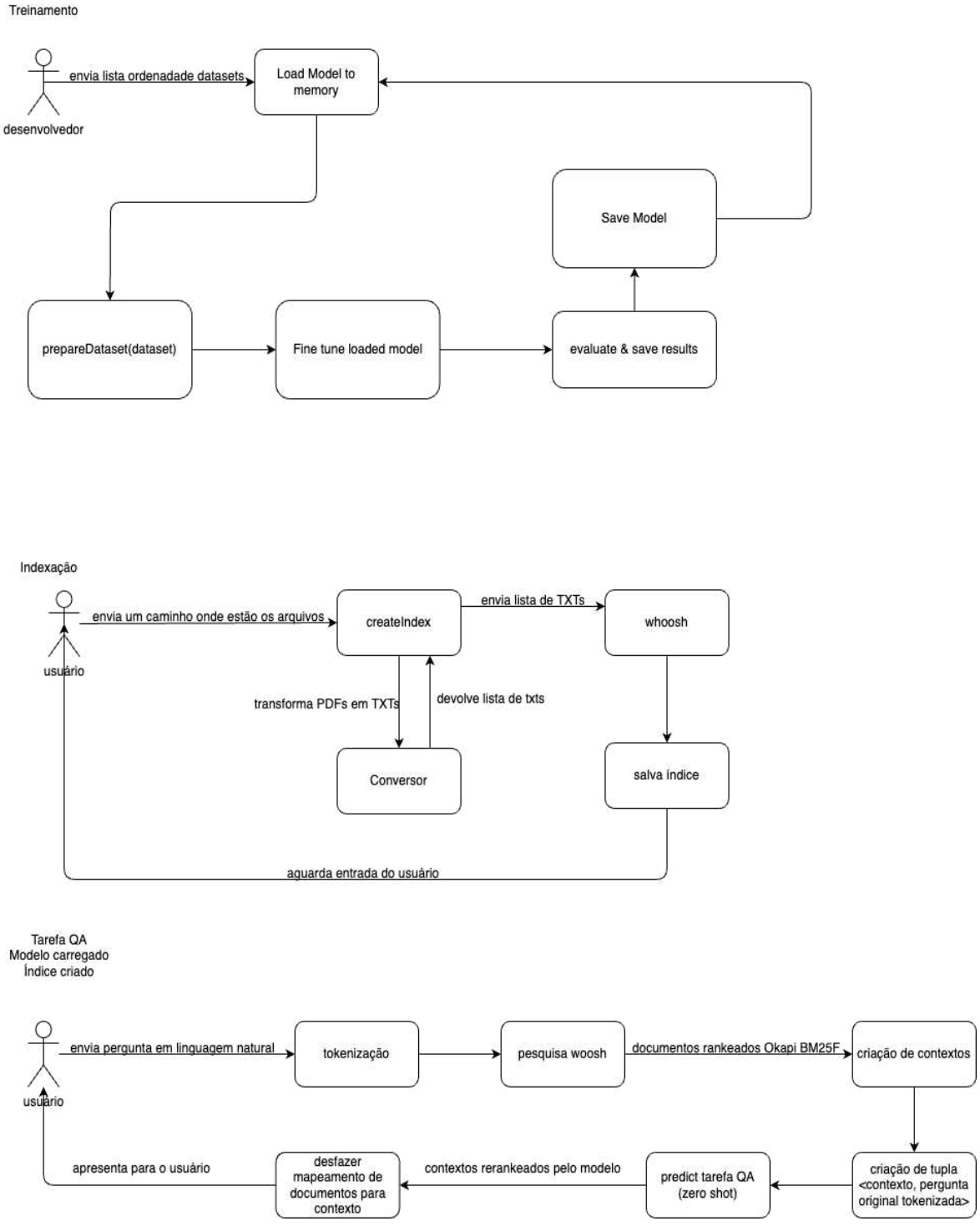


Fonte: Elaborada pelo autor (2023).

Na sequência, vem a fase de avaliação dos resultados, onde ideias e técnicas apresentadas por outros autores são combinadas. Por fim, esse processo é repetido incrementando o modelo até que sejamos capazes de não só responder perguntas nos *datasets* escolhidos, mas em conjunto de arquivos que não foram inicialmente feitos para OpenQA. Desta forma, atendendo o objetivo de servir como um repositório inteligente de arquivos para o usuário final.

A Figura 4 foi feita com o intuito de dividir a arquitetura da Figura 3 em casos de uso. O primeiro caso de uso é chamado de treinamento, mas na verdade esse caso de uso faz o ajuste fino dos *datasets* escolhidos. O segundo caso de uso é o processo de indexação dos arquivos enviados pelo usuário. O terceiro caso de uso demonstra o processo desde que o usuário submete a pergunta ao sistema até receber uma resposta, mostrando quais componentes são chamados.

Figura 4 - Arquitetura geral do sistema proposto dividido em 3 casos de uso: treinamento, carregamento de arquivos e busca de arquivos.



Fonte: Elaborada pelo autor (2023).

4.1 MODELO PRELIMINAR

Partimos de um modelo básico utilizando BERT puro, sem modificação, com indexação simples e pré treinado para a tarefa de OpenQA. Então avaliamos o modelo, submetendo-o aos *datasets SQuAD* e *NaturalQuestions* e comparando com os trabalhos relacionados. O *dataset NaturalQuestions* foi especificamente selecionado pelos trabalhos relacionados e pelo presente trabalho, pois suas perguntas foram escritas por seres humanos e sem o viés de saber se a resposta está presente nos documentos ou não.

A entrada do modelo no ajuste fino é um índice, utilizando o algoritmo Okapi BM-25F para ranqueamento como apresentado na seção de Fundamentos Teóricos, e a entrada do modelo após ajuste fino é apenas uma sentença. A saída do modelo é configurável, podendo retornar uma quantidade variável das respostas com maior confiança.

Nessa iteração utilizamos Python e a biblioteca *Datasets*[3] para conseguir trabalhar com os dados, prepará-los e extrair *features* dos mesmos. Então alimentamos essas *features* ao BERT puro, pré-treinado para a tarefa de openQA com o *dataset* SQuAD para realizar o ajuste fino. A entrada para esse modelo depois do ajuste fino é uma sentença. Na próxima etapa declaramos uma aplicação Python simples, que recebe uma pergunta e repassa ao modelo. A partir daí, o modelo retorna para a aplicação uma lista de 5 respostas e os documentos referentes às respostas. Por fim, apresentamos essa lista ao usuário.

4.1.1 Configuração

Cada versão de tamanho do BERT vem com dois arquivos: um arquivo representando a meta configuração do BERT (bert_config.json) e outro representando a configuração dos milhões de parâmetros que o BERT tem (bert_ckpt). O processo de ajuste fino justamente altera os parâmetros que vem do bert_ckpt, enquanto o arquivo bert_config contém configurações como quantidade de camadas, o tamanho de cada camada, tamanho do vocabulário, quantidade de ponteiros de atenção. Os diversos modelos de tamanho do BERT referem-se aos dois atributos: quantidade de camadas (*hidden layers* - L) e tamanho do vetor de representação das palavras (*token*) nas camadas internas (*hidden layers*).

---

[3] *Datasets* é uma biblioteca do *framework HuggingFace*, disponível em: https://github.com/huggingface/datasets. Acesso em: 18 ago. 2023.

Para validar as propostas de modelo aqui está sendo utilizado o BERT_base que tem 12L e 768H, o que gera 110.1 milhões de parâmetros. Entretanto o modelo BERT_large, que é o original, tem uma configuração de 24L 1024H, gerando um total de 340 milhões de parâmetros. Neste trabalho é utilizado o BERT_base pois é extremamente custoso ajustar o BERT_large mesmo que já tenha sido pré-treinado, o BERT_base não é possível ajustar num computador normal pois o tamanho mínimo de cada conjunto de dados enviados para treinamento (batch_size) não cabe numa memória RAM de 16GB. No entanto, como é apresentado conseguimos utilizar a nuvem da própria Google para execução que também tem seus limites, principalmente um limite de tempo de execução.

Figura 5 - Classificação dos subtipos do modelo BERT.

	H=128	H=256	H=512	H=768
L=2	2/128 (BERT-Tiny)	2/256	2/512	2/768
L=4	4/128	4/256 (BERT-Mini)	4/512 (BERT-Small)	4/768
L=6	6/128	6/256	6/512	6/768
L=8	8/128	8/256	8/512 (BERT-Medium)	8/768
L=10	10/128	10/256	10/512	10/768
L=12	12/128	12/256	12/512	12/768 (BERT-Base)

Fonte: Devlin et. al. (2019).

Figura 6 - Quantidade de parâmetros em milhões.

	H=128	H=256	H=512	H=768
L=2	4.4	9.7	22.8	39.2
L=4	4.8	11.3	29.1	53.4
L=6	5.2	12.8	35.4	67.5
L=8	5.6	14.4	41.7	81.7
L=10	6.0	16.0	48.0	95.9
L=12	6.4	17.6	54.3	110.1

Fonte: Turc *et al.* (2020).

4.1.2 Preparação dos *datasets*

Nesta seção é descrito o processo de preparação e ajuste dos *datasets* para que possamos treinar o modelo nos mesmos. Neste momento, ainda não vamos treinar o mesmo modelo em diferentes *datasets*, entretanto já estamos concebendo a preparação de tal forma que facilite para a próxima iteração do modelo.

4.1.2.1 Squad v1.1

Este *dataset* criado e mantido pela Universidade de Stanford é focado na compreensão do texto contido nele, sendo composto por questões criadas sobre artigos da Wikipédia. O *dataset* em si é um arquivo no formato .json com várias entradas. Cada entrada no conjunto de dados tem a seguinte estrutura:

```
{
  "data":[
    {
      "title":"um título do parágrafo",
      "paragraphs":[
        {
          "context":"texto original",
          "qas":[
            {
              "answers":{
                "answer_start":1,
                "text":"This is a test text"
              },
              "id":"1",
              "question":"Is this a test?"
            }
          ]
        }
      ]
    }
  ]
}
```

Answers é um dicionário onde cada entrada é uma resposta possível para uma dada pergunta *(question)*. Cada resposta contém um índice de onde ela começa no texto principal e o texto em si da resposta. Além disso, cada entrada do conjunto de dados contém um identificador(id) da pergunta.

Para realizar o ajuste fino de um dado modelo BERT (já ajustado previamente em outro *dataset* ou não), primeiro convertemos o *dataset* do formato json para instâncias de uma classe *SquadExample* que basicamente tem as propriedades já descritas anteriormente. O segundo passo é realizar o processo de tokenização dos dados, então é realizado a extração de features dos exemplos de treino, por fim é realizado o processo de predição baseado nas features extraídas anteriormente.

O primeiro processo foi converter o json no formato esperado pelo tokenizador oferecido pela própria google para trabalhar com o BERT. Para isso foi criado o

primeiro passo de conversão do *dataset* em json para objetos do tipo *SquadExample*. Esta classe consiste apenas de propriedades iguais do *dataset*, contendo -1 nos índices de início e fim da resposta para perguntas impossíveis. Ela contém apenas uma função que converte o exemplo de treinamento para uma *string* legível ao usuário, para fins de depuração.

Então o processo de tokenização e extração de *features* de cada exemplo é feito em um único laço de repetição, iterando cada objeto exemplo gerado na primeira etapa. A primeira parte tokenizada do exemplo é a pergunta em si. O tokenizador do BERT primeiro executa uma tokenização chamada de básica, onde é feito(a): conversão para *unicode*; separação por espaço em brancos; conversão para caracteres em caixa baixa; remoção de acentos; divisão de *tokens* por pontuação. A partir daí é executada uma tokenização chamada de WordPiece, desenvolvida pela Google para treinamento do BERT. Essa tokenização utiliza um algoritmo "guloso" de *longest-match-first*, que quer dizer que a palavra de entrada será quebrada encontrando o maior *token* possível presente no vocabulário dado. Por exemplo, se o vocabulário contiver o *token* "un" e a palavra de entrada for "unaffable", o primeiro maior *token* será "un", então a palavra será quebrada nos seguintes *tokens*: ["un", "##affable"] (Devlin *et. al*, 2019).

Após a tokenização completa da pergunta, é criada uma instância de uma classe chamada *InputFeatures*, concentrando várias informações como os *tokens* da pergunta, onde que está a resposta no documento original, a máscara de entrada, o mapeamento para desfazer os *tokens* em texto, o índice do exemplo, se tal exemplo é impossível (não tem resposta). Por fim, o preditor do tensorFlow é invocado tendo como entrada um modelo e uma função mapeadora que converte uma InputFeature para um dicionário esperado pelo TensorFlow. Como resultado, temos um modelo ajustado para os InputFeatures e um arquivo de predição utilizando o ID de cada *feature*.

