UNIVERSIDADE FEDERAL RURAL DO RIO DE JANEIRO
INSTITUTO MULTIDISCIPLINAR

EDUARDO DE ALMEIDA FERRO CABRAL
ROMULO MORAIS MENEZES

SACI PeReRe: um sistema para responder FAQs

Prof. Leandro Guimarães Marques Alvim, D.Sc.
Orientador

Nova Iguaçu, Agosto de 2023

SACI PeReRe: um sistema para responder FAQs

Eduardo de Almeida Ferro Cabral

Romulo Morais Menezes

Projeto Final de Curso submetido ao Departamento de Ciência da Computação do Instituto Multidisciplinar da Universidade Federal Rural do Rio de Janeiro como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Apresentado por:

_____

Eduardo de Almeida Ferro Cabral

_____

Romulo Morais Menezes

Aprovado por:

_____

Prof. Leandro Guimarães Marques Alvim, D.Sc.

_____

Prof. Marcel William Rocha da Silva, D.Sc.

_____

Prof. Filipe Braida do Carmo, D.Sc.

NOVA IGUAÇU, RJ - BRASIL

Agosto de 2023

Emitido em 07/08/2023

DOCUMENTOS COMPROBATÓRIOS Nº 13691/2023 - CoordCGCC (12.28.01.00.00.98)

(Nº do Protocolo: NÃO PROTOCOLADO)

(Assinado digitalmente em 07/08/2023 13:30)
FILIPE BRAIDA DO CARMO
PROFESSOR DO MAGISTERIO SUPERIOR
DeptCC/IM (12.28.01.00.00.83)
Matrícula: ###295#4

(Assinado digitalmente em 07/08/2023 17:23)
LEANDRO GUIMARAES MARQUES ALVIM
PROFESSOR DO MAGISTERIO SUPERIOR
DeptCC/IM (12.28.01.00.00.83)
Matrícula: ###008#2

(Assinado digitalmente em 07/08/2023 15:35)
MARCEL WILLIAM ROCHA DA SILVA
PROFESSOR DO MAGISTERIO SUPERIOR
DeptCC/IM (12.28.01.00.00.83)
Matrícula: ###807#6

(Assinado digitalmente em 07/08/2023 12:52)
ROMULO MORAIS MENEZES
DISCENTE
Matrícula: 2019######6

(Assinado digitalmente em 07/08/2023 12:54)
EDUARDO DE ALMEIDA FERRO CABRAL
DISCENTE
Matrícula: 2019######3

Visualize o documento original em https://sipac.ufrrj.br/documentos/ informando seu número: **13691**, ano: **2023**, tipo: **DOCUMENTOS COMPROBATÓRIOS**, data de emissão: **07/08/2023** e o código de verificação: **dc0bd2adb5**

Agradecimentos

Eduardo de Almeida Ferro Cabral

Em primeiro lugar, expresso minha profunda gratidão aos meus pais, Elisa e Amaro, cujo apoio e presença constante foram inestimáveis ao longo desta jornada. Cada investimento feito, ensinamento compartilhado e incentivo foram pilares essenciais para o desenvolvimento da minha formação acadêmica e pessoal. Sou profundamente grato por cada sacrifício e dedicação, pois sei que sem vocês, nada disso seria possível. Serei eternamente grato por tudo o que fizeram por mim.

Gostaria de expressar meu mais sincero agradecimento ao Romulo Menezes, meu parceiro nesse trabalho e amigo, por todo apoio ao longo da jornada acadêmica. Sua dedicação e companheirismo foram fundamentais para o sucesso deste trabalho, e estou imensamente grato por sua amizade ao longo desses anos.

Gostaria de expressar meu profundo agradecimento a todos os membros do grupo Softawii, cuja união esteve presente tanto nos momentos difíceis quanto nos momentos de alegria ao longo dessa jornada. Cada gameplay e grupo de estudos que compartilhamos juntos foram inestimáveis, e não encontro palavras suficientes para expressar o quanto sou grato por essas experiências.

Gostaria de expressar meus sinceros agradecimentos a todos os professores do Departamento de Ciência da Computação, que, ao longo desta jornada acadêmica, foram fundamentais em minha formação e no desenvolvimento deste trabalho. Seus conhecimentos compartilhados foram inestimáveis para o meu crescimento como estudante e futuro profissional.

Meus sinceros agradecimentos a todos.

Romulo Morais Menezes

Antes de tudo, gostaria de expressar minha profunda gratidão aos meus pais, cuja crença e dedicação foram fundamentais para a minha jornada educacional. Foi por meio do constante apoio deles e dos investimentos que fizeram em minha educação que tive a oportunidade de adquirir uma formação sólida, o que me permitiu alcançar esta graduação. Sou imensamente grato por tudo o que fizeram por mim.

Gostaria de agradecer também ao meu amigo Eduardo Cabral por concordar em realizar este trabalho final comigo. Reconheço que houve momentos difíceis e desafiadores, e gostaria de me desculpar por qualquer ação ou omissão da minha parte. Sou imensamente grato pela sua amizade e pela oportunidade de colaborarmos neste projeto.

Gostaria de expressar meu profundo agradecimento a todos os meus amigos e colegas de faculdade que estiveram ao meu lado durante todos esses anos, em particular ao pessoal do Softawii. Agradeço a cada um de vocês por esclarecerem minhas dúvidas, por organizarem grupos de estudo e momentos de diversão para aliviar um pouco o peso da faculdade. A presença e apoio de vocês foram inestimáveis e contribuíram significativamente para minha jornada acadêmica.

Por último, mas definitivamente não menos importante, gostaria de expressar minha gratidão a cada um dos professores do Departamento de Ciência da Computação que contribuíram significativamente para a minha formação. Além das valiosas aulas ministradas, eles compartilharam suas experiências e conselhos, enriquecendo ainda mais meu conhecimento. Também gostaria de agradecer especialmente ao meu orientador, Leandro Guimarães Marques Alvim, por sua disposição em nos guiar ao longo dessa jornada acadêmica.

RESUMO

SACI PeReRe: um sistema para responder FAQs

Eduardo de Almeida Ferro Cabral e Romulo Morais Menezes

Agosto/2023

Orientador: Leandro Guimarães Marques Alvim, D.Sc.

Nos últimos anos, o crescimento exponencial da internet revolucionou a forma como as pessoas interagem, resultando em um aumento significativo no número de usuários. A disseminação rápida de informações e o surgimento de novas formas de comunicação são características marcantes dessa transformação tecnológica. Nesse cenário, os chatbots desempenham um papel relevante ao fornecer respostas rápidas. Ao considerarmos o contexto educacional, especialmente em universidades públicas com grande ingresso de alunos, o desenvolvimento de um sistema de chatbot personalizado pode aliviar a carga de trabalho dessas instituições ao oferecer suporte constante aos estudantes e fornecer informações sobre cursos, processos seletivos, calendário acadêmico e outros tópicos relevantes. Este trabalho propõe desenvolver um sistema de gerenciamento de perguntas e respostas para que possam ser consumidos por um modelo de inteligência artificial que ficará responsável por responder perguntas frequentes. Dessa forma, os profissionais teriam mais tempo para questões complexas, enquanto as dúvidas comuns seriam prontamente atendidas pelo chatbot, contribuindo para aprimorar a experiência acadêmica como um todo.

ABSTRACT

SACI PeReRe: um sistema para responder FAQs

Eduardo de Almeida Ferro Cabral and Romulo Morais Menezes

Agosto/2023

Advisor: Leandro Guimarães Marques Alvim, D.Sc.

In recent years, the exponential growth of the internet has revolutionized the way people interact, resulting in a significant increase in the number of users. The rapid dissemination of information and the emergence of new forms of communication are striking features of this technological transformation. In this scenario, chatbots play a relevant role by providing quick responses. When considering the educational context, especially in public universities with a large influx of students, the development of a personalized chatbot system can alleviate the workload of these institutions by offering constant support to students and providing information about courses, admission processes, academic calendars and other relevant topics. This work aims to develop a question and answer management system that can be consumed by an artificial intelligence model responsible for answering frequently asked questions. In this way, professionals would have more time for complex issues, while common doubts would be promptly addressed by the chatbot, contributing to enhancing the overall academic experience.

Lista de Figuras

Lista de Tabelas

Lista de Códigos

Lista de Abreviaturas e Siglas

API	Application Programming Interface
MLP	Multilayer Perceptron
PLN	Processamento de Linguagem Natural
NLU	Natural Language Understanding
NLG	Natural Language Generation
BoW	Bag-of-Words
TF-IDF	Term Frequency-Inverse Document Frequency
CBOW	Continuous Bag-of-Words
WWW	World Wide Web
P2P	Peer-to-Peer
FTP	File Transfer Protocol
IPTV	Internet Protocol Television
HTTP	Hypertext Transfer Protocol
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
STS	Similaridade Textual Semântica
FAQ	Frequently Asked Questions
LSI	Latent Semantic Indexing
IA	Inteligência Artificial

SPA	Single-Page Application
SGBD	Sistema de Gerenciamento de Banco de Dados
JSON	JavaScript Object Notation
CI/CD	Continuous Integration/Continuous Delivery
JDA	Java Discord API
QA	Question Answering
CRUD	Create, Read, Update and Delete

Sumário

Capítulo 1

Introdução

A Internet tem experimentado um crescimento significativo nos últimos anos, com um aumento no número de usuários em todo o mundo. A rápida disseminação de informações e a transformação dos hábitos das pessoas são algumas das mudanças trazidas por essa revolução tecnológica. Nesse contexto, os chatbots, como o ChatGPT, estão desempenhando um papel importante.

Com o acesso fácil à Internet, as pessoas têm a capacidade de aprender sobre qualquer assunto de forma rápida e conveniente. Isso tem levado a uma necessidade de obter respostas imediatas e em tempo real. Surge então a possibilidade de utilizar chatbots, no contexto educacional, em universidades públicas.

O número grande de estudantes ingressando nas instituições de ensino superior acaba tornando difícil para os profissionais responderem a todas as perguntas de forma ágil e eficiente. É nesse ponto que um chatbot personalizado pode desempenhar um papel importante. Um chatbot desenvolvido especificamente para auxiliar estudantes universitários pode fornecer respostas rápidas e precisas sobre diversos tópicos relacionados à universidade, aliviando a demanda sobre os profissionais.