Para avaliar o resultado com o *script* fornecido pelo próprio *dataset*, precisamos desfazer nossa conversão e escrever um arquivo json contendo o id da pergunta e a resposta em texto que encontramos. Para isso foi utilizado o processo de destokenização fornecido pelo tokenizador da Google e depois com um laço de repetição simples escrevo um json com tuplas de id da pergunta e o texto da resposta predito pelo modelo para tal pergunta. Com esse json em mãos, o script foi executado

e duas medidas são obtidas, F1 e *Exact Match*. Para o desenvolvimento e validação de código, os parâmetros do BERT-BASE foram utilizados.

4.1.2.2 Natural Questions

O *dataset NaturalQuestions* (NQ) foi criado e é mantido pela Google com o objetivo de representar melhor possíveis perguntas de seres humanos sobre dados que nem sempre terão a resposta. O *dataset* contém 307373 exemplos de treino, 7830 de desenvolvimento e 7842 de teste. Um exemplo é composto de uma pergunta que foi submetida ao buscador do google.com, uma página do Wikipedia anotada com a resposta e uma pequena passagem com a resposta derivada da página original. Entretanto a anotação da página que demarca a resposta e a pequena passagem que representa a resposta podem estar vazias.

Assim como o Squad, o formato dos dados do NQ também é em JSON. O formato original dos dados NQ é complexo pois utiliza a página do Wikipedia em HTML, ou seja, o texto útil ainda não foi extraído. Segue exemplo de uma linha do *dataset* omitindo entradas nas *arrays* e o arquivo original HTML:


```
{
   “annotations”: [{
      {
         “annotation_id”: 5015853435362506856,
         “long_answer”: {
            “candidate_index”: 92,
            “end_byte”: 67824,
            “end_token”: 925,
            “start_byte”: 66429,
            “start_token”: 808
         },
         “short_answers”: [{
            “end_byte”: 66609,
            “end_token”: 819,
            “start_byte”: 66595,
            “start_token”: 817
         }],
         “yes_no_answer”: “NONE”
      }
   ],
   “document_html”: “ … “,
   “document_title”: “Therefore sign”,
   “document_tokens”: [{
         “end_byte”: 101,
         “html_token”: false,
```

```
            “start_byte”: 92,
            “token”: “Therefore”
        }, …
    ],
    “question_text”: “where does the last name painter come from”,
    “question_tokens”: [
        “Where”, …    ]
}
```


Para conseguir ler esse arquivo, foi preciso utilizar alguns utilitários fornecidos pelo próprio BERT para transformar cada linha em um exemplo/features esperado pelo BERT. Para isso, primeiro separamos o processo de extrair a pergunta do processo de extrair as respostas. Criamos uma lista com as possíveis respostas de cada pergunta, recuperando o texto da página HTML por meio dos índices indicados em cada resposta, nos casos que não há respostas curtas, simplesmente não adicionamos isso no objeto que representa uma entrada (NQEntry). Uma facilidade na hora de fazer a extração é a *tag* html_token, que permite saber se dado *token* é texto ou HTML.

Chamamos de entrada pois antes de virar um exemplo que o BERT consiga utilizar, temos que passar pelo processo de tokenização do texto extraído, bem como limpar de caracteres indesejados e manter também alguns identificadores, para que depois do treinamento e avaliação consigamos recuperar as entradas originais. A partir deste momento, o fluxo fica parecido com o do Squad, transformando os exemplos para InputFeatures como já explicado e dando isso de entrada para o preditor do TensorFlow configurado e instanciado com o BERT.

4.1.3 Ajuste fino

Para realizar as tarefas de ajuste fino foram utilizadas algumas ferramentas, o ajuste fino foi realizado utilizando Python e algumas bibliotecas como o *tensorflow, numpy*, *transformers*, *datasets* e deepspeed. Tensorflow é uma plataforma para codificar, treinar e analisar modelos de aprendizagem de máquinas, importante notar que é código aberto e mantido pela Google.

Numpy é uma biblioteca com diversas funções e tipos de dados numéricos, dando suporte para matrizes grandes e/ou multidimensionais. No nosso contexto

usamos para lidar com as matrizes durante a preparação dos dados para dar como entrada ao modelo no tensorflow.

HuggingFace disponibiliza diversas APIs para Deep Learning, entre elas as utilizadas aqui são a de transformers e *datasets*, para recuperar da internet os modelos bases e os conjuntos de dados, execução do ajuste fino e submissão de perguntas aos conjuntos de arquivos testes.

A biblioteca *DeepSpeed*[4] foi o que nos permitiu reduzir o tempo de ajuste fino do Squad de 33 horas para 6 horas, e o tempo de ajuste fino do NaturalQuestions de 96 horas para 11horas. Isso é extremamente importante, dado que a nuvem da Google não permite execuções com tempo maior de 12 horas na versão grátis e mais de 24 horas na versão paga.

4.1.4 Avaliação nos *datasets*

No final do processo de ajuste fino preliminar teremos dois modelos mudando o *dataset* em que foram ajustados. Agora para a parte final da tarefa, exportamos este modelo e carregamos ele numa aplicação simples que fica em laço esperando uma entrada do usuário. O usuário, por motivos de simplicidade, pode fazer duas operações. Enviar um arquivo para o seu repositório de dados ou enviar uma pergunta em texto simples.

No caso de enviar um arquivo, fazemos a indexação deste arquivo, atualizando a entrada do modelo. Este processo de forma mais detalhada consiste primeiro em identificar o tipo do arquivo, aqui aceitamos para simplificar apenas PDF e TXT, mas isto é facilmente escalável para outros tipos de documentos. Continuando o processo, esse texto então passa por um processo de tokenização, dependendo do seu tamanho. Após isso, fazemos a indexação deste texto, caso já tenha mais textos no repositório, o processo de indexação parte do resultado de indexação anterior.

Para facilitar a criação de indexação foi utilizado uma biblioteca em Python chamada *Whoosh*[5], que facilita a criação de um motor de busca. Usamos a funcionalidade de criar índice que por padrão utiliza o algoritmo Okapi BM25F. Então guardamos esse índice localmente para aguardar uma eventual busca do usuário.

[4] DeepSpeed Microsoft: https://github.com/microsoft/DeepSpeed
[5] Whoosh: https://github.com/mchaput/whoosh

No caso de o usuário enviar uma pergunta, primeiro avaliamos se há um índice previamente criado. Então, primeiro utilizamos um utilitário da biblioteca *ktrain* para tokenizar a pergunta, então criamos uma *query* para o *index* e retornamos diversos documentos que possam ter a resposta, previamente ranqueados com o algoritmo padrão do Whoosh. Ranqueamos os documentos primeiro dividindo os documentos em parágrafos, chamando tais parágrafos de contexto e guardamos um ID para saber recuperar o arquivo original.

Assim, usando esses contextos e a pergunta, construímos uma tupla de entrada para o modelo. Modelo retorna uma lista de possíveis respostas, sendo que cada resposta tem um valor de confiança e uma referência para o contexto original. Com esta lista em mãos reordenamos baseado na confiança, recuperamos o contexto original e dele recuperamos o arquivo original que possivelmente contém a resposta. Por fim, apresentamos isso para o usuário.

O primeiro modelo resultante desta versão preliminar é o modelo BERT-base ajustado ao *dataset Squad*. O resultado obtido, dado a configuração do modelo, é comparativo com o resultado original no Squad v1.1 que foi 85.1 de EM e 91.8 de F1 usando a configuração BERT_large(Devlin et al., 2018). Agora, o mesmo modelo sem ter sido ajustado para o NaturalQuestion, obteve 42% de EM e 56% de F1, o que é um desempenho relativamente inferior ao desempenho mais alto até a presente data de acordo com a tabela de melhores modelos curada pela própria Google, criadora do *dataset* Natural Question. Entretanto vale ressaltar que este resultado é no estilo *ZeroShot*, o modelo nunca havia recebido nenhum exemplo do *dataset* NaturalQuestion antes.

O segundo modelo é análogo ao anterior, com a diferença que o ajuste fino foi realizado ao *dataset NaturalQuestions* e então submetido à validação nos dois *datasets*. No *NaturalQuestions*, o resultado obtido foi de 59% EM e 73%F1. Ficando relativamente próximo dos resultados obtidos pelos modelos com maior desempenho. Até a presente data (23/10/2023), o melhor modelo é *PoolingFormer*, com um desempenho de 79.8% F1. O que de uma certa forma era esperado, já que nos melhores modelos, muitos utilizam o BERT de base, mas em configurações superiores (BERT_large).

Agora, no *dataset Squad*, o desempenho foi análogo ao ZeroShot do primeiro modelo com 42% de EM e 62% de F1. Este desempenho deve ser melhorado quando ajustarmos o modelo no *dataset* também para que não seja mais uma validação

ZeroShot. De qualquer forma, esse desempenho em ZeroShot é promissor para o objetivo do modelo, que é lidar com textos nunca antes vistos pelo modelo. A seguir uma tabela com os resultados dos modelos preliminares

Tabela 2 - Comparação modelos preliminares.

Dataset	***Squad***		***Natural Questions***	
Modelo/Métricas	F1 (%)	EM (%)	F1 (%)	EM (%)
bert_base_finetuned_squad	60.96	78.36	56.21	42.03
bert_base_finetuned_nq	62.88	42.35	59.57	73.05

Fonte: Elaborada pelo autor (2023).

4.1.5 Melhorias

Na próxima iteração são exploradas diferentes formas de criar um índice. Outra possibilidade a ser explorada é a evolução da base de documentos utilizando os documentos fornecidos pelo usuário, atualizando o índice e aplicando novamente o ajuste fino ao novo índice. Além disso, faremos o ajuste fino como já mencionado em 2 *datasets*, exaurindo as diferentes combinações de ordem, dado que a ordem dos *datasets* no ajuste fino importa, como é demonstrado nos resultados do modelo final.

Outra conclusão é que apenas o BERT puro com ajuste fino em algum *dataset* de *Question Answering* já nos permite um desempenho considerável, com pouco custo computacional, se comparado ao processo de pré-treinamento, pois precisamos fazer apenas o ajuste fino. Por fim, demonstramos que é possível utilizar mais de um *dataset*, basta incluí-lo no processo de tokenização para converter pro padrão esperado, assim poderemos construir um sistema que tenha como treinamento base vários *datasets*.

Além disso, ainda podemos também usar os *datasets* como conjunto de arquivos unidos aos arquivos do usuário, enriquecendo a base de conhecimento. Com isso, entregamos ao usuário uma aplicação que, se o usuário desejar, contenha informação de outras fontes, além dos arquivos compartilhados pelo próprio usuário.

4.2 MODELO FINAL

Nesta segunda iteração as principais alterações são o processo de ajuste fino e de indexação. O processo de ajuste fino é alterado para ser aplicado em todos os *datasets* e não só em um único. O objetivo é conseguir desenvolver as qualidades que cada *dataset* requer para um bom desempenho. Como temos 2 *datasets* escolhidos e a ordem em que o ajuste fino é feito em cada *dataset* importa, teremos 2 modelos treinados e ajustados prontos para serem avaliados na tarefa deste trabalho, além dos 2 modelos base, totalizando 4 modelos produzidos neste trabalho.

O processo de indexação terá uma abordagem mais agressiva, utilizando os 2 *datasets* como fonte de consulta adicional aos arquivos entregues pelo usuário. O objetivo é ter um modelo mais generalista e que consiga desempenhar de forma satisfatória em qualquer contexto. Desta forma, um modelo rodando sem nenhum arquivo do usuário no repositório conseguiria pelo menos responder as perguntas dos *datasets* inclusos na indexação.

A dificuldade foi reaplicar o ajuste fino aos *datasets* pois cada um tem sua peculiaridade. Por isso, entramos um pouco mais em detalhe de cada *dataset*, sua estrutura, seus pontos fortes e fracos, seus objetivos e como foi utilizá-los.

4.2.1 Configuração

Os parâmetros de configuração do modelo permanecem o mesmo para as versões finais. O motivo disso é comparar se aplicar dois ajustes com *datasets* diferentes permite uma melhoria no desempenho dos modelos resultantes. Então se compararmos modelos com parâmetros diferentes e ajuste fino diferentes, não conseguiremos isolar o motivo de melhora ou piora no desempenho.

4.2.2 Preparação dos *datasets*

A tokenização dos *datasets* é igual ao primeiro modelo de cada *dataset*, pois já foi feito de tal forma que o resultado pós tokenização dos dois *datasets* seja igual. Isso não foi uma decisão tomada assim como na configuração para equalizar os modelos. O ferramental utilizado para realizar o ajuste fino obriga os dados a serem estruturados da mesma maneira antes de serem utilizados. Mais detalhes sobre o formato esperado estão descritos na seção preparação de *datasets* do modelo preliminar. Entretanto,

esse requisito da ferramenta permite que consigamos isolar ainda mais o impacto de um duplo ajuste fino, dado que até o formato dos dados são iguais.