1.1 Objetivo

O objetivo deste trabalho é então desenvolver um sistema de chatbot completo, abrangendo desde a criação de um site para cadastrar perguntas até a implementação de uma API que permita a integração com aplicativos externos, como o Discord, WhatsApp e entre outros. Para atingir esse objetivo, serão definidos os seguintes objetivos específicos:

- Projetar e desenvolver uma interface Web intuitiva e amigável para permitir o cadastro de perguntas e respostas no sistema de chatbot.
- Desenvolver um sistema que utilize um modelo de redes neurais para identificar qual é a resposta de uma pergunta.
- Projetar e implementar uma API RESTful que permita a integração do sistema de chatbot com aplicativos externos.
- Utilizando a API, criar uma aplicação no Discord para os usuários finais.

1.2 Resumo dos Resultados

Foi criado um sistema administrativo essencial para o controle de perguntas e respostas, permitindo a inclusão de novos usuários e oferecendo uma visão completa do histórico e estatísticas das respostas do modelo com base no *feedback* dos usuários. O sistema também identifica as perguntas que o modelo não conseguiu responder ou não encontrou na base de dados. Na página de estatísticas, informações relevantes, como o total de perguntas feitas, *feedbacks* dos usuários e sua distribuição em positivos e negativos, são apresentadas, juntamente com os *feedbacks* textuais dos usuários. Um gráfico claro e intuitivo mostra a quantidade de *feedbacks* positivos e negativos, oferecendo uma visão imediata da satisfação geral dos usuários em relação ao conteúdo do *bot* e suas respostas esperadas.

Foi desenvolvido um *bot* para Discord com o propósito de integrar-se à base de dados e ao modelo. Esse *bot* possibilita que os usuários façam perguntas e recebam

respostas automáticas. Após receber a resposta, o usuário tem à sua disposição três botões de *feedback* para avaliar a adequação da resposta. Esses botões permitem uma avaliação rápida e fácil, permitindo ao usuário indicar se a resposta foi útil ou não em relação à pergunta original, além de fornecer um *feedback* textual. Essas avaliações são de extrema importância para aprimorar o desempenho e a precisão do *bot*, capacitando-o a oferecer respostas mais relevantes e satisfatórias.

1.3 Principais Contribuições

- *Uso de inteligência artificial para responder FAQs em português brasileiro*
  Este trabalho contribuiu como material de estudo sobre a utilização de inteligência artificial (IA) na resposta a perguntas frequentes em português brasileiro. Mais especificamente, foi explorado o uso de similaridade semântica para comparar perguntas feitas em linguagem natural com as perguntas cadastradas em uma base de dados, com o objetivo de identificar a pergunta mais semelhante e, consequentemente, fornecer a resposta correspondente de forma precisa e eficiente.

- *Desenvolvimento de um sistema administrativo Web*
  Uma outra contribuição importante deste trabalho está relacionada ao desenvolvimento de um sistema administrativo Web, que serve como referência para projetos futuros. Neste estudo, são apresentadas soluções que desempenham bem, enquanto outras podem servir como aprendizados para aprimoramentos futuros.

1.4 Organização do Trabalho

Os capítulos seguintes deste trabalho estão divididos da seguinte forma:

- O capítulo 2 consiste em dar o embasamento teórico do trabalho, para facilitar o entendimento das técnicas utilizadas.

- No capítulo 3 é apresentado a motivação, trabalhos relacionados e a proposta

deste trabalho.

- No capítulo 4 abordam-se as tecnologias utilizadas para o desenvolvimento, a arquitetura de todo o sistema e os resultados obtidos.
- Por fim, no capítulo 5 é apresentado as conclusões obtidas ao longo do desenvolvimento deste trabalho

Capítulo 2

Fundamentação teórica

Neste capítulo, será apresentada a fundamentação teórica dos principais conceitos utilizados no desenvolvimento deste trabalho, fornecendo uma base sólida para compreender seu funcionamento. Serão abordados conceitos fundamentais sobre aprendizado de máquina, redes neurais artificiais e processamento de linguagem natural, que são essenciais para entender o modelo utilizado na identificação de respostas para as perguntas formuladas. Além disso, serão apresentados conceitos relacionados à Internet, com o intuito de compreender o funcionamento do sistema Web de cadastramento de perguntas, bem como a API.

2.1 Aprendizado de Máquina

O aprendizado de máquina é uma área da inteligência artificial que de acordo com a definição de Mitchell (1997), um programa de computador é considerado capaz de aprender com a prática se a sua habilidade em realizar determinadas tarefas, que fazem parte de uma classe de tarefas, melhora conforme adquire experiência. Essa melhora é medida através de uma métrica de desempenho específica para a tarefa em questão. Em outras palavras, o aprendizado de máquina é o processo de ensinar ao programa como executar uma tarefa com maior eficiência, utilizando os dados obtidos da experiência anterior.

Existem três razões principais pelas quais seria desejável ter um agente que aprende em vez de apenas programar uma melhoria no agente desde o início. Primeiro, os projetistas não podem antecipar todas as situações possíveis que o agente possa enfrentar. Em segundo lugar, os projetistas não podem antecipar todas as mudanças ao longo do tempo. Terceiro, às vezes os programadores humanos não sabem como programar uma solução por si só, e a aprendizagem é necessária para resolver o problema. Por exemplo, é difícil programar um computador para reconhecer rostos familiares sem usar algoritmos de aprendizado (RUSSELL; NORVIG; DAVIS, 2010).

Existem três tipos principais de aprendizado de máquina: a aprendizagem não supervisionada, a aprendizagem por reforço e a aprendizagem supervisionada. Na aprendizagem não supervisionada, o sistema aprende a encontrar padrões ou estruturas do mesmo grupo nos dados de entrada por conta própria. Já no aprendizado por reforço, o sistema aprende por meio de tentativa e erro, tomando decisões com base nas recompensas e punições que recebe por suas ações. E por último, na aprendizagem supervisionada, o sistema é treinado com dados previamente rotulados, aprendendo a relacionar os rótulos com os dados brutos.

2.2 Redes Neurais Artificiais

As redes neurais artificiais são um dos principais algoritmos do Aprendizado de Máquina que foram desenvolvidos para simular o funcionamento dos neurônios do cérebro humano. Essas redes são compostas por um grande número de neurônios artificiais interconectados, organizados em camadas, que são capazes de aprender com exemplos e melhorar seu desempenho ao longo do tempo. As redes neurais artificiais são particularmente úteis em problemas complexos, como classificação de imagens, reconhecimento de fala e processamento de linguagem natural.

2.2.1 Perceptrons

Em 1958, Rosenblatt propôs o perceptron como o primeiro modelo de aprendizagem supervisionada. O perceptron é uma forma básica de rede neural usada para

classificar padrões que podem ser separados linearmente, ou seja, padrões que estão em lados opostos de um hiperplano. Essencialmente, o perceptron consiste em um único neurônio com pesos sinápticos ajustáveis e *bias* (viés) (HAYKIN, 2001).

Na figura 2.1, podemos observar um perceptron que recebe um vetor de valores reais como entrada. Esse perceptron realiza uma combinação linear dessas entradas e, em seguida, produz uma saída de 1 se o resultado dessa combinação linear for maior que um limite específico, caso contrário, produz uma saída de -1. Cada w_i é uma constante com valor real, ou peso, que determina a contribuição da entrada x_i para a saída do perceptron (MITCHELL, 1997).

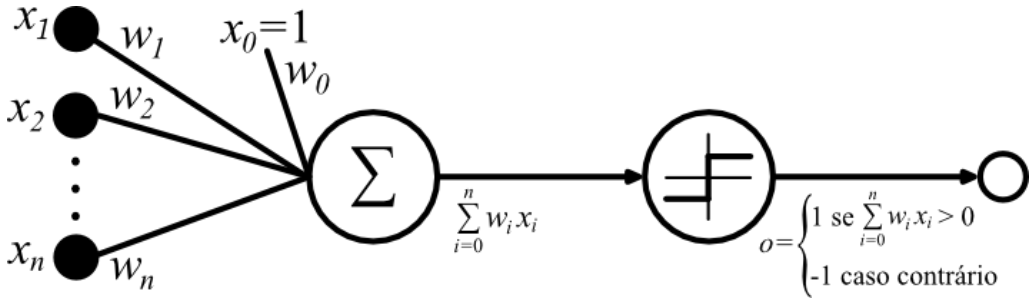

Figura 2.1: Perceptron
Fonte: (MITCHELL, 1997)

Normalmente, as redes neurais de múltiplas camadas (*Multilayer Perceptron* - MLP), são compostas por uma camada de entrada com unidades sensoriais, uma ou mais camadas ocultas com nós de processamento e uma camada de saída. Segundo Haykin (2001), as MLPs têm sido amplamente utilizados com sucesso para resolver problemas complexos por meio do treinamento supervisionado com o algoritmo popular chamado *error backpropagation*. Esse algoritmo se baseia na regra de aprendizagem que corrige os erros cometidos durante o treinamento, ajustando os pesos da rede neural.

De acordo com Haykin (2001), um perceptron de múltiplas camadas apresenta três características distintas:

- Cada neurônio na rede possui uma função de ativação não linear suave, em contraste com a limitação utilizada no perceptron de Rosenblatt. Essa não linearidade suave permite um processamento mais flexível dos dados, sendo

diferenciável em qualquer ponto.

- A rede possui uma ou mais camadas ocultas de neurônios, que não estão diretamente ligados à entrada ou à saída da rede. Esses neurônios ocultos permitem que a rede aprenda tarefas complexas, extraindo gradualmente as características mais relevantes dos padrões de entrada.

- A rede exibe um alto nível de conectividade, determinado pelas sinapses. Alterações na conectividade exigem ajustes nas conexões sinápticas ou em seus pesos. Isso possibilita a flexibilidade da rede em adaptar-se a diferentes problemas e aprender de maneira eficiente.

2.2.2 Transformers

Transformer é uma arquitetura de rede neural proposta no trabalho Vaswani et al. (2017), que se destacou ao obter bons resultados em diversas tarefas, como tradução de textos, question answering e language inference. Sua característica principal é o uso de mecanismos de *self-attention*, tanto no *encoder* quanto no *decoder*.

De acordo com Lin et al. (2021), a camada de *self-attention* apresenta algumas vantagens, como flexibilidade para lidar com *inputs* de tamanho variável e a capacidade de realizar operações de forma paralela, o que a diferencia das camadas recorrentes.

A arquitetura do Transformer serviu de base para o desenvolvimento de diversos modelos e suas variações. Esses modelos, como o GPT-2 (RADFORD et al., 2019), BERT (DEVLIN et al., 2019) e RoBERTa (LIU et al., 2019), foram influenciados pelo sucesso do Transformer e trouxeram avanços significativos em várias tarefas de processamento de linguagem natural. Eles representam uma evolução na forma como os modelos de linguagem são construídos e demonstram a importância da arquitetura do Transformer no campo da inteligência artificial.