4.2.3 Ajuste fino conjunto

O desafio foi conseguir realizar o ajuste fino nos dois *datasets*. Aqui temos duas formas de fazer tal tarefa, uma é carregar os dois conjuntos de dados para alguma abstração em comum ou de certa forma salvar os parâmetros resultantes do ajuste fino em um *dataset* para então usar isso como ponto de partida para um segundo ajuste fino. A segunda estratégia foi escolhida por alguns motivos: recursos limitados (um conjunto de dados unificados demandaria muito mais processamento); complexidade de unir os 2 *datasets* para um mesmo tipo de objeto; complexidade de separar as predições unificadas para um *dataset* específico.

Como explicado na seção de preparação dos *datasets*, o processo de tokenização resolveria os dois últimos problemas mencionados aqui, mas ainda resta o problema de recursos limitados. Simulações para descobrir o tempo estimado de execução dos modelos foram executadas apenas para saber os limites de configuração do modelo e tamanho do dado. A ferramenta utilizada para conseguir rodar esses ajustes finos e simulações foi o Google Colab que tem acesso a máquinas virtuais com GPUs poderosas, memórias RAMs enormes e memórias de disco maiores ainda. Ainda assim, não foi o suficiente para podermos utilizar a primeira estratégia, pois o Google Colab tem uma limitação de tempo de uso, sendo 12 horas para a versão grátis/estudante e 24 horas para a versão paga.

Por exemplo, o ajuste fino no *dataset Squad* utilizando a ferramenta *DeepSpeed* para acelerar o processamento e usar os recursos disponíveis (GPU 16 GB, RAM 32 GB, 240 GB SSD) ao máximo ainda teve uma duração total de 8 horas, enquanto o ajuste fino no *dataset NaturalQuestion* teve uma duração total de 14 horas. Vale ressaltar que sem o uso da ferramenta *DeepSpeed* esses tempos foram ridiculamente maiores, sendo o pior o *NaturalQuestions* com um tempo estimado de execução de 108 horas. Se fossemos realizar o ajuste dos dois *datasets* em uma única execução ultrapassaríamos o tempo máximo ou os recursos disponíveis. Dessa forma, tivemos que nos preocupar em salvar o ajuste e depois partir o segundo ajuste do primeiro.

Primeiro realizamos o processo de ajuste fino no *dataset Squad* como descrito na seção do modelo preliminar. Submetemos esse modelo à avaliação do SQUAD para manter um histórico, bem como ao *Natural Questions*, sendo que esses resultados são os dos modelos preliminares. O próximo passo foi salvar o modelo intermediário antes de ter o segundo ajuste realizado, para isso utilizamos novamente uma API do *HuggingFace* que permite salvarmos o modelo num repositório remoto gratuito, sendo obrigatória a autenticação no repositório que por sua vez requer uma conta criada no site da API.

Agora para realizar o segundo ajuste fino, a principal alteração é mudar o modelo de partida no script, ao invés de partir de um modelo cru do BERT, é realizado o login no repositório de modelos e alterado o modelo usado para o modelo salvo remotamente no primeiro ajuste fino. Então realizamos o ajuste fino do mesmo modelo no *Natural Questions dataset*. Salvamos esse modelo também no repositório de modelos, alteramos o script de avaliação para realizar o login no repositório e buscar o nosso modelo. Por fim, submetemos novamente a avaliação dos dois *datasets* e guardamos essa informação para análise. Então realizamos o mesmo processo invertendo o *dataset* inicial de ajuste fino, agora sendo *Natural Questions* o primeiro e SQUAD o segundo.

4.2.4 Avaliação com os *datasets*

Dado o processo de ajuste fino descrito, temos dois modelos resultantes, além dos dois modelos preliminares. O primeiro é o modelo ajustado ao *dataset* Squad e então ao *dataset Natural Questions*. O segundo é o modelo ajustado ao *dataset Natural Questions* e então ao *dataset Squad*. Os modelos resultantes são submetidos ao mesmo processo de avaliação que os modelos preliminares. Os resultados do primeiro modelo e segundo modelo estão na tabela abaixo, para facilitar a comparação os modelos preliminares também estão na Tabela 3.

Tabela 3 - Resultados dos modelos e resultados de trabalhos relacionados.

	Squad		***Natural Questions***	
Modelo/Métricas	F1 (%)	EM (%)	F1 (%)	EM (%)
finetuned_squad	78.36	60.96	56.21	42.03
finetuned_nq	62.88	42.35	73.05	59.57
Squad_NaturalQuestion	70.50	50.18	75.93	62.43
NaturalQuestion_Squad	77.87	60.25	74.42	60.80
Resultados dos trabalhos relacionados				
Retrieve and Read	39.8	32.7	X	X
BERT	92.8	86.7	X	X
ORQA	X	33.2	X	33.3
REALM	X	X	X	40.4
Reading Comprehension	67.34	59.14	X	X

Fonte: Elaborada pelo autor (2023).

Como vimos ao longo deste texto, o *dataset Squad* costuma exigir a capacidade de representação de dados maior, ou seja, quanto mais parâmetros o modelo, maior o desempenho do modelo no *dataset*. Ou seja, um modelo que seja bom de decorar os dados durante o treinamento irá apresentar desempenho melhor, enquanto o *Natural Question* exige a capacidade de compreender o contexto para encontrar a resposta, já que suas perguntas nem sempre têm resposta no texto e os modelos têm que saber identificar a ausência de resposta também.

Podemos observar que os modelos ajustado ao *dataset Squad* em algum momento do processo tiveram melhor desempenho no *dataset Natural Question*. Entende-se que a capacidade exigida pelo *dataset Squad* faz com que o modelo consiga capturar uma representação melhor em seus milhões de parâmetros. Porém, tais parâmetros não necessariamente resultam em um modelo mais generalista ou se resultam, não são observados quando submetidos ao *Squad*.

Agora, quando os modelos com ajuste finos em dois *datasets* são submetidos ao *Natural Question*, o desempenho foi consideravelmente melhor. Podemos concluir que a generalização do modelo beneficiou o resultado em tal *dataset* que justamente requer uma capacidade de, forma limitada, compreender o contexto.

Se compararmos os modelos deste trabalho com os trabalhos relacionados, vemos que tivemos desempenho superior a todos eles menos o BERT original que foi usado na sua configuração mais poderosa (LARGE), enquanto aqui usamos a configuração intermediária (BASE). É argumentável que a diferença de resultados entre nosso modelo e os modelos de trabalhos relacionados vem do fato que os trabalhos relacionados foram feitos a alguns anos e que na época não havia ferramentais elaborados como os que foram utilizados aqui nesse trabalho.

Os resultados alcançados são relevantes e melhores do que o esperado pois conseguimos demonstrar que o ajuste fino em diferentes *datasets* pode sim melhorar o desempenho, dependendo da natureza de tais *datasets*. Se compararmos com nossa baseline que é o BERT, ainda tivemos um desempenho muito inferior. Mas isso se deve ao tamanho do modelo usado, que neste caso foi um intermediário (BASE).

4.2.5 Avaliação Zero *Shot*

Utilizamos o *dataset* 20 newsgroup[6] para a avaliação em formato zero shot devido ao seu tamanho reduzido em relação aos *datasets* utilizados pelos trabalhos relacionados. Uma avaliação no formato ZeroShot quer dizer que o modelo sendo avaliado nunca viu os dados sobre os quais está sendo avaliado, desta forma tornando a tarefa relativamente mais difícil, pois o modelo não foi treinado neste conjunto de dados. De qualquer forma, o *dataset* 20 newsgroups já traz consigo o desafio de contexto e domínio aberto. Para a baseline é utilizado o modelo BERT-large apresentado por Devlin *et al.* (2019), pois comparado aos outros trabalhos relacionados foi o que de longe teve um melhor desempenho.

4.2.5.1 Conjunto News20_group

Nesta seção são apresentados os resultados dos 2 modelos finais criados neste trabalho e o resultado do modelo BERT-large que foi treinado no Squad como descrito por Devlin *et al.* (2019) quando submetidos a perguntas em texto natural sobre o conjunto de dados News20_group. As imagens a seguir mostram os resultados dos

[6] Disponível em: https://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html. Acesso em: 13/04/2023.

diferentes modelos em diferentes perguntas aleatórias criadas para validar a capacidade ZeroShot. Detalhe, as perguntas não necessariamente tem respostas nos dados fornecidos.

Pergunta 1: When did the Cassini probe launch?

BERT-large

	Candidate Answer	Context	Confidence	Document Reference
0	in october of 1997	cassini is scheduled for launch aboard a titan iv / centaur in october of 1997 .	0.819032	1257
1	on january 26,1962	ranger 3, launched on january 26,1962 , was intended to land an instrument capsule on the surface of the moon, but problems during the launch caused the probe to miss the moon and head into solar orbit.	0.151230	9591
2	- 10 / 06 / 97	key scheduled dates for the cassini mission (vvejga trajectory)----------10 / 06 / 97-titan iv / centaur launch 04 / 21 / 98-venus 1 gravity assist 06 / 20 / 99-venus 2 gravity assist 08 / 16 / 99-earth gravity assist 12 / 30 / 00-jupiter gravity assist 06 / 25 / 04-saturn arrival 01 / 09 / 05-titan probe release 01 / 30 / 05-titan probe entry 06 / 25 / 08-end of primary mission (schedule last updated 7 / 22 / 92) - 10 / 06 / 97	0.029694	1257
3	* 98	cassini * * * * * * * * * * * * * * * * * * * 98 ,115 * * * *	0.000026	12775
4	the latter part of the 1990s	scheduled for launch in the latter part of the 1990s , the craf and cassini missions are a collaborative project of nasa, the european space agency and the federal space agencies of germany and italy, as well as the united states air force and the department of energy.	0.000017	16868

SquadNq

	Candidate Answer	Context	Confidence	Document Reference
0	in october of 1997	cassini is scheduled for launch aboard a titan iv / centaur in october of 1997 .	0.925283	1257
1	on 2 / 20 / 93	+ asuka (astro d)-isas (japan) x ray astronomy satellite, launched into + earth orbit on 2 / 20 / 93 .	0.065131	12775
2	on january 26,1962	ranger 3, launched on january 26,1962 , was intended to land an instrument capsule on the surface of the moon, but problems during the launch caused the probe to miss the moon and head into solar orbit.	0.008563	9591
3	in october 1989, will investigate main belt asteroids on its way to jupiter. the comet rendezvous / asteroid flyby (craf) and cassini missions will also study these far flung objects. scheduled for launch in the latter part of the 1990s	the galileo orbiter, launched by nasa in october 1989, will investigate main belt asteroids on its way to jupiter. the comet rendezvous / asteroid flyby (craf) and cassini missions will also study these far flung objects. scheduled for launch in the latter part of the 1990s , the craf and cassini missions are a collaborative project of nasa, the european space agency and the federal space agencies of germany and italy, as well as the united states air force and the department of energy.	0.000998	16868
4	in 2001	o fire and ice possible launch in 2001 , will use a gravity assist flyby of earth in 2003, and use a final gravity assist from jupiter in 2005, where the probe will split into its fire and ice components : the fire probe will journey into the sun, taking measurements of our star ' s upper atmosphere until it is vaporized by the intense heat.	0.000015	1257

NqSquad

	Candidate Answer	Context	Confidence	Document Reference
0	in october 1989	the galileo orbiter, launched by nasa in october 1989 , will investigate main belt asteroids on its way to jupiter.	0.657163	16868
1	in october of 1997	cassini is scheduled for launch aboard a titan iv / centaur in october of 1997 .	0.178742	1257
2	on january 26,1962	ranger 3, launched on january 26,1962 , was intended to land an instrument capsule on the surface of the moon, but problems during the launch caused the probe to miss the moon and head into solar orbit.	0.090151	9591
3	on 2 / 20 / 93	+ asuka (astro d)-isas (japan) x ray astronomy satellite, launched into + earth orbit on 2 / 20 / 93 .	0.039734	12775

Pergunta 2: When did Ayrton Senna die?