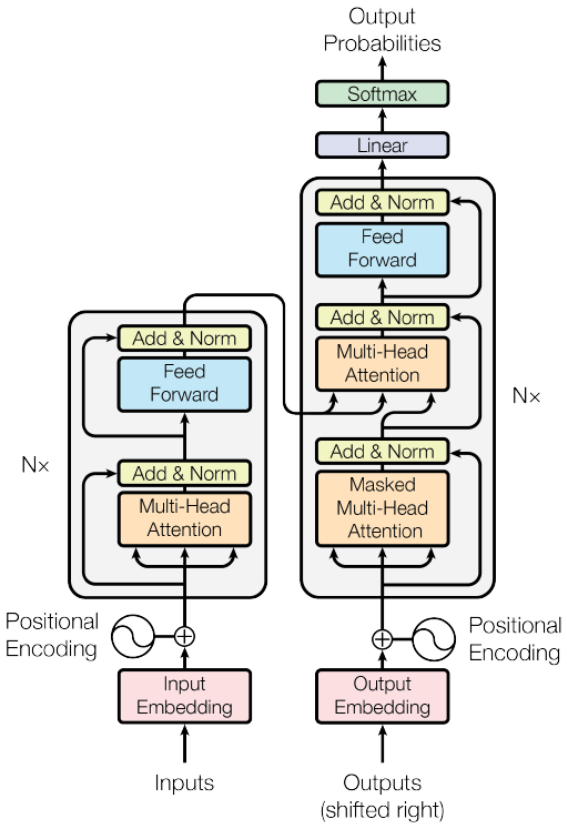

Figura 2.2: Proposta inicial da arquitetura de um Transformer
Fonte: (VASWANI et al., 2017)

2.2.3 Redes Neurais Siamesas

As redes neurais siamesas são baseadas em uma arquitetura que emprega duas redes neurais idênticas, compartilhando os mesmos pesos. Essas redes são geralmente utilizadas em conjunto com uma métrica de similaridade, como a similaridade de cosseno, para calcular a semelhança entre os dados de entrada. Para realizar o cálculo da similaridade de cosseno é utilizada a seguinte fórmula:

$$\frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \cdot \|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i \cdot B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}},$$

sendo A e B os vetores de saida de uma rede neural siamesa. A figura 2.3 ilustra uma arquitetura de rede neural siamesa que emprega a similaridade de cosseno como medida.

Essa arquitetura siamesa tem sido empregada em trabalhos anteriores para

obter a similaridade entre faces (CHOPRA; HADSELL; LECUN, 2005), assinaturas (BROMLEY et al., 1993) e textos (NECULOIU; VERSTEEGH; ROTARU, 2016).

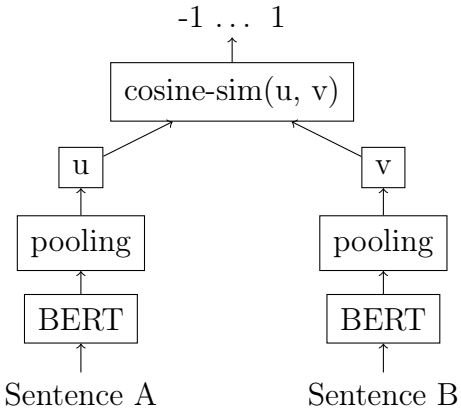

Figura 2.3: Arquitetura do SBERT utilizada para computar a similaridade
Fonte: (REIMERS; GUREVYCH, 2019)

2.3 Processamento de Linguagem Natural

Segundo Bird, Klein e Loper (2009), diferente das linguagens artificiais, como as linguagens de programação e as notações matemáticas, as linguagens naturais sofrem alterações à medida que passam de gerações, tornando-se difícil definir regras explícitas. Isso ocorre devido à natureza complexa e dinâmica de uma linguagem, que incluem variações culturais, regionais, ambiguidades e mudanças ao longo do tempo. Além disso, as linguagens naturais apresentam características como polissemia, onde uma mesma palavra pode ter múltiplos significados dependendo do contexto, e ambiguidade estrutural, em que uma frase pode apresentar mais de uma interpretação possível.

O Processamento de Linguagem Natural (PLN) surge então como uma subárea da inteligência artificial que tem como objetivo fazer com que computadores consigam compreender a linguagem natural humana. Existem duas classificações para o PLN, uma é a Compreensão de Linguagem Natural (*Natural Language Understanding* - NLU) e a outra é a Geração de Linguagem Natural (*Natural Language Generation* - NLG).

2.3.1 Pré-Processamento de Texto

O pré-processamento de texto é uma etapa importante na análise de dados textuais para que os computadores consigam compreender a linguagem natural. Algumas das principais técnicas e métodos para limpar e preparar um conjunto de dados, segundo Kowsari et al. (2019), são:

- Tokenização: processo de dividir um texto em palavras, frases, símbolos ou outros elementos significativos chamados *tokens*.
- *Stop words*: processo de remoção de palavras comuns que não possuem significado importante para a classificação de texto.
- Capitalização: processo onde todas as letras são transformadas em minúsculas. No entanto, isso pode causar problemas de interpretação para algumas palavras, como gírias e abreviações.
- Gírias e abreviações: para evitar problemas no processo de capitalização, esse processo visa trocar abreviações pelas palavras por extenso e as gírias são convertidas para a linguagem formal.
- Remoção de ruído: processo que envolve a exclusão de caracteres desnecessários, como pontuação e caracteres especiais, que podem prejudicar algoritmos de classificação.
- Correção ortográfica: processo opcional do pré-processamento para tratar erros de digitação que são encontrados em textos e documentos, especialmente em dados textuais de mídias sociais.
- *Stemming*: processo onde as palavras são modificadas para obter variantes usando diferentes processos linguísticos, como afixação. Esta etapa é importante para diminuir a quantidade de palavras que têm o mesmo significado semântico, visto que uma palavra pode aparecer em diferentes formas, por exemplo, um substantivo no singular e plural.

- *Lemmatization*: processo onde substitui o sufixo de uma palavra por outro sufixo ou remove completamente o sufixo de uma palavra para obter a forma básica da palavra.

2.3.2 Processamento de Texto

Uma vez feito todo o pré-processamento de texto, chega o momento de realizar o processo de extração de características. Esta etapa é responsável por transformar os documentos de texto pré-processados em representações numéricas ou vetoriais para que possam ser utilizadas por algoritmos de aprendizado de máquina.

A extração de características tem como objetivo identificar as características relevantes dos dados de entrada para uma tarefa em questão. Isso envolve reconhecer padrões, estruturas, relações ou propriedades que possam auxiliar na classificação, agrupamento, detecção de sentimentos ou outras atividades de processamento de texto. De acordo com Kowsari et al. (2019), os dois métodos mais comuns de extração de características são: técnicas de palavras ponderadas (*weighted word*) e incorporação de palavras (*word embedding*). Nas subseções a seguir, serão apresentados dois métodos de técnicas de palavras ponderadas, sendo eles Bag-of-Word (BoW) e TF-IDF, e dois métodos de incorporação de palavras, sendo eles Word2Vec e Doc2Vec.

2.3.2.1 Bag-of-Words

A técnica de BoW, demonstrada na figura 2.4, é um método em que todas as palavras de um documento são adicionadas a um "saco". Isso permite que os textos sejam representados como vetores numéricos, em que cada posição do vetor corresponde a uma palavra presente no documento. Portanto, a estrutura gramatical e a ordem das palavras não são consideradas nessa técnica.

Além disso, os vetores numéricos podem ser representados de duas formas distintas. Na primeira, utiliza-se um valor binário (0 ou 1) para indicar se uma palavra está presente ou ausente no texto. Na segunda forma, a contagem da frequência das palavras é considerada, atribuindo-se valores numéricos correspondentes às ocorrências de cada palavra.

Entrada: *Never gonna give you up Never gonna let you down*
Saída: { *Never*: 2, *gonna*: 2, *you*: 2 *give*: 1, *up*: 1, *let*: 1, *down*: 1 }

Figura 2.4: Exemplo de processamento do BoW

2.3.2.2 TF-IDF

Segundo Zhang, Yoshida e Tang (2011), TF-IDF (*term frequency–inverse document frequency*), é um método que avalia a relevância de um termo dado a sua quantidade de ocorrências de um documento e a quantidade de ocorrências do termo no conjunto de documentos. Termos que aparecem com muita frequência no conjunto de documentos vão ter um peso menor que aqueles que com uma baixa quantidade de ocorrências.

2.3.2.3 Word2Vec

O método que é popularmente conhecido como Word2Vec foi proposto por Mikolov et al. (2013a), Mikolov et al. (2013b). Esse método consiste em criar representações vetoriais densas de palavras, com o objetivo de formar um espaço vetorial onde palavras com significados semânticos semelhantes estejam próximas umas das outras. Para a criação deste espaço vetorial, foi proposto dois modelos de redes neurais: CBOW (*Continuous Bag-of-Words*) e *Skip-gram*.

No CBOW, a camada de projeção é compartilhada por todas as palavras, resultando na projeção das palavras na mesma posição, onde seus vetores são calculados como a média. Como a ordem das palavras não afetam a projeção, esse modelo é chamado de *bag-of-word*. Além disso, o modelo também leva em consideração palavras do futuro durante o treinamento. Para melhorar o desempenho em uma tarefa específica, como classificar palavras corretamente, o modelo utiliza um classificador log-linear que recebe quatro palavras do futuro e quatro palavras do contexto histórico como entrada. O objetivo é prever a palavra atual com base nessas palavras contextuais.

O *Skip-gram* tem uma arquitetura similar ao CBOW, no entanto ele tenta maximizar a classificação de uma palavra de acordo com outra palavra na mesma

frase. Esse modelo usa cada palavra atual como entrada para um classificador log-linear com uma camada de projeção contínua e é previsto palavras dentro de uma determinada faixa antes e depois da palavra atual. Aumentando a faixa, melhora a qualidade dos vetores de palavras resultantes, porém é aumentado também a complexidade computacional. Considerando que palavras mais distantes tendem a ter menos relação com a palavra atual em comparação às próximas, é atribuído um peso menor para essas palavras distantes, reduzindo a amostragem dessas palavras nos exemplos de treinamento.

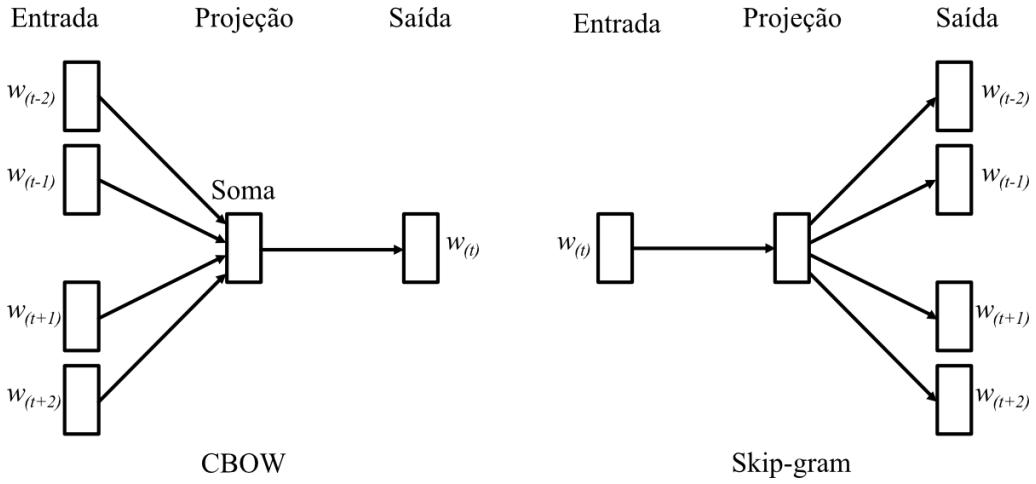

Figura 2.5: Representação dos modelos CBOW e Skip-gram
Fonte: (MIKOLOV et al., 2013a)

2.3.2.4 Doc2Vec

O *Doc2Vec*, originalmente chamado de *Paragraph Vector*, é um modelo que visa representar parágrafos, textos ou documentos em um espaço vetorial contínuo, sendo diferente do *Word2Vec*, que representa apenas palavras em um espaço vetorial. Esse modelo foi proposto por Le e Mikolov (2014) e tem como objetivo a aprendizagem de representações vetoriais de textos com base no contexto, seja de um parágrafo ou do documento inteiro, que ele aparece.

Como na figura 2.6, cada parágrafo é mapeado para um vetor único, representado como uma coluna na matriz D, e cada palavra também é mapeada para um vetor único, representado por uma coluna na matriz W. O vetor do parágrafo e os vetores de palavras são combinados por meio de uma média ou concatenação para prever a

próxima palavra em um contexto (LE; MIKOLOV, 2014).

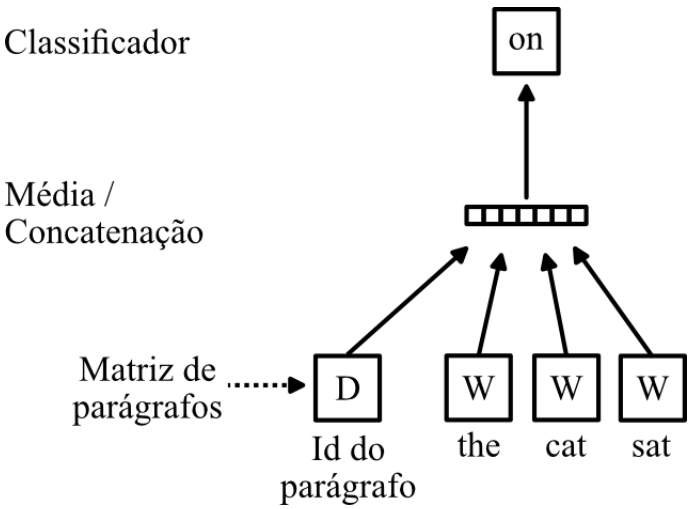

Figura 2.6: Representação do modelo *Paragraph Vector*
Fonte: (LE; MIKOLOV, 2014)

2.3.3 *Question Answering*

Question Answering (QA) é uma área de pesquisa que visa fornecer respostas diretas e precisas para perguntas, diferentemente dos sistemas de recuperação de informação convencionais que apenas retornam documentos relevantes. Ao combinar técnicas de Recuperação de Informação, Extração de Informação e Processamento de Linguagem Natural, os sistemas de QA buscam compreender as perguntas dos usuários e recuperar as respostas relevantes dos documentos. Isso permite que os usuários obtenham informações específicas e solucionem suas dúvidas de forma mais eficiente (ALLAM; HAGGAG, 2012).

Existem duas categorias principais de sistemas de QA: os de domínio aberto e os de domínio fechado. O QA de domínio aberto lida com perguntas abrangendo diversos assuntos e baseia-se em ontologias universais e informações disponíveis na *World Wide Web*. Por outro lado, o QA de domínio fechado trata de perguntas dentro de um domínio específico, como música ou previsão do tempo. Nesses casos, o sistema de QA é construído com base em técnicas de processamento de linguagem natural e na criação de uma ontologia específica para o domínio em questão (ALLAM; HAGGAG, 2012).

Harabagiu, Paşca e Maiorano (2000) exemplificam uma arquitetura que processa a pergunta por meio de transformações lógicas e semânticas, extrai respostas com base em palavras-chave obtidas no processamento da pergunta e realiza o processamento das respostas extraídas usando transformações lógicas para a comprovação de teoremas, descartando respostas que não podem ser comprovadas.

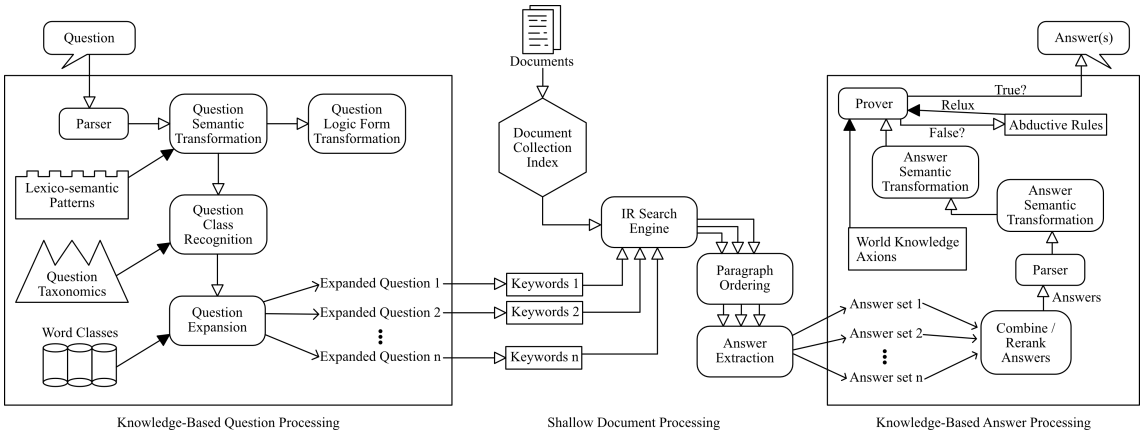

Figura 2.7: Arquitetura do sistema proposta em Harabagiu, Paşca e Maiorano (2000)
Fonte: (HARABAGIU; PAsCA; MAIORANO, 2000)

O módulo *Knowledge-Based Question Processing*, presente na figura 2.7, tem a função de processar a pergunta feita pelo usuário. Esse processo envolve a conversão da pergunta em uma representação semântica, como é possível observar na figura 2.8, para ser combinada com a representação semântica dos nós da taxonomia relacionada à pergunta. Isso permite a classificação adequada da pergunta (HARABAGIU; PAsCA; MAIORANO, 2000).

Já o módulo *Shallow Document Processing* tem a responsabilidade de processar os dados provenientes do módulo anterior, visando identificar documentos que contenham possíveis respostas. Para realizar essa busca de documentos, são utilizados os nós da taxonomia associados à pergunta para obter palavras-chave relevantes. Uma vez obtidos os documentos, as respostas são extraídas, combinadas e organizadas (HARABAGIU; PAsCA; MAIORANO, 2000).

Finalmente, o módulo *Knowledge-Based Answer Processing* é responsável por processar as respostas obtidas pelo módulo anterior e fornecer resposta(s) com base na pergunta inicial do usuário. Após receber as respostas do módulo anterior, elas

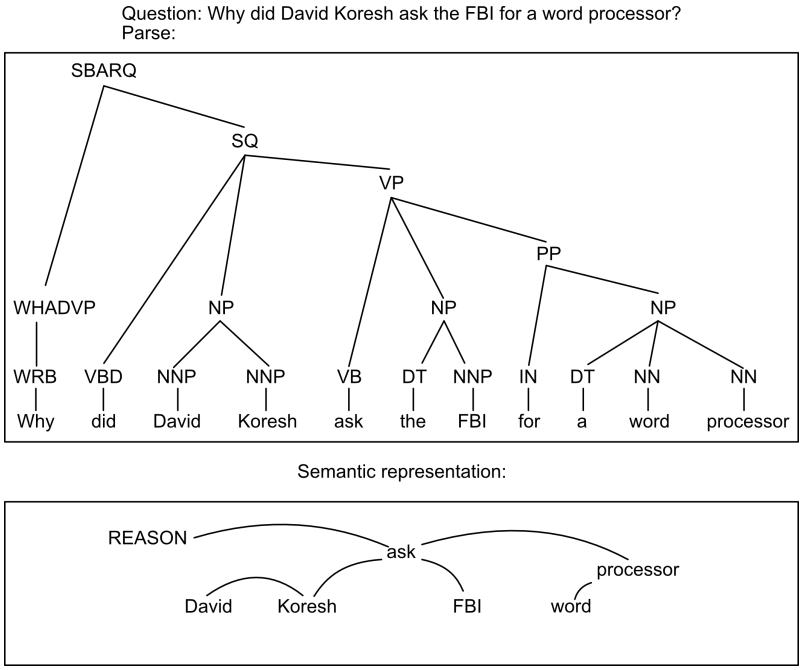

Figura 2.8: Transformação semântica da pergunta
Fonte: (HARABAGIU; PAsCA; MAIORANO, 2000)

são convertidas em representações semânticas e, posteriormente, transformadas em representações lógicas para serem utilizadas em um provador de teoremas. Se o provador não conseguir provar uma resposta, ela é descartada; caso contrário, é retornada ao usuário (HARABAGIU; PAsCA; MAIORANO, 2000).