BERT-large

	Candidate Answer	Context	Confidence	Document Reference
0	when did ayrton senna die ? if ayrton senna can drive a racecar with fully automatic transmission, it ca not be half bad.. : -)	: -) when did ayrton senna die ? if ayrton senna can drive a racecar with fully automatic transmission, it ca not be half bad.. : -)	0.557836	10331
1	hardly	hardly .	0.442148	7544
2	! i am darkness eternal !	bitnet	rochester ny 14623	die ! !----------	die ! ! i am darkness eternal ! call me ! my phone number is (716) 475-4197	die ! !	0.000016	4645

SquadNq

	Candidate Answer	Context	Confidence	Document Reference
0	: christ died	this would make sense : christ died , and his mother, who waited at the foot of the cross, would want to share in his death.	0.583499	6295
1	? the one where they lived, peacibly, to all known purposes (until proven in court	whose " cause " did they die for ? the one where they lived, peacibly, to all known purposes (until proven in court , folks !), or the cuase of righteous government safe guarding the freedom of the children	0.177488	7119
2		die ! !		e mail :	robert voss	die ! ! especially with	rgv9488 @ ultb.	0.094057	4645
3	in vietnam war	maybe also vietamiese did not die in vietnam war killed by american napalm they were just pyromaniacs and that ' s all.	0.094057	18627
4	in vietnam war	maybe also vietamiese did not die in vietnam war killed by american napalm they were just pyromaniacs and that ' s all.	0.027451	13095

NqSquad

	Candidate Answer	Context	Confidence	Document Reference
0	: christ	this would make sense : christ died, and his mother, who waited at the foot of the cross, would want to share in his death.	0.438385	6295
1	? the one where they lived, peacibly, to all known purposes	whose " cause " did they die for ? the one where they lived, peacibly, to all known purposes (until proven in court, folks !), or the cuase of righteous government safe guarding the freedom of the children	0.438385	7119
2	in vietnam war	maybe also vietamiese did not die in vietnam war killed by american napalm they were just pyromaniacs and that ' s all.	0.104870	13095
3	in vietnam war	maybe also vietamiese did not die in vietnam war killed by american napalm they were just pyromaniacs and that ' s all.	0.016465	18627

Pergunta 3: What is Gamma Ray Bursts?

	Candidate Answer	Context	Confidence	Document Reference
0	are a necessary consequence of the general unified theory of the physical universe	gamma ray bursts are a necessary consequence of the general unified theory of the physical universe developed by the late physicist dewey b.	0.484715	6897
1	are a necessary consequence of the general unified theory of the physical universe	gamma ray bursts are a necessary consequence of the general unified theory of the physical universe developed by the late physicist dewey b.	0.484715	5928
2	perhaps some of the mass extinctions of the past, which are now being blamed on impacts of comets and asteroids, were actually caused by nearby gamma ray bursts !	perhaps some of the mass extinctions of the past, which are now being blamed on impacts of comets and asteroids, were actually caused by nearby gamma ray bursts !	0.014496	6897
3	perhaps some of the mass extinctions of the past, which are now being blamed on impacts of comets and asteroids, were actually caused by nearby gamma ray bursts !	perhaps some of the mass extinctions of the past, which are now being blamed on impacts of comets and asteroids, were actually caused by nearby gamma ray bursts !	0.014496	5928
4	from supernova explosions in the anti matter half of the physical universe	larson ' s explanation is that the gamma ray bursts originate from supernova explosions in the anti matter half of the physical universe , which larson calls the " cosmic sector ".	0.001456	15997

SquadNq

	Candidate Answer	Context	Confidence	Document Reference
0	" gravisphere	what on earth is the " gravisphere " ? anyway, before it ' s decay the pioneer venus orbiter had a gamma ray detector, as does ulysses, they detect the brightest bursts that the earth orbit detectors do, so the bursts are at least at oort cloud distances.	0.399030	5613
1	from supernova explosions in the anti matter half of the physical universe	larson ' s explanation is that the gamma ray bursts originate from supernova explosions in the anti matter half of the physical universe , which larson calls the " cosmic sector ".	0.399029	15997
2	are a necessary consequence of the general unified theory of the physical universe	gamma ray bursts are a necessary consequence of the general unified theory of the physical universe developed by the late physicist dewey b.	0.135442	6897
3	are a necessary consequence of the general unified theory of the physical universe	gamma ray bursts are a necessary consequence of the general unified theory of the physical universe developed by the late physicist dewey b.	0.057121	5928
4	perhaps some of the mass extinctions of the past, which are now being blamed on impacts of comets and asteroids	perhaps some of the mass extinctions of the past, which are now being blamed on impacts of comets and asteroids , were actually caused by nearby gamma ray bursts !	0.004387	6897

NqSquad

	Candidate Answer	Context	Confidence	Document Reference
0	the mysterious gamma ray bursts. they were originally thought to originate from " neutron stars " in the disc of our galaxy	astro physicists and astronomers are still scratching their heads about the mysterious gamma ray bursts. they were originally thought to originate from " neutron stars " in the disc of our galaxy .	0.317934	6897
1	the mysterious gamma ray bursts. they were originally thought to originate from " neutron stars " in the disc of our galaxy	astro physicists and astronomers are still scratching their heads about the mysterious gamma ray bursts. they were originally thought to originate from " neutron stars " in the disc of our galaxy .	0.317932	5928
2	from supernova explosions in the anti matter half of the physical universe	larson ' s explanation is that the gamma ray bursts originate from supernova explosions in the anti matter half of the physical universe , which larson calls the " cosmic sector ".	0.180998	15997
3	now puzzling orthodox physicists and astronomers, such as gamma ray bursts and the nature of quasars	larson ' s theory has excellent explanations for many things now puzzling orthodox physicists and astronomers, such as gamma ray bursts and the nature of quasars .	0.042772	6897
4	now puzzling orthodox physicists and astronomers, such as gamma ray bursts and the nature of quasars	larson ' s theory has excellent explanations for many things now puzzling orthodox physicists and astronomers, such as gamma ray bursts and the nature of quasars .	0.042772	5928

Pergunta 4: What is XFree86 ?

	Candidate Answer	Context	Confidence	Document Reference
0	binaries are available via anonymous ftp from : ftp. physics. su. oz. au-svr4	binaries are available via anonymous ftp from : ftp. physics. su. oz. au-svr4 binaries under / xfree86 / svr4 ftp.	0.303914	3992
1	binaries are available via anonymous ftp from : ftp. physics. su. oz. au-svr4	binaries are available via anonymous ftp from : ftp. physics. su. oz. au-svr4 binaries under / xfree86 / svr4 ftp.	0.303914	17811
2	be isa, eisa, or local bus	it could be isa, eisa, or local bus .	0.175457	3992
3	be isa, eisa, or local bus	it could be isa, eisa, or local bus .	0.175457	17811
4	i) what options do i have for x software on my intel based unix system ?	i) what options do i have for x software on my intel based unix system ? 1.	0.011641	17811

SquadNq

	Candidate Answer	Context	Confidence	Document Reference
0	binaries are available via anonymous ftp from : ftp. physics. su. oz. au-svr4 binaries	binaries are available via anonymous ftp from : ftp. physics. su. oz. au-svr4 binaries under / xfree86 / svr4 ftp.	0.459319	17811
1	binaries are available via anonymous ftp from : ftp. physics. su. oz. au-svr4 binaries	binaries are available via anonymous ftp from : ftp. physics. su. oz. au-svr4 binaries under / xfree86 / svr4 ftp.	0.459319	3992
2	) ftp. win. tue. nl	au (under / xfree86) ftp. win. tue. nl (under / pub / xfree86) (for the rest of this faq, these 3 location will be called $ ftp)	0.036742	17811
3	) ftp. win. tue. nl	au (under / xfree86) ftp. win. tue. nl (under / pub / xfree86) (for the rest of this faq, these 3 location will be called $ ftp)	0.036742	3992
4	? 1. free options 2. commercial options ii) what is xfree86 and where do i get it	i) what options do i have for x software on my intel based unix system ? 1. free options 2. commercial options ii) what is xfree86 and where do i get it ? 3.	0.001942	3992

NqSquad

	Candidate Answer	Context	Confidence	Document Reference
0	free options 2. commercial options ii) what is xfree86 and where do i get it	free options 2. commercial options ii) what is xfree86 and where do i get it ? 3.	0.331823	17811
1	free options 2. commercial options ii) what is xfree86 and where do i get it	free options 2. commercial options ii) what is xfree86 and where do i get it ? 3.	0.331823	3992
2	- svr4 binaries	edu-mach binaries under / i386 - svr4 binaries	0.149770	17811
3	- svr4 binaries	edu-mach binaries under / i386 - svr4 binaries	0.149770	3992
4	be isa, eisa, or local bus	it could be isa, eisa, or local bus .	0.013974	17811

Pergunta 5: What are Gaucher`s disease symptoms?

	Candidate Answer	Context	Confidence	Document Reference
0	(eg chest complaints	my wife has become interested through an acquaintance in post polio syndrome this apparently is not recognised in new zealand and different symptons (eg chest complaints) are treated separately.	0.613429	12569
1	i beg to differ. chronic * active * hepatitis	i beg to differ. chronic * active * hepatitis implies that the disease remains active, and generally leads to liver failure.	0.219886	15675
2	i have a friend who has just been diagnosed with lupus	i have a friend who has just been diagnosed with lupus , and i know nothing about this disease.	0.143485	1331
3	i ' m not sure that this condition	i ' m not sure that this condition is " recognised " anywhere (in the sense of a disease with diagnostic criteria, clear boundaries between it and other diseases, unique pathologic or physiologic features, etc), but here goes with what many neurologists agree on.	0.011612	9486
4	> chest complaints	nz (keith stewart) ks > my wife has become interested through an acquaintance in post polio syndrome ks > this apparently is not recognised in new zealand and different symptons (eg ks > chest complaints) are treated separately.	0.007138	9486

SquadNq

	Candidate Answer	Context	Confidence	Document Reference
0	: brittle bones	gaucher ' s disease symptoms include : brittle bones (he lost 9 inches off his hieght) ; enlarged liver and spleen ; internal bleeding ; and fatigue (all the time).	0.909430	8700
1	have no symptoms of hiv disease	volunteers must be hivinfected but have no symptoms of hiv disease .	0.047254	14403
2	of dizziness	if a patient complains of dizziness , faintness, sweating, palpitations, etc.	0.037895	4851
3	having osteopporosis	i have a 42 yr old male friend, misdiagnosed as having osteopporosis for two years, who recently found out that his illness is the rare gaucher ' s disease.	0.001626	8700
4	to loss of memory and mental functioning	as the disease progresses, it leads to loss of memory and mental functioning , followed by changes in personality, loss of control of bodily functions, and, eventually, death.	0.001537	6815

NqSquad

	Candidate Answer	Context	Confidence	Document Reference
0	: brittle bones	gaucher ' s disease symptoms include : brittle bones (he lost 9 inches off his hieght) ; enlarged liver and spleen ; internal bleeding ; and fatigue (all the time).	0.991742	8700
1	to loss of memory and mental functioning	as the disease progresses, it leads to loss of memory and mental functioning , followed by changes in personality, loss of control of bodily functions, and, eventually, death.	0.004536	6815
2	have no symptoms of hiv disease	volunteers must be hivinfected but have no symptoms of hiv disease .	0.002143	14403
3	having osteopporosis	i have a 42 yr old male friend, misdiagnosed as having osteopporosis for two years, who recently found out that his illness is the rare gaucher ' s disease.	0.000734	8700
4	of dizziness	if a patient complains of dizziness , faintness, sweating, palpitations, etc.	0.000700	4851

Para melhorar a visualização e entendimento dos resultados a Tabela 4 a seguir foi criada.

Tabela 4.

Modelos/Perguntas	BERT-large	SquadNq	NqSquad
1	Acerto	Acerto	Acerto
2	Falha	Falha	Falha
3	Acerto	Acerto	Acerto
4	Falha	Falha	Falha
5	Acerto	Acerto	Acerto

Fonte: Elaborada pelo autor (2023).

Pela tabela acima, podemos observar que os modelos obtiveram os mesmos resultados, mudando o intervalo de confiança e/ou a ordem com qual a resposta correta apareceu. Isso é promissor para arquitetura, já que não houve perda de desempenho em relação ao BERT-large, sendo que o esperado era perder pois o BERT-large é um modelo com centenas de milhões de parâmetros a mais que o BERT-base, que já tem mais de uma centena de milhões de parâmetros. A suposição que surge disso é: e se o BERT-large fosse usado como base da arquitetura ao invés do BERT-base, o desempenho melhoraria? A tendência baseada nos testes acima é que sim.

5 CONCLUSÃO

Este trabalho teve como objetivo propor um repositório de arquivos que consiga realizar buscas em linguagem natural. Para isso, o estado da arte e trabalhos relacionados foram analisados. Uma arquitetura foi proposta envolvendo o modelo BERT e o ranqueamento de arquivos Okapi BM25F. Os critérios de validação escolhidos foram as métricas F1 e EM nos *datasets* Natural Question e Squad v1.1. Desta forma, entende-se que todos os objetivos propostos foram cumpridos. Entretanto, a arquitetura proposta tem seus pontos fracos e pontos fortes.