Na figura 2.8, cada sigla significa uma categoria que é utilizada para compreender a estrutura gramatical das frases. SBARQ representa uma oração subordinada interrogativa, WHADVP indica uma frase adverbial interrogativa, SQ representa uma oração subordinada, WRB se refere a um advérbio interrogativo, VBD indica verbo no passado, NP significa uma frase nominal, VP representa uma frase verbal, NNP se refere a um substantivo próprio, VB significa verbo base, DT indica um determinante, PP representa uma frase preposicional e NN se refere a um substantivo.

BERT (DEVLIN et al., 2019) propõe uma arquitetura baseada em Transformer, diferindo significativamente do modelo anteriormente mencionado. A fim de obter a resposta desejada, o modelo requer a inclusão de um contexto juntamente com a pergunta, possibilitando a extração da resposta desejada a partir desse mesmo contexto. Isso se diferencia do modelo mencionado anteriormente, que consultava

uma coleção de documentos em busca de respostas.

2.4 Rede Mundial de Computadores

A Rede Mundial de Computadores (World Wide Web - WWW), também conhecida como Internet, é definida como um universo de informações globalmente acessíveis por meio de redes. Sua existência marca o fim de uma era de problemas causados pela incompatibilidade entre sistemas de computadores (BERNERS-LEE, 1997). Atualmente a Internet desempenha um papel fundamental na forma como as pessoas interagem, comunicam-se e compartilham informações online. Para isso, existem duas arquiteturas que são as mais utilizadas nas aplicações da rede mundial de computadores: cliente-servidor e par a par (*Peer-to-Peer* - P2P).

Na arquitetura cliente-servidor, um servidor está sempre ativo e atende às solicitações de vários clientes. Um exemplo comum é a aplicação Web, em que um servidor Web em funcionamento responde às solicitações dos navegadores dos clientes. Os clientes não se comunicam diretamente entre si nessa arquitetura. O servidor tem um endereço IP fixo e conhecido, permitindo que os clientes o contactem enviando pacotes. Algumas aplicações populares que utilizam essa arquitetura são a Web, FTP (*File Transfer Protocol*), Telnet e e-mail (KUROSE; ROSS, 2013). A figura 2.9 representa esse tipo de arquitetura.

A arquitetura P2P é caracterizada pela comunicação direta entre pares conectados, sem depender de servidores dedicados em centros de dados. Os pares são computadores controlados por usuários individuais, geralmente localizados em residências, universidades e escritórios. Essa arquitetura é amplamente utilizada em aplicações populares, como compartilhamento de arquivos, telefonia pela Internet e IPTV (*Internet Protocol Television*). Além disso, algumas aplicações podem ter arquiteturas híbridas, combinando elementos cliente-servidor e P2P (KUROSE; ROSS, 2013). A figura 2.10 representa esse tipo de arquitetura.

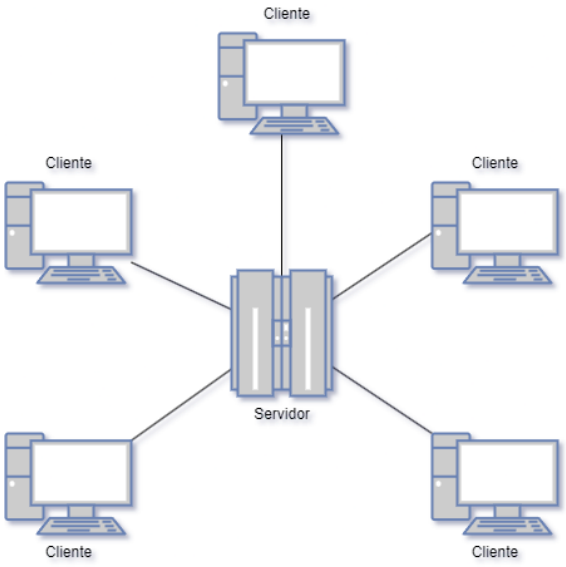

Figura 2.9: Exemplo de arquitetura cliente-servidor

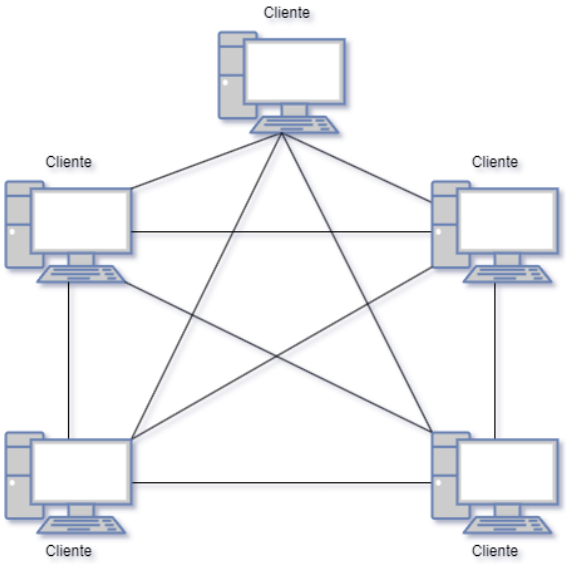

Figura 2.10: Exemplo de arquitetura P2P

2.4.1 Servidor Web

No servidor Web, a comunicação é feita através do protocolo de transferência de hipertexto (*Hypertext Transfer Protocol* - HTTP), que permite a solicitação e transferência de objetos entre clientes e o servidor. Quando um usuário digita um endereço no navegador Web, ele está enviando uma solicitação HTTP para o servidor que armazena os objetos daquele site. Ao receber essa solicitação, o servidor processa

a solicitação e envia uma resposta HTTP contendo os dados solicitados, por exemplo, uma página HTML (*Hypertext Markup Language*), um arquivo CSS (*Cascading Style Sheets*), uma imagem, um *script* e entre outros objetos.

Como a Web utiliza o hipertexto como forma de organizar informações, é utilizado a linguagem de marcação HTML para a escrita desses documentos de hipertexto. O HTML permite a estruturação e formatação do conteúdo, além de possibilitar a inclusão de *links* para outras solicitações HTTP, o que permite uma navegação entre diferentes páginas.

2.4.2 API

Os sistemas finais ligados à Internet possuem uma Interface de Programação de Aplicação (*Application Programming Interface* - API) que define como um programa solicita o envio de dados à infraestrutura da Internet para um programa específico em outro sistema final. Essa API é composta por um conjunto de regras que o software emissor deve seguir para que os dados sejam corretamente transmitidos ao programa de destino (KUROSE; ROSS, 2013). Ou seja, uma API é um conjunto de regras e protocolos que permite diferentes softwares e sistemas se comunicarem entre si de forma padronizada.

2.4.2.1 API RESTful

Uma API RESTful (*Representational State Transfer*) é uma arquitetura de software que se baseia nos princípios do protocolo HTTP, ou seja, ela utiliza métodos HTTP como GET e POST para fazer a comunicação entre o cliente e o servidor. Esse tipo de API são projetadas para serem simples, leves e escaláveis, perfeitas para serem usadas por dispositivos móveis e dispositivos de Internet das Coisas.

Capítulo 3

SACI PeReRe

Este capítulo tem como objetivo fornecer uma visão detalhada das principais motivações que estimularam a escolha do tema deste trabalho, além de serem apresentados também os trabalhos relacionados que embasam o estudo e ajudam a definir qual é a melhor abordagem para solucionar o problema.

Para a solucionar o problema, é proposto ainda neste capítulo um sistema para auxiliar na resolução de dúvidas de terceiros sobre um determinado assunto, tendo então como objetivo responder rapidamente o usuário.

3.1 Motivação

Nos últimos anos, tem sido claramente observado o notável crescimento da Internet, como evidenciado pelos dados do relatório de Kemp (2023), que indicam que até janeiro de 2023 a Internet possuía um número de usuários de 5,16 bilhões, um crescimento de 1,9% nos últimos 12 meses. A Internet é uma revolução tecnológica que transformou os hábitos e a forma como as pessoas interagem com o mundo ao nosso redor. Ela trouxe consigo uma imensidão de possibilidades, desde a rápida disseminação de informações até o surgimento de novas formas de comunicação. Nesse contexto, os chatbots, como o ChatGPT[1], têm desempenhado um papel cada

[1]https://openai.com/blog/chatgpt

vez mais relevante.

O crescimento da Internet tem tido um impacto profundo na vida das pessoas, alterando a maneira como trabalham, aprendem, se relacionam e se divertem. Com a facilidade de acesso a informações, a Internet se tornou uma fonte infinita de conhecimento, possibilitando que as pessoas aprendam sobre qualquer assunto com apenas alguns cliques. Essa disponibilidade instantânea de informações tem causado uma necessidade de busca por respostas mais rápidas, em tempo real, de acordo com Rushkoff (2013).

Um campo em que um sistema de chatbot pode ser especialmente benéfico é o contexto educacional, mais especificamente nas universidades públicas. Segundo dados do IBGE - Instituto Brasileiro de Geografia e Estatística (2022), aproximadamente 4 milhões de pessoas ingressaram em cursos de graduação. Com esse número de estudantes ingressando nessas instituições, muitas vezes é difícil para os profissionais responderem todas as dúvidas de forma ágil e eficiente. É nesse ponto que a criação de um chatbot personalizado pode desempenhar um papel importante no alívio dessa demanda.

Desenvolver esse sistema de chatbot auxiliaria estudantes universitários no fornecimento de respostas rápidas e precisas sobre diversos tópicos relacionados à universidade, como informações sobre cursos, processos seletivos, calendário acadêmico, bolsas de estudo e muito mais. Com um chatbot bem projetado, os profissionais teriam mais tempo para lidar com questões mais complexas e específicas, enquanto as perguntas mais comuns e genéricas seriam respondidas prontamente pelo chatbot.

Além disso, esse sistema de chatbot pode oferecer suporte 24 horas por dia, 7 dias por semana, garantindo que os estudantes tenham acesso às informações de que precisam a qualquer momento, independentemente do horário ou da disponibilidade dos profissionais. Essa disponibilidade constante de recursos pode contribuir para uma melhor experiência acadêmica.

3.2 Trabalhos Relacionados

Reimers e Gurevych (2019) desenvolveram uma variação do modelo BERT de Devlin et al. (2019) que performa melhor em tarefas como Similaridade Textual Semântica (*Semantic Textual Similarity* - STS) chamado SBERT. Umas das principais diferenças se dá pelo uso de redes neurais siamesas/trigêmeas, que permite gerar *sentence embeddings* semanticamente significativos, isso é, sentenças com valores semânticos semelhantes estão próximas umas das outras no espaço vetorial. Esse tipo de tarefa já era feito no BERT, mas sua arquitetura não permitia um desempenho muito bom. As avaliações realizadas demonstram que o SBERT alcançou um desempenho significativamente notável em comparação com outros modelos, tais como BERT e GloVe (PENNINGTON; SOCHER; MANNING, 2014).