Alguns pontos fortes foram identificados após a análise dos resultados, sendo que todos eles derivam da generalidade do sistema, que não se amarra a nenhum *dataset* específico e nenhum modelo específico. O código feito neste trabalho permite que qualquer modelo baseado em BERT publicado no repositório de modelos HuggingFace seja utilizado como base. Além disso, permite que qualquer *dataset* seja utilizado, contanto que o desenvolvedor forneça o processo de tokenização para transformar o *dataset* no padrão esperado. Permite também que diversas iterações de ajuste fino sejam realizadas, no mesmo *dataset* ou diferentes *datasets*, em diferentes ordens. Por isso, a maior contribuição do trabalho é a arquitetura proposta, pois ela pode ser utilizada para criar modelos generalistas.

O principal ponto fraco do trabalho é a capacidade de treinamento reduzida pela falta de acesso a recursos computacionais como os utilizados pelos autores do BERT para treinar a configuração LARGE. Conseguimos utilizar apenas a configuração BASE, que já foi suficiente para comprovar a hipótese de ajuste fino com 2 ou mais *datasets* pode melhorar o desempenho do modelo nas métricas escolhidas.

Outro ponto fraco do sistema, como já descrito na seção de resultados, é o algoritmo Okapi BM25F de ranqueamento probabilístico usado para pré-selecionar alguns arquivos antes de ranqueá-los com o modelo. O principal problema do algoritmo é que não consegue encontrar arquivos que utilizem sinônimos corretamente, apenas as palavras contidas na busca.

Essa arquitetura pode ser amplamente utilizada quando o objetivo for generalismo, entretanto se o desafio é lidar com algum tipo de conhecimento específico, outra arquitetura deve ser considerada. Como vimos nos resultados, o modelo obteve resultados consideráveis com o *dataset* generalista NaturalQuestion. Agora os resultados no *dataset* Squad v1.1, que requer uma maior especialização

maior, tiveram uma piora conforme novos ajustes finos foram realizados. Portanto, se o objetivo for lidar com conjuntos de dados genéricos, esta arquitetura se mostra viável.

5.1 TRABALHOS FUTUROS

O trabalho futuro mais óbvio para melhorar o desempenho é conseguir realizar o ajuste fino utilizando modelos com configuração do BERT_large. Para a fase de ajuste fino, seria interessante trabalhar com mais conjuntos de dados que exigem habilidades diferentes do modelo. Para a parte de avaliação de desempenho ZeroShot, novos *datasets* com domínios específicos deveriam ser criados ao estilo do *dataset* NaturalQuestion, para que possamos também validar de uma forma formal e não apenas empírica. A avaliação empírica é enviesada e de amostragem muito limitada, impedindo conclusões mais assertivas sobre os resultados alcançados. Outra possível parte da arquitetura a ser revisada é o primeiro ranqueamento que atualmente foi feito com o algoritmo Okapi BM25-F, entretanto todos os algoritmos da família TF-IDF tem o problema de conseguirem considerar só arquivos que contenham textos estatisticamente semelhante com a consulta. Para resolver este problema, podemos considerar o *Elastic Search* que vem sendo amplamente utilizado para toda a parte de indexação e recuperação de informação do índice.

REFERÊNCIAS

AHO, A. V.; ULLMAN, J. D. Chapter 10. Patterns, Automata, and Regular Expressions. *In*: AHO, A. V.; ULLMAN, J. D. **Foundations of Computer Science**. Stanford: Computer Science Press, 1992. p. 529-590.

BAHDANAU, D.; CHO, K.; BENGIO, Y. Neural Machine Translation by Jointly Learning to Align and Translate. In: International Conference on Learning Representations (ICLR 2015), 3rd, May 7-9, San Diego. **Proceedings** [...]. San Diego: [S. n.], 2015. p. 1-15. Disponível em: https://arxiv.org/pdf/1409.0473.pdf. Acesso em: 12 nov. 2023.

BEAULIEU, M. M. *et al*. Okapi at TREC-5. *In*: VOORHEES, E. M.; HARMAN, D. K. (ed.). THE FIFTH TEXT RETRIEVAL CONFERENCE (TREC-5), 5th, 1997, Gaithersburg. **Proceedings** [...]. Gaithersburg: National Institute of Standards and Technology, 1997. p. 143–165. (NIST Special Publication 500-238). Disponível em: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication500-238.pdf. Acesso em: 12 nov. 2023.

BENGIO, Y. Learning Deep Architectures for AI. **Foundations and Trends® in Machine Learning**, Hanover, v. 2, n. 1, p. 1-127, 2009. Disponível em: http://dx.doi.org/10.1561/2200000006. Acesso em: 3 set. 2022.

BET, S. **Aperfeiçoamento de algoritmo genético para seleção de variáveis de entrada para rede neural para previsão de carga elétrica ativa de curto prazo**. 2005. 65f. Dissertação (Mestrado em Ciência da Computação) - Universidade Federal de Santa Catarina, Florianópolis, 2005. Disponível em: https://repositorio.ufsc.br/bitstream/handle/123456789/102678/221624.pdf?sequence=1&isAllowed=y. Acesso em: 12 nov. 2023.

BURKE, R. *et al*. Question Answering from Frequently Asked Question Files: Experiences with the FAQ FINDER System. **AI Magazine**, Whashington, v. 18, n. 2, p. 57-66, 1997. Disponível em: https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/1294/1195. Acesso em: 12 nov. 2023.

CHEN, D. *et al*. Reading Wikipedia to answer open-domain questions. *In*: ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS (VOLUME 1: LONG PAPERS), 55th, 2017, Vancouver. **Proceedings** […]. Vancouver: Association for Computational Linguistics, 2017. p. 1870–1879.

CLARK, C.; GARDNER, M. Simple and effective multiparagraph reading comprehension. *In*: ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS (VOLUME 1: LONG PAPERS), 56th, 2018, Melbourne. **Proceedings** […]. Melbourne: Association for Computational Linguistics, 2018. p. 845–855.

DEVLIN, J. BERT. [2023]. Disponível em: https://github.com/google-research/bert. Acesso em: 12 nov. 2023.

DEVLIN, J. *et al*. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *In*: ANNUAL CONFERENCE OF THE NORTH AMERICAN CHAPTER OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS: HUMAN LANGUAGE TECHNOLOGIES (NAACL-HLT 2019), 17th, 2 a 7 jun. 2019, Minneapolis. **Proceedings** [...]. Minneapolis: NAACL-HLT, 2 a 7 jun. 2019. p. 4171-4186. Disponível em: https://arxiv.org/pdf/1810.04805.pdf. Acesso em: 12 nov. 2023.

GRAVES, A. Generating sequences with recurrent neural networks. 2013. p. 1-43. Disponível em: http://arxiv.org/abs/1308.0850. Acesso em: 12 nov. 2023.

GUU, K. *et al*. REALM: Retrieval-Augmented Language Model Pre-Training. **Arxiv**, New York, p. 1-12, 2020. Disponível em: https://arxiv.org/pdf/2002.08909.pdf. Acesso em: 12 nov. 2023.

JURAFSKY, D.; MARTIN, J. H. Question Answering and Information Retrieval. *In*: JURAFSKY, D.; MARTIN, J. H. **Speech and Language Processing**. 3rd ed. draft. [S.l.]: [s.n.], [2023]. p. 1-29. Disponível em: https://web.stanford.edu/~jurafsky/slp3/14.pdf. Acesso em: 12 nov. 2023.

KENTER, T. *et al*. Neural networks for information retrieval. *In*: INTERNATIONAL ACM SIGIR CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL, 40th, 2017, Shinjuku. **Proceedings** […]. Shinjuku: ACM, 2017. p. 1403–1406. Disponível em: https://dl.acm.org/doi/proceedings/10.1145/3077136. Acesso em: 12 nov. 2023.

KRATZWALD, B.; EIGENMANN, A.; FEUERRIEGEL, S. RankQA: Neural question answering with answer re-ranking. *In*: ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, 57th, 2019, Florence. **Proceedings** […]. Florence: Association for Computational Linguistics, 2019. p. 6076–6085.

KWIATKOWSKI, T. *et al*. Natural Questions: A Benchmark for Question Answering Research. **Transactions of the Association for Computational Linguistics**, Cambridge, v. 7, p. 452–466, 2019. Disponível em: https://storage.googleapis.com/pub-tools-public-publication-data/pdf/1f7b46b5378d757553d3e92ead36bda2e4254244.pdf. Acesso em: 12 nov. 2023.

MANNING, C. D.; RAGHAVAN, P.; SCHUTZE, H. **Introduction to Information Retrieval**. Cambridge: MIT Press, 2008.

MIKOLOV, T. *et al*. Efficient Estimation of Word Representations in Vector Space. **Arxiv**, New York, p. 1-12, 2013. Disponível em: https://arxiv.org/abs/1301.3781. Acesso em: 12 nov. 2023.

NISHIDA, K. *et al*. Retrieve-and-read: Multi-task learning of information retrieval and reading comprehension. *In*: ACM INTERNATIONAL CONFERENCE ON INFORMATION AND KNOWLEDGE MANAGEMENT, 27th, 2018, New York. **Proceedings** […]. Torino: ACM, 2018. p. 647–656.

OTTER, D. W.; MEDINA, J. R.; KALITA, J. K. A Survey of the Usages of Deep Learning for Natural Language Processing. **IEEE Transactions on Neural Networks and Learning Systems**, [S.l.], v. 32, n. 2, p. 604-624, fev. 2021. Disponível em: https://doi.org/10.1109/TNNLS.2020.2979670. Acesso em: 12 set. 2022.

POWERS, D. Preface: Goals, Issues and Directions in Machine Learning of Natural Language and Ontology. *In*: AAAI SPRING SYMPOSIUM ON MACHINE LEARNING OF NATURAL LANGUAGE AND ONTOLOGY, 1991, Stanford. **Proceedings** [...]. Stanford: Stanford University, 1991. p. 131-136. Disponível em: https://www.researchgate.net/publication/264122458_Preface_Goals_Issues_and_Directions_in_Machine_Learning_of_Natural_Language_and_Ontology. Acesso em: 12 nov. 2023.

RAJPURKAR, P. *et al*. SQuAD: 100,000+ Questions for Machine Comprehension of Text. *In*: SU, J.; DUH, K.; CARRERAS, X. (ed.). CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING, 2016, Austin. **Proceedings** [...]. Austin: Association for Computational Linguistics, 2016. p. 2383-2392. Disponível em: https://aclanthology.org/D16-1264/. Acesso em: 12 nov. 2023.

ROBERTSON, S. E. *et al*. Okapi at TREC-3. *In*: HARMAN, D. K. (ed.). OVERVIEW OF THE THIRD TEXT RETRIEVAL CONFERENCE (TREC-3), 3rd, 1995, Gaithersburg. **Proceedings** [...]. Gaithersburg: National Institute of Standards and Technology, 1995. p. 109–126. (NIST Special Publication 500-225). Disponível em: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication500-225.pdf. Acesso em: 12 nov. 2023.

ROBERTSON, S.; ZARAGOZA, H. The Probabilistic Relevance Framework: BM25 and Beyond. **Foundations and Trends in Information Retrieval**, Hanover, v. 3, n. 4, p. 333–389, 2009. Disponível em: https://doi.org/10.1561/1500000019. Acesso em: 1º fev. 2023.

SINCLAIR, John. Corpus and Text: Basic Principles. *In*: WYNNE, M. (ed.). **Developing Linguistic Corpora**: a Guide to Good Practice. [London]: [King's Digital Lab], 2005. (AHDS Guides to Good Practice). Disponível em: http://icar.cnrs.fr/ecole_thematique/contaci/documents/Baude/wynne.pdf. Acesso em: 11 nov. 2023.

SIPSER, M. Part One: Automata and Languages. *In*: SIPSER, M. **Introduction to the Theory of Computation**. [S.l.]: PWS Publishing, 1998. p. 29-119.

TRIM, C. The Art of Tokenization. **Language Processing**, [S. l.], 2013. Disponível em: https://trimc-nlp.blogspot.com/2020/11/the-art-of-tokenization.html. Acesso em: 12 nov. 2023.

TURC, I. *et al*. Well-Read Students Learn Better: On The Importance Of Pre-Training Compact Models. *In*: INTERNATIONAL CONFERENCE ON LEARNING REPRESENTATIONS (ICLR 2020), 8th, 2020, Addis Ababa. **Proceedings** [...]. Addis Ababa: [S. n.], 2020. Disponível em: https://openreview.net/forum?id=BJg7x1HFvB. Acesso em: 12 nov. 2023.

VASWANI, A. *et al*. Attention is all you need. *In*: GUION, I. *et al*. (ed.). CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS (NIPS 2017), 31st, 2017, Long Beach. **Proceedings** [...]. Long Beach: [S. n.], 2017. p. 6000-6010. Disponível em: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf. Acesso em: 12 nov. 2023.

WEISS, K.; KHOSHGOFTAAR, T. M.; WANG, D. A survey of transfer learning. **J Big Data**, [S. l.], v. 3, n. 9, 2016. Disponível em: https://doi.org/10.1186/s40537-016-0043-6. Acesso em: 12 nov. 2023.

ZHU, F. *et al*. Retrieving and Reading: A Comprehensive Survey on Open-domain Question Answering. **Arxiv**, New York, 2021. Disponível em: https://arxiv.org/pdf/2101.00774.pdf. Acesso em: 12 nov. 2023.

APÊNDICE A – SCRIPTS USADOS NESTE TRABALHO

Os *scripts* usados neste trabalho estão disponíveis em https://github.com/lschluter/TCC.

APÊNDICE B – ARTIGO FORMATO SBC

Recuperação de informação em documentos baseada em consultas de linguagem natural utilizando *Machine Learning*

Leonardo Schlüter Leite[1], Elder Rizzon Santos[1]

[1] Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
Campus Universitário Trindade – Caixa Postal. 476 / CEP 88040-370 - Florianópolis / SC

Abstract. *The developed project focuses on file retrieval and text extraction based on natural language queries, utilizing concepts and techniques from the field of Natural Language Processing (NLP. Fine-tuning, crucial for enhancing performance, proved to be a complex task due to the unique aspects of each dataset. The selected approach, despite challenges arising from limited computational resources, resulted in the creation of more generalized models. In spite of the encountered difficulties, this work showcased that fine-tuning across multiple datasets could substantially enhance the performance of NLP models in question-answering tasks, surpassing several related approaches.*

Resumo. *O projeto desenvolvido trata da recuperação de arquivos e da extração de texto baseado em uma pergunta de texto natural. Para realizar tal desafio, conceitos e técnicas da área de Processamento de Linguagem Natural (PLN) foram empregados. O ajuste fino, essencial para melhorar o desempenho, foi uma tarefa complexa devido às peculiaridades de cada conjunto de dados. A abordagem escolhida, embora desafiadora devido a recursos computacionais limitados, resultou em modelos mais generalistas. Apesar dos desafios, este trabalho demonstrou que o ajuste fino em múltiplos conjuntos de dados pode levar a melhorias significativas no desempenho dos modelos de PLN em tarefas de questionamento, proporcionando resultados superiores a várias abordagens relacionadas.*

1. Introdução

Uma área em crescimento nos últimos anos é a de Processamento de Linguagem Natural (PLN, ou Natural Language Processing – NLP). O PLN abrange uma variedade de tarefas relacionadas à compreensão da linguagem humana (Otter; Medina; Karina, 2021). Muitas das aplicações do PLN têm impacto direto em problemas cotidianos, como o uso da arquitetura Bidirectional Encoder Representations from Transformers (BERT) em uma variedade de tarefas de NLP (Devlin et al., 2019), incluindo dúvidas na hora de estudar e tradução de textos.

Uma área das aplicações do PLN é aquela capaz de "interpretar" o conteúdo de documentos para responder às consultas do usuário final. Na literatura, já existem tecnologias capazes de ler, interpretar, criar sentenças e até mesmo gerar perguntas sobre arquivos de texto (exemplos: Otter; Marina; Karina, 2021; Burke et al., 1997; Devlin et al., 2019; Nishida et al., 2018). Assim, surge a pergunta que direciona esta pesquisa: será possível criar um sistema que una e coordene as mais recentes tecnologias para recuperar documentos de um repositório com base em consultas em linguagem natural para um usuário comum?

2. Fundamentação teórica

Neste capítulo, são apresentados conceitos fundamentais para embasar o desenvolvimento do sistema proposto. As métricas de avaliação, como o "EM" (Exact Match) que quantifica a

precisão exata das predições, e a métrica "F1", uma média harmônica entre recall e precisão, são essenciais para analisar o desempenho do modelo.

No âmbito do ranqueamento de textos, abordamos o clássico "TF-IDF" e suas variações, como "BM-25" e "BM-25F", que consideram contextos específicos e partes distintas do texto, refinando a precisão na recuperação de informações. A seção de Processamento de Linguagem Natural (PLN) explora conceitos como "Corpus" (coleção de texto como base de dados) e "OpenQA", que amplia o domínio das perguntas para diversas áreas do conhecimento.

Na abordagem do PLN neural, destacamos a "Representação Vetorial de Palavras", essencial para mapear palavras em vetores numéricos, e a importância das "Redes Neurais" no processamento da linguagem natural, empregando modelos como "BERT". A Compreensão de Perguntas e Respostas é explorada com enfoque na análise sintática e semântica, na classificação de perguntas em categorias específicas e no uso de "Modelos de Linguagem Pré-treinados", como o BERT, treinados em grandes volumes de texto.

A avaliação da qualidade das respostas é vital, com métricas que incluem precisão, completude, relevância, coerência, adequação e fluidez, enquanto desafios como diversidade linguística e ambiguidades são considerados. Este resumo proporciona uma visão panorâmica, preparando o terreno para a implementação do sistema e destacando elementos-chave para a compreensão e busca eficaz de informações.

3. Trabalhos relacionados

Este capítulo destaca diversas arquiteturas de sistemas avaliados na tarefa de responder a perguntas de domínio aberto (OpenQA), considerando critérios como desempenho em datasets específicos, tipo de treinamento, uso de transferência de aprendizagem e reprodutibilidade.

A arquitetura "Retrieve and Read" proposta por Nishida et al. (2018) incorpora a compreensão de leitura à recuperação de informações. Baseado em aprendizado de máquina multitarefa, o modelo compartilha camadas entre tarefas de compreensão de leitura e recuperação de informação.

O BERT introduz uma abordagem de pré-treinamento bidirecional para representações de linguagem. Utilizando um modelo de linguagem mascarado (MLM), supera limitações de modelos unidirecionais. Sua arquitetura, fundamentada nos Transformers de Vaswani et al. (2017), permite maior paralelização no treinamento. Os experimentos revelam desempenho notável, superando concorrentes em datasets como SQuAD v1.1 e v2.0 (Devlin et al., 2019).

Inspirado no BERT, o Open-Retrieval Question Answering (ORQA) de Lee et al. (2019) realiza a tarefa de QA de ponta a ponta, eliminando etapas intermediárias de recuperação. O modelo se destaca ao abordar a tarefa inversa de Cloze, permitindo treinamento eficiente. Embora tenha mostrado superioridade em alguns datasets, observa-se perda em cenários de respostas já conhecidas pelos escritores das perguntas.

O Retrieval-Augmented Language Model Pre-Training (REALM) utiliza o BERT como base e apresenta um modelo que recupera conhecimento de forma latente. Composto por um Recuperador de Conhecimento Neural (NKR) e um Codificador de Conhecimento (KAE), o REALM supera o estado da arte em datasets como NaturalQuestion-Open, WebQuestions e CuratedTrec (Guu et al., 2020).

DrQA (Chen et al., 2017): Uma arquitetura que combina métodos estáticos e de rede neural, composta por recuperador e leitor de documentos. Apesar do desempenho competitivo, sugere melhorias, como treinamento ponta a ponta. RankQA (Kratzwald et al., 2019): Introduzindo um terceiro passo na tarefa de QA, realiza um novo ranqueamento de respostas. Apesar da simplicidade, supera o BERT em diversos datasets, introduzindo características únicas no processo de ranqueamento. Reading Comprehension (Clark C., Gardner M., 2017): Uma arquitetura complexa baseada em pipeline e treinamento em dois datasets distintos (TriviaQA e SQuAD). Apesar do desempenho superior em TriviaQA, coloca-se em oitavo lugar no conjunto SQuAD.

A comparação entre esses trabalhos, destacando critérios como a execução de QA em domínio aberto, o tipo de treinamento, o uso de transfer learning, autoatenção e a reprodutibilidade, orienta a seleção de conceitos para a presente pesquisa. O emprego de BERT, ORQA e elementos de REALM promete uma abordagem robusta e eficaz.

4. Desenvolvimento

A arquitetura de forma geral é apresentada na Figura 1, onde podemos ver que temos 3 entradas, 10 passos e 6 principais componentes. A entrada *dataset* se refere aos conjuntos de dados utilizados para o treinamento, na nossa arquitetura isso pode ser uma lista de *datasets*, mas por simplicidade utilizamos apenas dois, *NaturalQuestions* e *Squad v1.1*. Para cada *dataset* de entrada repetimos um processo como descrito no caso de uso Treinamento da Figura 2. Para isso precisamos de uma implementação que transforma um *dataset* específico para *features* e exemplos, como explicado em mais detalhes na subseção Preparação dos *datasets* na seção Modelo preliminar. Após, enviamos o *dataset* convertido para o processo de ajuste fino (passo 2 da Figura 1) que por sua vez retorna um modelo, esse modelo é salvo (passo 3 da Figura 1) para que futuramente possamos ou ajustá-lo a outro *dataset* ou submetê-lo ao passo 4 da arquitetura.

Seguindo a Figura 1, temos o passo 5, que é receber um conjunto de arquivos do usuário para serem indexados e então recuperados dado uma entrada de busca, passo 6 e 7. Então realizamos um processo de conversão da pergunta e pré-busca com o algoritmo de indexação, passo 8 e 9. Por fim, criamos tuplas com esses arquivos e a pergunta original e usamos isso de entrada para um novo ranqueamento que é realizado pelo modelo ajustado. Vale ressaltar que o que melhoramos de uma iteração para outra são justamente os componentes responsáveis pelos passos descritos acima. A arquitetura como um todo permanece a mesma de iteração para iteração.

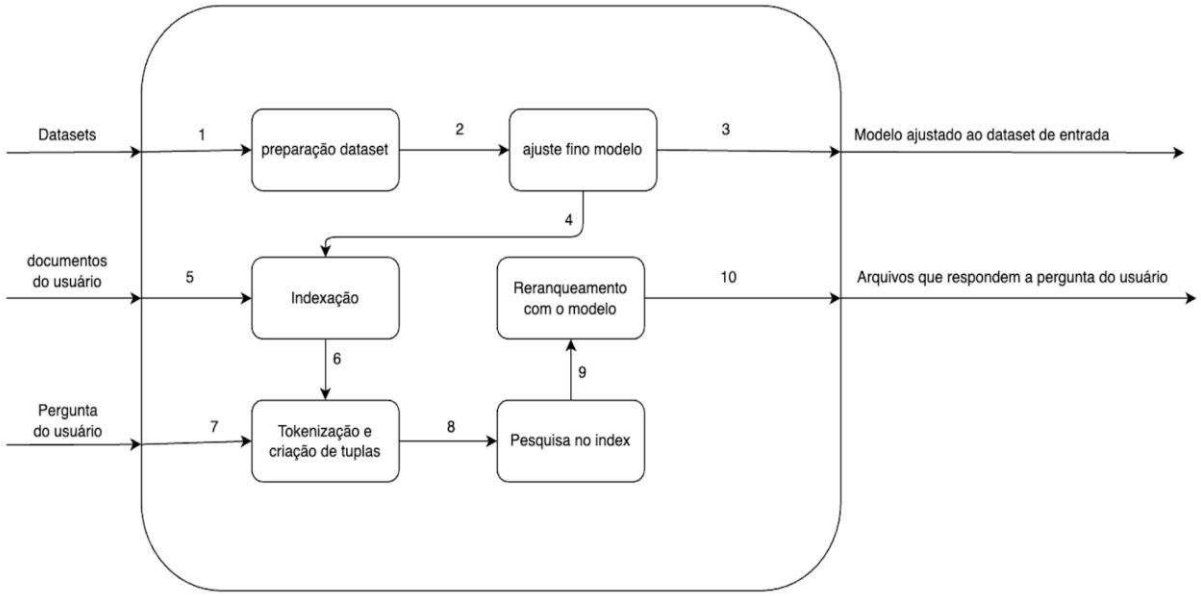


Figura 1 - Arquitetura geral do sistema proposto. Fonte: Elaborada pelo autor (2023).

Na sequência, vem a fase de avaliação dos resultados, onde ideias e técnicas apresentadas por outros autores são combinadas. Por fim, esse processo é repetido incrementando o modelo até que sejamos capazes de não só responder perguntas nos *datasets* escolhidos, mas em conjunto de arquivos que não foram inicialmente feitos para OpenQA. Desta forma, atendendo o objetivo de servir como um repositório inteligente de arquivos para o usuário final.