Lee et al. (2019) propõem um experimento para avaliar a carga de trabalho de dois departamentos de uma universidade coreana quando assistidos por um chatbot que responde FAQ (*Frequently Asked Questions*). Após utilizar o NASA-TLX[2] para avaliar o estado emocional dos funcionários antes e depois da implementação do chatbot, constatou-se um impacto positivo na percepção da carga de trabalho. Além disso, notou-se uma mudança no padrão de envio de e-mails, ocorrendo uma redução na frequência de envio de e-mails relacionados a requisitos de graduação e calendário acadêmico durante o período em que o chatbot estava ativo em um dos departamentos avaliados.

O trabalho proposto por Júnior e Barbosa (2018) visa desenvolver um chatbot para responder FAQs. Para atingir o objetivo, foi utilizado o método *Latent Semantic Indexing* (LSI) para identificar a similaridade entre a pergunta feita pelo usuário e as perguntas do *dataset*.

No trabalho da Alves (2021) o objetivo é a construção de um chatbot para responder dúvidas de alunos, servidores e cidadãos que não têm vínculo com o CEFET/RJ no Facebook[3]. Para auxiliar no desenvolvimento desse projeto, foi utilizado o modelo de desenvolvimento de software cascata juntamente com conceitos

[2]https://humansystems.arc.nasa.gov/groups/TLX/
[3]https://www.facebook.com/

de *Design Thinking*. Para a implementação do chatbot foi utilizada uma ferramenta já existente no mercado, o Manychat[4]. O lado positivo dessa abordagem é sua facilidade de implementação já que não envolve programação, no entanto diminui a possibilidade de exportar o chatbot para outras plataformas, sendo limitado apenas pelas plataformas disponíveis pela ferramenta.

3.3 Proposta

Este trabalho propõe a criação do SACI PeReRe (Sistema de Auxílio à Comunicação Interativa para Perguntas e Respostas Recorrentes). Será desenvolvido um sistema administrativo Web para o cadastro de perguntas e respostas, bem como um modelo de redes neurais para identificar automaticamente as respostas às perguntas. Além disso, será implementado um *bot* para o Discord que permitirá que os usuários finais façam perguntas e que essas sejam processadas pelo modelo de rede neural. Tanto o sistema administrativo Web quanto o sistema do modelo de redes neurais acessarão o mesmo banco de dados, conforme é possível observar na figura 3.1.

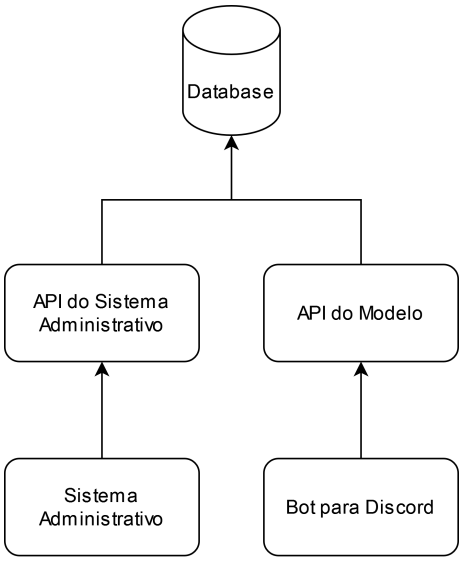

Figura 3.1: Arquitetura do sistema

[4]https://manychat.com/

3.3.1 Sistema Administrativo Web

O sistema administrativo Web será o espaço destinado aos servidores públicos das universidades para cadastrar perguntas e respostas. Para possibilitar a utilização do sistema por diferentes departamentos, cursos e/ou instituições, será implementado um sistema hierárquico de cadastro, representado pela figura 3.2. Esse sistema hierárquico incluirá tópicos, dentro de cada tópico haverá categorias e, por fim, dentro de cada categoria estarão as perguntas, acompanhadas de suas respectivas respostas.

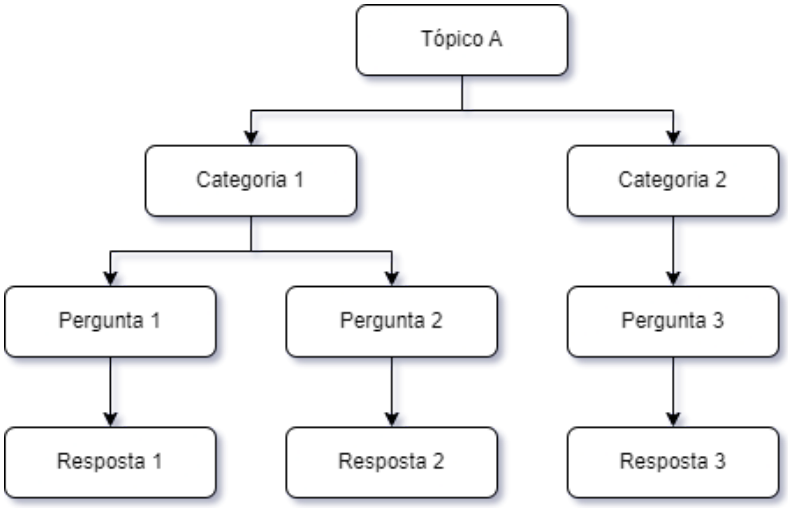

Figura 3.2: Hierarquia das perguntas

Esse sistema também terá um painel com estatísticas fornecidas a partir de *feedback* dos usuários sobre as respostas que serão fornecidas pelo modelo de redes neurais. Com essas estatísticas será possível identificar a precisão e eficácia do modelo de redes neurais, além de permitir que o usuário do sistema administrativo possa identificar e reformular as perguntas de maneira que melhore a precisão do modelo.

Para auxiliar os usuários do sistema administrativo na reformulação mais eficiente das perguntas, o sistema contará com um histórico das perguntas feitas pelos usuários finais. Isso permitirá que os usuários do sistema administrativo se baseie na forma pela qual os usuários finais redigem suas perguntas, a fim de aprimorá-las de maneira adequada.

Além disso, o sistema contará com uma lista de perguntas não respondidas. Essa

listagem permitirá identificar novas perguntas que ainda não foram cadastradas no sistema. Com essa funcionalidade, será possível garantir que todas as perguntas dos usuários finais sejam consideradas, mesmo aquelas que ainda não fazem parte do banco de dados.

3.3.2 Sistema do Modelo de Rede Neural

Com relação ao sistema do modelo de redes neurais, ele funcionará de maneira independente do sistema administrativo. Esse sistema terá a função de identificar as respostas para as perguntas dos usuários, o que requer a construção de uma API capaz de suportar a integração com diversos aplicativos externos ao nosso sistema, como Discord, WhatsApp, Telegram, entre outros. Essa abordagem possibilitará a utilização do modelo em diferentes plataformas e garantirá a interação com uma ampla gama de usuários, aumentando a acessibilidade e a utilidade do sistema como um todo.

Essa API terá a importante função de gerenciar não apenas a identificação das respostas, mas também o tratamento dos *feedbacks* enviados pelos usuários. Essa abordagem permitirá uma interação bidirecional com os usuários, possibilitando que eles forneçam *feedbacks* sobre as respostas identificadas pelo sistema de redes neurais.

Para identificar as respostas, adotaremos um modelo de STS, que permitirá a comparação entre a pergunta do usuário final e as perguntas cadastradas no banco de dados pelo sistema administrativo. Perguntas que tiverem uma similaridade abaixo de um limiar pré-estabelecido serão salvas no banco de dados como perguntas não respondidas.

3.3.3 *Bot* para Discord

Será desenvolvido um *bot* com a finalidade de esclarecer as dúvidas dos usuários finais do SACI PeReRe e para testar o modelo de rede neural. Para isso, o *bot* fará o uso da API do modelo para processar as perguntas dos usuários e tratará o retorno para exibir as respostas relevantes.

No entanto, antes de apresentar a resposta ao usuário, o *bot* verificará se a pontuação de similaridade está acima de um limiar predefinido. Caso a similaridade esteja acima, o *bot* exibirá a resposta ao usuário e, além disso, permitirá que ele forneça feedback sobre a qualidade e utilidade da resposta recebida.

Por outro lado, se a similaridade entre a resposta e a pergunta não atingir o limiar estabelecido, o *bot* informará ao usuário que a pergunta não está cadastrada na base de dados e, portanto, não pode fornecer uma resposta adequada naquele momento. Nesse caso, o *bot* também poderá sugerir ao usuário que reformule sua pergunta ou busque outros meios de obter a resposta desejada.

3.3.4 Requisitos de Sistema

Para identificar os requisitos do sistema, foi realizada uma elicitação de requisitos, que consistiu principalmente na técnica de *brainstorming*. A partir desta técnica, foram geradas diversas ideias que foram discutidas para identificar as necessidades dos usuários e as funcionalidades que o sistema deve oferecer para atendê-las. Os requisitos identificados serão apresentados na tabela 3.1 nesta seção e servirão como base para o desenvolvimento do sistema.

Código	Descrição do Requisito
RF 01	O sistema deve permitir a criação de um novo tópico informando um nome e podendo ou não ter uma descrição.
RF 02	O sistema deve permitir a edição do nome e/ou descrição de um tópico.
RF 03	O sistema deve permitir a exclusão de um tópico, sendo que ao excluir o tópico, também serão excluídas as categorias, perguntas e respostas associados ao mesmo.
RF 04	O sistema deve incluir a listagem de tópicos cadastrados.
RF 05	O sistema deve permitir a criação de uma nova categoria dentro de um tópico informando um nome e podendo ou não ter uma descrição.

Código	Descrição do Requisito
RF 06	O sistema deve permitir a edição do nome e/ou descrição de uma categoria.
RF 07	O sistema deve incluir a listagem de categorias associadas aos seus respectivos tópicos.
RF 08	O sistema deve permitir a exclusão de uma categoria, sendo que ao excluir a categoria, também serão excluídas as perguntas e respostas associadas à mesma.
RF 09	O sistema deve permitir a criação de uma nova pergunta e resposta dentro de uma categoria informando qual é a pergunta e a resposta.
RF 10	O sistema deve permitir a edição de uma pergunta e/ou resposta.
RF 11	O sistema deve permitir a exclusão de uma pergunta e resposta.
RF 12	O sistema deve incluir a listagem de perguntas e respostas associadas às suas respectivas categorias.
RF 13	O sistema deve permitir o cadastro de um novo usuário informando os campos nome, e-mail e nome de usuário.
RF 14	O sistema deve permitir que um usuário edite suas informações de perfil, tais como nome, e-mail, nome de usuário e senha.
RF 15	O sistema deve permitir que um usuário administrador transforme outros usuários como administradores.
RF 16	O sistema deve incluir a listagem de usuários cadastrados.
RF 17	O sistema deve ter um painel de estatísticas para cada plataforma cadastrada.
RF 18	O sistema deve ter um histórico de perguntas feitas em cada plataforma.
RF 19	O sistema deve ter um histórico de perguntas que o modelo de rede neural não conseguiu responder.
RF 20	O sistema deve incluir uma API para ter acesso ao modelo de rede neural.

Código	Descrição do Requisito
RF 21	O sistema deve ser capaz de responder perguntas a partir de interações de usuários por meio de automações em plataformas cadastradas.
RF 22	O sistema deve permitir gerar um relatório contendo todos os tópicos, suas categorias, perguntas e respostas.
RF 23	O sistema deve permitir adicionar tópicos e categorias como favoritos.

Tabela 3.1: Requisitos funcionais

Capítulo 4

Projeto

Neste capítulo, abordaremos três seções distintas. A primeira seção apresentará as tecnologias utilizadas, destacando as ferramentas e recursos empregados em cada etapa do desenvolvimento. Em seguida, a segunda seção focará no desenvolvimento do sistema, explorando a divisão entre a parte administrativa e a parte do modelo de rede neural. Por fim, na terceira seção, apresentaremos os resultados obtidos com a implementação do sistema.

4.1 Tecnologias Utilizadas

Para a implementação de um sistema de chatbot, é essencial a escolha de tecnologias adequadas para cada parte do sistema. Desde a coleta de dados, modelos de rede neural até a interface com o usuário final, diversas ferramentas e recursos devem ser considerados. Nesta seção, serão apresentadas as principais tecnologias utilizadas na implementação do sistema, bem como seus benefícios.

4.1.1 Sistema Administrativo

Para o sistema administrativo, foi utilizado o *framework* Vue para JavaScript para construir interfaces de usuário usando um modelo de programação declarativa e baseada em componentes. O *framework* tem se tornado bastante popular para

a criação de *Single-Page Application* (SPA). Os SPAs são aplicações Web que carregam todo o seu conteúdo em uma única página, permitindo que as transições entre seções ocorram de forma dinâmica, sem a necessidade de recarregar a página inteira ou mudar para uma nova página. Isso melhora a experiência do usuário, pois as transições são mais rápidas e suaves, e também ajuda a economizar o uso de recursos do navegador, quando utilizado corretamente pelo desenvolvedor.

4.1.2 Sistema Administrativo API

Para a API do sistema administrativo, foi utilizado o Express, que é um *framework* Web para Node minimalista e flexível que permite construir diversos tipos de serviços Web, como APIs e páginas Web, com poucas linhas de código. Por não ser um *framework* opinado, deu liberdade para construir uma arquitetura própria e um maior controle para toda a aplicação.

4.1.3 Banco de Dados

Já para o banco de dados, foi optado pelo PostgreSQL, que é um Sistema de Gerenciamento de Banco de Dados (SGBD) relacional de código aberto, altamente reconhecido por sua estabilidade e confiabilidade. Sua escalabilidade e capacidade de lidar com grandes quantidades de dados o tornaram uma escolha ideal como base de dados para o sistema. Além disso, o PostgreSQL oferece recursos avançados, como suporte a JSON, índices *full-text* e consultas avançadas.

4.1.4 Bot para Discord

Para a criação do *bot* para o Discord, foram utilizadas duas bibliotecas para Java, sendo a primeira o JDA[1] que funciona como um *wrapper* para a API do Discord e seus eventos via WebSockets. Através dessa biblioteca, é possível criar *bots* que automatizam tarefas e interagem com usuários do Discord por meio de comandos. A função desses comandos fica a cargo do desenvolvedor e da limitação da API, sendo possível gerenciar e moderar servidores, criar *bots* de música e integrações entre

[1] https://github.com/discord-jda/JDA

serviços externos.

A segunda biblioteca utilizada foi o Curupira[2] que fornece uma camada de abstração acima do JDA e seus eventos. Utiliza a API *Reflection* do Java para fornecer uma interface simples e declarativa para lidar com *features* do Discord, como *slash commands*, *menus*, botões e *modals*, além de melhorar a manutenção do código.

4.1.5 API do Modelo

Para a API do modelo, foi utilizado o *framework* SentenceTransformers que serve para gerar *embeddings* de texto, imagens e sentenças. Por debaixo dos panos, utiliza o PyTorch para realizar o treinamento e inferência dos modelos. Em conjunto com o *framework*, são fornecidos diversos modelos pré-treinados que podem ser utilizados em um vasto conjunto de tarefas, como STS.

Esses modelos podem ser facilmente utilizados utilizando a plataforma Hugging Face[3], que concentra quase 200.000 modelos de aprendizado de máquina e mais de 30.000 *datasets* fornecidos gratuitamente por diversas pessoas e empresas de todo o mundo. No momento que esse texto foi escrito, 124 modelos para SentenceTransformers estavam disponíveis para uso.

Para a implementação da API, optou-se pela utilização o FastAPI, que é um *framework* Web moderno, utilizado para construir APIs em Python. Ele se destaca pela sua facilidade de uso, documentação automática, tipagem de dados, suporte a WebSockets e alta velocidade de processamento. A escolha do FastAPI para a criação da API para o modelo de rede neural se dá pela sua facilidade de implementação e integração com outras tecnologias utilizadas no projeto.

4.2 Desenvolvimento do Sistema

Nessa parte do trabalho, abordaremos as tecnologias utilizadas no desenvolvimento do sistema administrativo, do modelo de rede neural, do *bot* do Discord e do banco

[2]https://github.com/Softawii/curupira
[3]https://huggingface.co/

de dados. Será detalhado como foram empregadas para criar um sistema funcional, capaz de atender às demandas dos usuários e garantir uma experiência satisfatória.

4.2.1 Sistema Administrativo

A implementação do sistema administrativo foi realizada utilizando JavaScript tanto para o *front-end* quanto para o *back-end*. Para o desenvolvimento do *front-end*, foi utilizado o *framework* Vue.js em conjunto com a biblioteca de componentes Naive UI. O Vue.js é uma ferramenta que vem cada vez mais se tornando popular para construir interfaces de usuário interativas, permitindo uma renderização eficiente e uma experiência de desenvolvimento ágil. O uso do Naive UI[4] forneceu uma coleção de componentes estilizados e responsivos, acelerando o desenvolvimento da interface do sistema administrativo.

No que diz respeito ao *back-end*, o *framework* Express foi escolhido como base para a construção das APIs e rotas do sistema. O Express é uma estrutura leve e flexível que simplifica o desenvolvimento de aplicativos Web em Node.js, oferecendo recursos para criar *endpoints* de maneira rápida e eficiente. Além disso, o Prisma foi adotado para permitir a manipulação dos dados no banco de dados PostgreSQL. O Prisma permite a definição de modelos de dados em JavaScript e oferece uma camada abstrata para realizar consultas e operações de CRUD de forma intuitiva.

A fim de assegurar a qualidade e a integridade dos *endpoints* empregados pelo sistema administrativo, optou-se pela adoção do *framework* de testes Jest[5]. Apesar de não possuir suporte nativo para testes de APIs, é viável combinar sua utilização com outras bibliotecas, como o SuperTest[6], a fim de atingir tal propósito. Adicionalmente, foi estabelecido um pipeline de integração e entrega contínuas (CI/CD) no GitHub Actions, o qual realiza automaticamente a execução desses testes em cada envio efetuado ao repositório.

Essa abordagem arquitetural, no qual o *front-end* e *back-end* funcionam de forma

[4]https://www.naiveui.com
[5]https://jestjs.io/
[6]https://github.com/ladjs/supertest

independente, proporciona uma maior flexibilidade e liberdade ao desenvolvedor no processo de criação de recursos. Ao separar claramente as camadas de apresentação (*front-end*) e de lógica de negócios (*back-end*), é possível desenvolver novas funcionalidades e realizar modificações específicas sem afetar diretamente outros sistemas ou componentes da aplicação.

4.2.2 Sistema do Modelo de Rede Neural

No desenvolvimento do sistema do modelo de rede neural, foram utilizados dois componentes principais, sendo o primeiro o FastAPI, que desempenhou um papel fundamental como uma ferramenta eficiente no contexto do desenvolvimento do sistema. Por meio dessa API, foi possível receber solicitações dos serviços, processá-las de forma eficaz e fornecer respostas correspondentes de maneira rápida e confiável.

Já o segundo componente foi o *framework* SentenceTransformers, que é uma parte crucial para o sistema desenvolvido. Ele é utilizado para processar as perguntas dos usuários, que foram encaminhadas à API, utilizando um modelo pré-treinado. Através desse processamento, é gerada uma lista ordenada de forma decrescente. Para essa ordenação, é utilizada a pontuação de similaridade semântica entre as perguntas armazenadas no banco de dados e a pergunta feita pelo usuário. Esse recurso permite identificar as perguntas mais relevantes e semelhantes à pergunta do usuário, fornecendo informações valiosas para a resposta adequada.

Após o processamento realizado pelo modelo, a API retornava ao serviço as informações das perguntas similares, juntamente com suas respostas correspondentes. Cabe então ao serviço que fez a requisição interpretar esses dados de acordo com suas necessidades específicas. Por exemplo, é possível utilizar um limiar para filtrar perguntas com pontuações baixas e focar apenas naquelas com maior relevância, adaptando a resposta de acordo com os critérios estabelecidos.

4.2.2.1 Modelo

Antes de adotar os modelos e arquitetura atuais, foi conduzida uma série de experimentos utilizando diversas tecnologias. Inicialmente, foi desenvolvida uma rede neural que utilizava o método TF-IDF para classificar perguntas. No entanto, essa abordagem se mostrou ineficaz devido à simplicidade da base de dados utilizada neste trabalho, resultando em um modelo com baixa capacidade de generalização.

Foi testado também as técnicas word2vec e doc2vec. Essas abordagens exigiam um corpus consideravelmente grande e a necessidade de treinar os modelos sempre que os dados fossem atualizados, o que acabou tornando inviável de acordo com o nosso contexto. Além disso, foi testado o recurso de busca em texto completo nativo do PostgreSQL, que apresentou bom desempenho ao lidar com buscas por palavras-chave, porém revelou-se ineficiente quando sinônimos eram utilizados.