4.1 Modelo Preliminar

Partimos de um modelo básico utilizando BERT puro, sem modificação, com indexação simples e pré treinado para a tarefa de OpenQA. Então avaliamos o modelo, submetendo-o aos *datasets SQuAD* e *NaturalQuestions* e comparando com os trabalhos relacionados. O *dataset NaturalQuestions* foi especificamente selecionado pelos trabalhos relacionados e pelo presente trabalho, pois suas perguntas foram escritas por seres humanos e sem o viés de saber se a resposta está presente nos documentos ou não.

A entrada do modelo no ajuste fino é um índice, utilizando o algoritmo Okapi BM-25F para ranqueamento como apresentado na seção de Fundamentos Teóricos, e a entrada do modelo após ajuste fino é apenas uma sentença. A saída do modelo é configurável, podendo retornar uma quantidade variável das respostas com maior confiança.

Nessa iteração utilizamos Python e a biblioteca *Datasets*[7] para conseguir trabalhar com os dados, prepará-los e extrair *features* dos mesmos. Então alimentamos essas *features* ao BERT puro, pré-treinado para a tarefa de openQA com o *dataset* SQuAD para realizar o ajuste fino. A entrada para esse modelo depois do ajuste fino é uma sentença. Na próxima etapa declaramos uma aplicação Python simples, que recebe uma pergunta e repassa ao modelo. A

[7] *Datasets* é uma biblioteca do *framework HuggingFace*, disponível em: https://github.com/huggingface/datasets. Acesso em: 18 ago. 2023.

partir daí, o modelo retorna para a aplicação uma lista de 5 respostas e os documentos referentes às respostas. Por fim, apresentamos essa lista ao usuário.

Cada versão de tamanho do BERT vem com dois arquivos: um arquivo representando a meta configuração do BERT (bert_config.json) e outro representando a configuração dos milhões de parâmetros que o BERT tem (bert_ckpt). O processo de ajuste fino justamente altera os parâmetros que vem do bert_ckpt, enquanto o arquivo bert_config contém configurações como quantidade de camadas, o tamanho de cada camada, tamanho do vocabulário, quantidade de ponteiros de atenção. Os diversos modelos de tamanho do BERT referem-se aos dois atributos: quantidade de camadas (*hidden layers* - L) e tamanho do vetor de representação das palavras (*token*) nas camadas internas (*hidden layers*).

Para validar as propostas de modelo aqui está sendo utilizado o BERT_base que tem 12L e 768H, o que gera 110.1 milhões de parâmetros. Entretanto o modelo BERT_large, que é o original, tem uma configuração de 24L 1024H, gerando um total de 340 milhões de parâmetros. Neste trabalho é utilizado o BERT_base pois é extremamente custoso ajustar o BERT_large mesmo que já tenha sido pré-treinado, o BERT_base não é possível ajustar num computador normal pois o tamanho mínimo de cada conjunto de dados enviados para treinamento (batch_size) não cabe numa memória RAM de 16GB. No entanto, como é apresentado conseguimos utilizar a nuvem da própria Google para execução que também tem seus limites, principalmente um limite de tempo de execução.

4.1.1 Ajuste fino

Para realizar as tarefas de ajuste fino foram utilizadas algumas ferramentas, o ajuste fino foi realizado utilizando Python e algumas bibliotecas como o *tensorflow, numpy*, *transformers*, *datasets* e deepspeed. Tensorflow é uma plataforma para codificar, treinar e analisar modelos de aprendizagem de máquinas, importante notar que é código aberto e mantido pela Google.

Numpy é uma biblioteca com diversas funções e tipos de dados numéricos, dando suporte para matrizes grandes e/ou multidimensionais. No nosso contexto usamos para lidar com as matrizes durante a preparação dos dados para dar como entrada ao modelo no tensorflow. *HuggingFace* disponibiliza diversas APIs para Deep Learning, entre elas as utilizadas aqui são a de transformers e *datasets*, para recuperar da internet os modelos bases e os conjuntos de dados, execução do ajuste fino e submissão de perguntas aos conjuntos de arquivos testes. A biblioteca *DeepSpeed*[8] foi o que nos permitiu reduzir o tempo de ajuste fino do Squad de 33 horas para 6 horas, e o tempo de ajuste fino do NaturalQuestions de 96 horas para 11horas. Isso é extremamente importante, dado que a nuvem da Google não permite execuções com tempo maior de 12 horas na versão grátis e mais de 24 horas na versão paga.

No final do processo de ajuste fino preliminar teremos dois modelos mudando o *dataset* em que foram ajustados. Agora para a parte final da tarefa, exportamos este modelo e carregamos ele numa aplicação simples que fica em laço esperando uma entrada do usuário. O usuário, por motivos de simplicidade, pode fazer duas operações. Enviar um arquivo para o seu repositório de dados ou enviar uma pergunta em texto simples.

Para facilitar a criação de indexação foi utilizado uma biblioteca em Python chamada *Whoosh*[9], que facilita a criação de um motor de busca. Usamos a funcionalidade de criar índice

[8] DeepSpeed Microsoft: https://github.com/microsoft/DeepSpeed
[9] Whoosh: https://github.com/mchaput/whoosh

que por padrão utiliza o algoritmo Okapi BM25F. Então guardamos esse índice localmente para aguardar uma eventual busca do usuário. No caso de o usuário enviar uma pergunta, primeiro avaliamos se há um índice previamente criado. Então, primeiro utilizamos um utilitário da biblioteca *ktrain* para tokenizar a pergunta, então criamos uma *query* para o *index* e retornamos diversos documentos que possam ter a resposta, previamente ranqueados com o algoritmo padrão do Whoosh. Ranqueamos os documentos primeiro dividindo os documentos em parágrafos, chamando tais parágrafos de contexto e guardamos um ID para saber recuperar o arquivo original.

Assim, usando esses contextos e a pergunta, construímos uma tupla de entrada para o modelo. Modelo retorna uma lista de possíveis respostas, sendo que cada resposta tem um valor de confiança e uma referência para o contexto original. Com esta lista em mãos reordenamos baseado na confiança, recuperamos o contexto original e dele recuperamos o arquivo original que possivelmente contém a resposta. Por fim, apresentamos isso para o usuário.

O primeiro modelo resultante desta versão preliminar é o modelo BERT-base ajustado ao *dataset Squad*. O resultado obtido, dado a configuração do modelo, é comparativo com o resultado original no Squad v1.1 que foi 85.1 de EM e 91.8 de F1 usando a configuração BERT_large(Devlin et al., 2018). Agora, o mesmo modelo sem ter sido ajustado para o NaturalQuestion, obteve 42% de EM e 56% de F1, o que é um desempenho relativamente inferior ao desempenho mais alto até a presente data de acordo com a tabela de melhores modelos curada pela própria Google, criadora do *dataset* Natural Question. Entretanto vale ressaltar que este resultado é no estilo *ZeroShot*, o modelo nunca havia recebido nenhum exemplo do *dataset* NaturalQuestion antes.

O segundo modelo é análogo ao anterior, com a diferença que o ajuste fino foi realizado ao *dataset NaturalQuestions* e então submetido à validação nos dois *datasets*. No *NaturalQuestions*, o resultado obtido foi de 59% EM e 73%F1. Ficando relativamente próximo dos resultados obtidos pelos modelos com maior desempenho. Até a presente data (23/10/2023), o melhor modelo é *PoolingFormer*, com um desempenho de 79.8% F1. O que de uma certa forma era esperado, já que nos melhores modelos, muitos utilizam o BERT de base, mas em configurações superiores (BERT_large).

Agora, no *dataset Squad*, o desempenho foi análogo ao ZeroShot do primeiro modelo com 42% de EM e 62% de F1. Este desempenho deve ser melhorado quando ajustarmos o modelo no *dataset* também para que não seja mais uma validação ZeroShot. De qualquer forma, esse desempenho em ZeroShot é promissor para o objetivo do modelo, que é lidar com textos nunca antes vistos pelo modelo.

Na próxima iteração são exploradas diferentes formas de criar um índice. Outra possibilidade a ser explorada é a evolução da base de documentos utilizando os documentos fornecidos pelo usuário, atualizando o índice e aplicando novamente o ajuste fino ao novo índice. Além disso, faremos o ajuste fino como já mencionado em 2 *datasets*, exaurindo as diferentes combinações de ordem, dado que a ordem dos *datasets* no ajuste fino importa, como é demonstrado nos resultados do modelo final.

Outra conclusão é que apenas o BERT puro com ajuste fino em algum *dataset* de *Question Answering* já nos permite um desempenho considerável, com pouco custo computacional, se comparado ao processo de pré-treinamento, pois precisamos fazer apenas o ajuste fino. Por fim, demonstramos que é possível utilizar mais de um *dataset*, basta incluí-lo

no processo de tokenização para converter pro padrão esperado, assim poderemos construir um sistema que tenha como treinamento base vários *datasets*.

4.2 Modelo Final

Nesta segunda iteração a principal alteração é o processo de ajuste fino. O processo de ajuste fino é alterado para ser aplicado em todos os *datasets* e não só em um único. O objetivo é conseguir desenvolver as qualidades que cada *dataset* requer para um bom desempenho. Como temos 2 *datasets* escolhidos e a ordem em que o ajuste fino é feito em cada *dataset* importa, teremos 2 modelos treinados e ajustados prontos para serem avaliados na tarefa deste trabalho, além dos 2 modelos base, totalizando 4 modelos produzidos neste trabalho. A dificuldade foi reaplicar o ajuste fino aos *datasets* pois cada um tem sua peculiaridade.

Os parâmetros de configuração do modelo permanecem o mesmo para as versões finais. O motivo disso é comparar se aplicar dois ajustes com *datasets* diferentes permite uma melhoria no desempenho dos modelos resultantes. Então se compararmos modelos com parâmetros diferentes e ajuste fino diferentes, não conseguiremos isolar o motivo de melhora ou piora no desempenho.

A tokenização dos *datasets* é igual ao primeiro modelo de cada *dataset*, pois já foi feito de tal forma que o resultado pós tokenização dos dois *datasets* seja igual. Isso não foi uma decisão tomada assim como na configuração para equalizar os modelos. O ferramental utilizado para realizar o ajuste fino obriga os dados a serem estruturados da mesma maneira antes de serem utilizados. Mais detalhes sobre o formato esperado estão descritos na seção preparação de *datasets* do modelo preliminar. Entretanto, esse requisito da ferramenta permite que consigamos isolar ainda mais o impacto de um duplo ajuste fino, dado que até o formato dos dados são iguais.

4.2.1 Ajuste fino conjunto

O desafio foi conseguir realizar o ajuste fino nos dois *datasets*. Aqui temos duas formas de fazer tal tarefa, uma é carregar os dois conjuntos de dados para alguma abstração em comum ou de certa forma salvar os parâmetros resultantes do ajuste fino em um *dataset* para então usar isso como ponto de partida para um segundo ajuste fino. A segunda estratégia foi escolhida por alguns motivos: recursos limitados (um conjunto de dados unificados demandaria muito mais processamento); complexidade de unir os 2 *datasets* para um mesmo tipo de objeto; complexidade de separar as predições unificadas para um *dataset* específico.

Como explicado na seção de preparação dos *datasets*, o processo de tokenização resolveria os dois últimos problemas mencionados aqui, mas ainda resta o problema de recursos limitados. Simulações para descobrir o tempo estimado de execução dos modelos foram executadas apenas para saber os limites de configuração do modelo e tamanho do dado. A ferramenta utilizada para conseguir rodar esses ajustes finos e simulações foi o Google Colab que tem acesso a máquinas virtuais com GPUs poderosas, memórias RAMs enormes e memórias de disco maiores ainda. Ainda assim, não foi o suficiente para podermos utilizar a primeira estratégia, pois o Google Colab tem uma limitação de tempo de uso, sendo 12 horas para a versão grátis/estudante e 24 horas para a versão paga.

Por exemplo, o ajuste fino no *dataset Squad* utilizando a ferramenta *DeepSpeed* para acelerar o processamento e usar os recursos disponíveis (GPU 16 GB, RAM 32 GB, 240 GB SSD) ao máximo ainda teve uma duração total de 8 horas, enquanto o ajuste fino no *dataset*

NaturalQuestion teve uma duração total de 14 horas. Vale ressaltar que sem o uso da ferramenta *DeepSpeed* esses tempos foram ridiculamente maiores, sendo o pior o *NaturalQuestions* com um tempo estimado de execução de 108 horas. Se fossemos realizar o ajuste dos dois *datasets* em uma única execução ultrapassaríamos o tempo máximo ou os recursos disponíveis. Dessa forma, tivemos que nos preocupar em salvar o ajuste e depois partir o segundo ajuste do primeiro.

Primeiro realizamos o processo de ajuste fino no *dataset Squad* como descrito na seção do modelo preliminar. Submetemos esse modelo à avaliação do SQUAD para manter um histórico, bem como ao *Natural Questions*, sendo que esses resultados são os dos modelos preliminares. O próximo passo foi salvar o modelo intermediário antes de ter o segundo ajuste realizado, para isso utilizamos novamente uma API do *HuggingFace* que permite salvarmos o modelo num repositório remoto gratuito, sendo obrigatória a autenticação no repositório que por sua vez requer uma conta criada no site da API.

Agora para realizar o segundo ajuste fino, a principal alteração é mudar o modelo de partida no script, ao invés de partir de um modelo cru do BERT, é realizado o login no repositório de modelos e alterado o modelo usado para o modelo salvo remotamente no primeiro ajuste fino. Então realizamos o ajuste fino do mesmo modelo no *Natural Questions dataset*. Salvamos esse modelo também no repositório de modelos, alteramos o script de avaliação para realizar o login no repositório e buscar o nosso modelo. Por fim, submetemos novamente a avaliação dos dois *datasets* e guardamos essa informação para análise. Então realizamos o mesmo processo invertendo o *dataset* inicial de ajuste fino, agora sendo *Natural Questions* o primeiro e SQUAD o segundo.

4.2.2 Avaliação com os *datasets*

Dado o processo de ajuste fino descrito, temos dois modelos resultantes, além dos dois modelos preliminares. O primeiro é o modelo ajustado ao *dataset* Squad e então ao *dataset Natural Questions*. O segundo é o modelo ajustado ao *dataset Natural Questions* e então ao *dataset Squad*. Os modelos resultantes são submetidos ao mesmo processo de avaliação que os modelos preliminares. Os resultados do primeiro modelo e segundo modelo estão na tabela abaixo, para facilitar a comparação os modelos preliminares também estão na Tabela 3.

	Squad		***Natural Questions***	
Modelo/Métricas	F1 (%)	EM (%)	F1 (%)	EM (%)
finetuned_squad	78.36	60.96	56.21	42.03
finetuned_nq	62.88	42.35	73.05	59.57
Squad_NaturalQuestion	70.50	50.18	75.93	62.43
NaturalQuestion_Squad	77.87	60.25	74.42	60.80
Resultados dos trabalhos relacionados				
Retrieve and Read	39.8	32.7	X	X
BERT	92.8	86.7	X	X
ORQA	X	33.2	X	33.3

REALM	X	X	X	40.4
Reading Comprehension	67.34	59.14	X	X

Tabela 5 - Resultados dos modelos e resultados de trabalhos relacionados. Fonte: Elaborada pelo autor (2023).

Como vimos ao longo deste texto, o *dataset Squad* costuma exigir a capacidade de representação de dados maior, ou seja, quanto mais parâmetros o modelo, maior o desempenho do modelo no *dataset*. Ou seja, um modelo que seja bom de decorar os dados durante o treinamento irá apresentar desempenho melhor, enquanto o *Natural Question* exige a capacidade de compreender o contexto para encontrar a resposta, já que suas perguntas nem sempre têm resposta no texto e os modelos têm que saber identificar a ausência de resposta também.

Podemos observar que os modelos ajustados ao *dataset Squad* em algum momento do processo tiveram melhor desempenho no *dataset Natural Question*. Entende-se que a capacidade exigida pelo *dataset Squad* faz com que o modelo consiga capturar uma representação melhor em seus milhões de parâmetros. Porém, tais parâmetros não necessariamente resultam em um modelo mais generalista ou se resultam, não são observados quando submetidos ao *Squad*. Agora, quando os modelos com ajuste finos em dois *datasets* são submetidos ao *Natural Question*, o desempenho foi consideravelmente melhor. Podemos concluir que a generalização do modelo beneficiou o resultado em tal *dataset* que justamente requer uma capacidade de forma limitada, compreender o contexto.

Se compararmos os modelos deste trabalho com os trabalhos relacionados, vemos que tivemos desempenho superior a todos eles menos o BERT original que foi usado na sua configuração mais poderosa (LARGE), enquanto aqui usamos a configuração intermediária (BASE). É argumentável que a diferença de resultados entre nosso modelo e os modelos de trabalhos relacionados vem do fato que os trabalhos relacionados foram feitos a alguns anos e que na época não havia ferramentais elaborados como os que foram utilizados aqui nesse trabalho.

Os resultados alcançados são relevantes e melhores do que o esperado pois conseguimos demonstrar que o ajuste fino em diferentes *datasets* pode sim melhorar o desempenho, dependendo da natureza de tais *datasets*. Se compararmos com nossa baseline que é o BERT, ainda tivemos um desempenho muito inferior. Mas isso se deve ao tamanho do modelo usado, que neste caso foi um intermediário (BASE).

5. Conclusão

Este trabalho teve como objetivo propor um repositório de arquivos que consiga realizar buscas em linguagem natural. Para isso, o estado da arte e trabalhos relacionados foram analisados. Uma arquitetura foi proposta envolvendo o modelo BERT e o ranqueamento de arquivos Okapi BM25F. Os critérios de validação escolhidos foram as métricas F1 e EM nos datasets Natural Question e Squad v1.1. Desta forma, entende-se que todos os objetivos propostos foram cumpridos. Entretanto, a arquitetura proposta tem seus pontos fracos e pontos fortes.

Alguns pontos fortes foram identificados após a análise dos resultados, sendo que todos eles derivam da generalidade do sistema, que não se amarra a nenhum dataset específico e nenhum modelo específico. O código feito neste trabalho permite que qualquer modelo baseado

em BERT publicado no repositório de modelos HuggingFace seja utilizado como base. Além disso, permite que qualquer dataset seja utilizado, contanto que o desenvolvedor forneça o processo de tokenização para transformar o dataset no padrão esperado. Permite também que diversas iterações de ajuste fino sejam realizadas, no mesmo dataset ou diferentes datasets, em diferentes ordens. Por isso, a maior contribuição do trabalho é a arquitetura proposta, pois ela pode ser utilizada para criar modelos generalistas.

O principal ponto fraco do trabalho é a capacidade de treinamento reduzida pela falta de acesso a recursos computacionais como os utilizados pelos autores do BERT para treinar a configuração LARGE. Conseguimos utilizar apenas a configuração BASE, que já foi suficiente para comprovar a hipótese de ajuste fino com 2 ou mais datasets pode melhorar o desempenho do modelo nas métricas escolhidas. Outro ponto fraco do sistema, como já descrito na seção de resultados, é o algoritmo Okapi BM25F de ranqueamento probabilístico usado para pré-selecionar alguns arquivos antes de ranqueá-los com o modelo. O principal problema do algoritmo é que não consegue encontrar arquivos que utilizem sinônimos corretamente, apenas as palavras contidas na busca.

Essa arquitetura pode ser amplamente utilizada quando o objetivo for generalismo, entretanto se o desafio é lidar com algum tipo de conhecimento específico, outra arquitetura deve ser considerada. Como vimos nos resultados, o modelo obteve resultados consideráveis com o dataset generalista NaturalQuestion. Agora os resultados no dataset Squad v1.1, que requer uma maior especialização maior, tiveram uma piora conforme novos ajustes finos foram realizados. Portanto, se o objetivo for lidar com conjuntos de dados genéricos, esta arquitetura se mostra viável.

6. Trabalhos Futuros

O trabalho futuro mais óbvio para melhorar o desempenho é conseguir realizar o ajuste fino utilizando modelos com configuração do BERT_large. Para a fase de ajuste fino, seria interessante trabalhar com mais conjuntos de dados que exigem habilidades diferentes do modelo. Outra possível parte da arquitetura a ser revisada é o primeiro ranqueamento que atualmente foi feito com o algoritmo Okapi BM25-F, entretanto todos os algoritmos da família TF-IDF têm o problema de conseguirem considerar só arquivos que contenham textos estatisticamente semelhante com a consulta. Para resolver este problema, podemos considerar o Elastic Search que vem sendo amplamente utilizado para toda a parte de indexação e recuperação de informação do índice.

7. Referências

BEAULIEU, M. M. *et al*. Okapi at TREC-5. *In*: VOORHEES, E. M.; HARMAN, D. K. (ed.). THE FIFTH TEXT RETRIEVAL CONFERENCE (TREC-5), 5th, 1997, Gaithersburg. **Proceedings** [...]. Gaithersburg: National Institute of Standards and Technology, 1997. p. 143–165. (NIST Special Publication 500-238). Disponível em: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication500-238.pdf. Acesso em: 12 nov. 2023.

BURKE, R. *et al*. Question Answering from Frequently Asked Question Files: Experiences with the FAQ FINDER System. **AI Magazine**, Whashington, v. 18, n. 2, p. 57-66, 1997. Disponível em:

https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/1294/1195. Acesso em: 12 nov. 2023.

CHEN, D. *et al*. Reading Wikipedia to answer open-domain questions. *In*: ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS (VOLUME 1: LONG PAPERS), 55th, 2017, Vancouver. **Proceedings** […]. Vancouver: Association for Computational Linguistics, 2017. p. 1870–1879.

CLARK, C.; GARDNER, M. Simple and effective multiparagraph reading comprehension. *In*: ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS (VOLUME 1: LONG PAPERS), 56th, 2018, Melbourne. **Proceedings** […]. Melbourne: Association for Computational Linguistics, 2018. p. 845–855.

DEVLIN, J. BERT. [2023]. Disponível em: https://github.com/google-research/bert. Acesso em: 12 nov. 2023.

DEVLIN, J. *et al*. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *In*: ANNUAL CONFERENCE OF THE NORTH AMERICAN CHAPTER OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS: HUMAN LANGUAGE TECHNOLOGIES (NAACL-HLT 2019), 17th, 2 a 7 jun. 2019, Minneapolis. **Proceedings** [...]. Minneapolis: NAACL-HLT, 2 a 7 jun. 2019. p. 4171-4186. Disponível em: https://arxiv.org/pdf/1810.04805.pdf. Acesso em: 12 nov. 2023.

GUU, K. *et al*. REALM: Retrieval-Augmented Language Model Pre-Training. **Arxiv**, New York, p. 1-12, 2020. Disponível em: https://arxiv.org/pdf/2002.08909.pdf. Acesso em: 12 nov. 2023.

KRATZWALD, B.; EIGENMANN, A.; FEUERRIEGEL, S. RankQA: Neural question answering with answer re-ranking. *In*: ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, 57th, 2019, Florence. **Proceedings** […]. Florence: Association for Computational Linguistics, 2019. p. 6076–6085.

MIKOLOV, T. *et al*. Efficient Estimation of Word Representations in Vector Space. **Arxiv**, New York, p. 1-12, 2013. Disponível em: https://arxiv.org/abs/1301.3781. Acesso em: 12 nov. 2023.

NISHIDA, K. *et al*. Retrieve-and-read: Multi-task learning of information retrieval and reading comprehension. *In*: ACM INTERNATIONAL CONFERENCE ON INFORMATION AND KNOWLEDGE MANAGEMENT, 27th, 2018, New York. **Proceedings** […]. Torino: ACM, 2018. p. 647–656.

OTTER, D. W.; MEDINA, J. R.; KALITA, J. K. A Survey of the Usages of Deep Learning for Natural Language Processing. **IEEE Transactions on Neural Networks and Learning Systems**, [S.l.], v. 32, n. 2, p. 604-624, fev. 2021. Disponível em: https://doi.org/10.1109/TNNLS.2020.2979670. Acesso em: 12 set. 2022.

RAJPURKAR, P. *et al*. SQuAD: 100,000+ Questions for Machine Comprehension of Text. *In*: SU, J.; DUH, K.; CARRERAS, X. (ed.). CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING, 2016, Austin. **Proceedings** [...]. Austin:

Association for Computational Linguistics, 2016. p. 2383-2392. Disponível em: https://aclanthology.org/D16-1264/. Acesso em: 12 nov. 2023.

TURC, I. *et al*. Well-Read Students Learn Better: On The Importance Of Pre-Training Compact Models. *In*: INTERNATIONAL CONFERENCE ON LEARNING REPRESENTATIONS (ICLR 2020), 8th, 2020, Addis Ababa. **Proceedings** [...]. Addis Ababa: [S. n.], 2020. Disponível em: https://openreview.net/forum?id=BJg7x1HFvB. Acesso em: 12 nov. 2023.

VASWANI, A. *et al*. Attention is all you need. *In*: GUION, I. *et al*. (ed.). CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS (NIPS 2017), 31st, 2017, Long Beach. **Proceedings** [...]. Long Beach: [S. n.], 2017. p. 6000-6010. Disponível em: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf. Acesso em: 12 nov. 2023.

ZHU, F. *et al*. Retrieving and Reading: A Comprehensive Survey on Open-domain Question Answering. **Arxiv**, New York, 2021. Disponível em: https://arxiv.org/pdf/2101.00774.pdf. Acesso em: 12 nov. 2023.