Por fim, foi analisado os modelos de *Question Answering*, que logo foi descartado por requererem um contexto específico para responder às perguntas, o que não se adequa com a ideia proposta para o sistema, visto que a nossa base armazenaria apenas perguntas e respostas. Com base nesses experimentos, foram filtradas as abordagens de aprendizado de máquina que não atenderam aos requisitos e, assim, identificamos os modelos com suporte à STS, os quais fornecem informações sobre a similaridade semântica entre textos. Com base nisso, foi estabelecido os seguintes critérios para a seleção dos modelos:

i Não seja necessário treinar a cada atualização da base de dados
   - Para garantir permitir que o sistema atualize em tempo real

ii Tenha suporte para português brasileiro

iii Não tenha um alto custo computacional e de memória para computadores
   - Para ter um bom desempenho em computadores de baixa performance e sem placa de vídeo dedicada

iv Seja open-source e gratuito

Com base em uma tabela de referência fornecida pelo site oficial do *framework* SentenceTransformers, que contém informações sobre diversos modelos e suas performances, foi realizado o processo de seleção do modelo "paraphrase-multilingual-MiniLM-L12-v2". Essa escolha foi fundamentada no menor consumo de memória e na maior eficiência em termos de inferência, mesmo que esse modelo apresente um desempenho inferior na busca semântica quando comparado com outros modelos.

Nome do modelo	Desempenho de Busca Semântica[a]	Velocidade[b]	Tamanho
paraphrase-multilingual-mpnet-base-v2	41.68	2500	970 MB
paraphrase-multilingual-MiniLM-L12-v2	39.19	7500	420 MB
distiluse-base-multilingual-cased-v1	29.87	4000	480 MB
distiluse-base-multilingual-cased-v2	27.35	4000	480 MB

Tabela 4.1: Modelos pré-treinados
[a] Desempenho em 6 tarefas diversas para busca semântica. Quanto maior, melhor.
[b] Velocidade de *encoding* (frases / segundos) em uma NVIDIA V100. Quanto maior, melhor.

No processo de cálculo da similaridade entre a pergunta do usuário e as perguntas armazenadas no banco de dados, as perguntas são processadas pelo modelo e transformadas em vetores que representam o seu conteúdo. Dado os vetores, é utilizado a similaridade de cossenos, que é uma medida utilizada para calcular o grau de semelhança entre dois vetores.

No processo de cálculo da similaridade entre a pergunta do usuário e as perguntas armazenadas no banco de dados, as perguntas são processadas pelo modelo e transformadas em vetores que representam o seu conteúdo. Dado os vetores, a similaridade de cossenos é utilizada para calcular o grau de semelhança entre eles.

O resultado desse cálculo varia entre -1 e 1. Um valor próximo de 1 indica alta similaridade entre as perguntas. Por outro lado, um valor próximo de 0 indica que as perguntas são independentes e não possuem relação direta. Um valor de -1 indica que as perguntas possuem sentidos opostos ou conteúdos totalmente diferentes.

4.2.3 Bot para Discord

Para viabilizar a interação dos usuários finais com o sistema e possibilitar testes do modelo, foi desenvolvido um *bot* para a plataforma Discord. Esse *bot* foi criado como um serviço que se assemelha a um chatbot, permitindo a interação direta com os usuários e atuando como interface com a API do modelo de rede neural. Para esse propósito, foram estabelecidos os seguintes requisitos:

- Ser capaz de listar os tópicos disponíveis e suas respectivas categorias
- Ser capaz de responder a perguntas com base em um tópico e categoria selecionados pelo usuário
- Ser capaz de fornecer respostas automáticas, mesmo quando a pergunta não foi diretamente dirigida ao *bot*
- Ser capaz de coletar feedback dos usuários, incluindo avaliações positivas, negativas e comentários em formato de texto

4.2.4 Banco de Dados

Conforme a figura 4.1, as perguntas nesse sistema seguem uma hierarquia, na qual os tópicos são subdivididos em categorias, as quais contêm perguntas e suas respostas correspondentes. É importante ressaltar que o banco foi projetado de tal maneira que é possível que diferentes perguntas façam referência à mesma pergunta, apesar de o sistema administrativo não permitir explicitamente essa operação.

Além disso, os usuários do sistema administrativo têm a capacidade de adicionar tópicos ou categorias como favoritos, personalizando sua experiência. Todas as perguntas feitas pelos usuários das plataformas/serviços são armazenadas na tabela de histórico, juntamente com a referência à plataforma utilizada para fazer a pergunta, o score obtido e a pergunta associada.

É relevante mencionar que as perguntas desconhecidas, ou seja, aquelas com baixo score, também são armazenadas no banco de dados. Isso permite que sejam registradas e analisadas posteriormente, caso necessário.

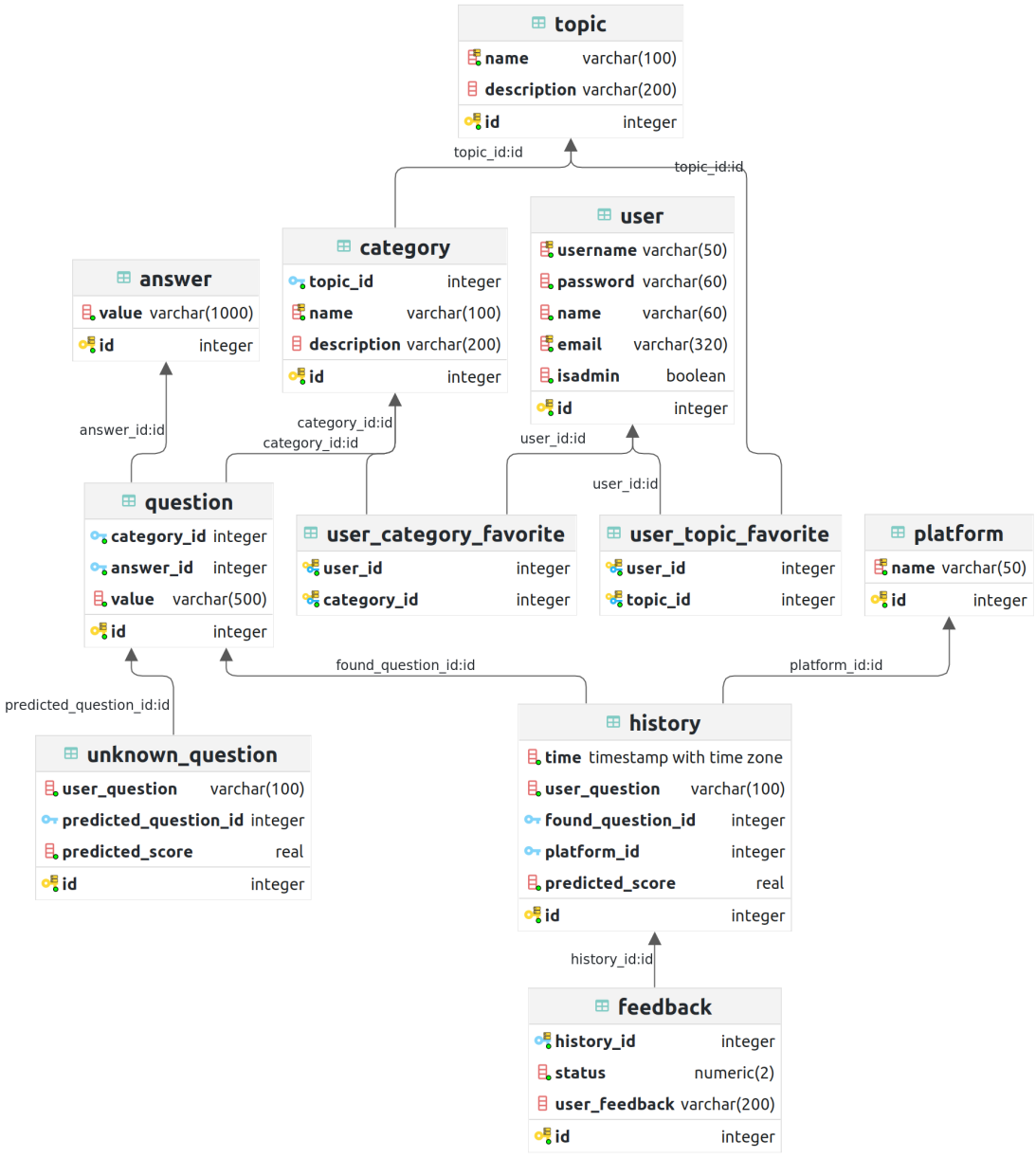

Figura 4.1: Tabelas do banco de dados

Os usuários dos diferentes serviços/plataformas têm a opção de fornecer feedbacks em relação às respostas retornadas pelos serviços. Esses feedbacks podem ser positivos, negativos ou expressos em formato de texto. Essa interação proporciona informações valiosas sobre a qualidade das respostas e ajuda a melhorar o sistema de maneira contínua.

Todas essas informações coletadas permitem obter estatísticas sobre o comporta-

mento dos usuários e contribuem significativamente para o desenvolvimento contínuo do sistema.

As tabelas do banco de dados desempenham um papel crucial na organização e análise desses dados, proporcionando *insights* importantes para aprimorar a experiência dos usuários e a eficiência do sistema como um todo.

4.3 Resultados

Nesta parte do trabalho, apresentaremos os principais resultados obtidos ao término de todo o desenvolvimento. Vamos exibir algumas das principais telas do sistema administrativo, o funcionamento do *bot* e outros resultados. É importante ressaltar que não obtivemos resultados significativos da utilização do *bot* pelo público em geral, uma vez que eles pouco o utilizaram.

4.3.1 Sistema Administrativo

A figura 4.2 mostra a página inicial do sistema. Na parte da esquerda, temos um menu lateral onde pode-se acessar os usuários cadastrados no sistema administrativo, é possível também criar novos usuários além de poder visualizar as perguntas que não foram respondidas, as estatísticas e o histórico de perguntas feitas pelos usuários.

No cabeçalho do site, temos um botão para mudar entre os temas claro e escuro, uma mensagem com o nome do usuário e um botão de menu suspenso onde é possível editar o perfil, baixar o relatório com todos os tópicos, categorias, perguntas e respostas e por fim um botão para fazer logout, como é possivel observar na figura 4.3.

Na parte central do site, temos os tópicos existentes onde podemos editar, apagar ou favoritar através do menu suspenso representado pelos três pontos, conforme representado na figura 4.4. Em cima da lista de tópicos, temos três abas, a primeira sendo todos os tópicos existentes, a segunda os tópicos favoritos do usuário e a terceira é para criar um novo tópico.

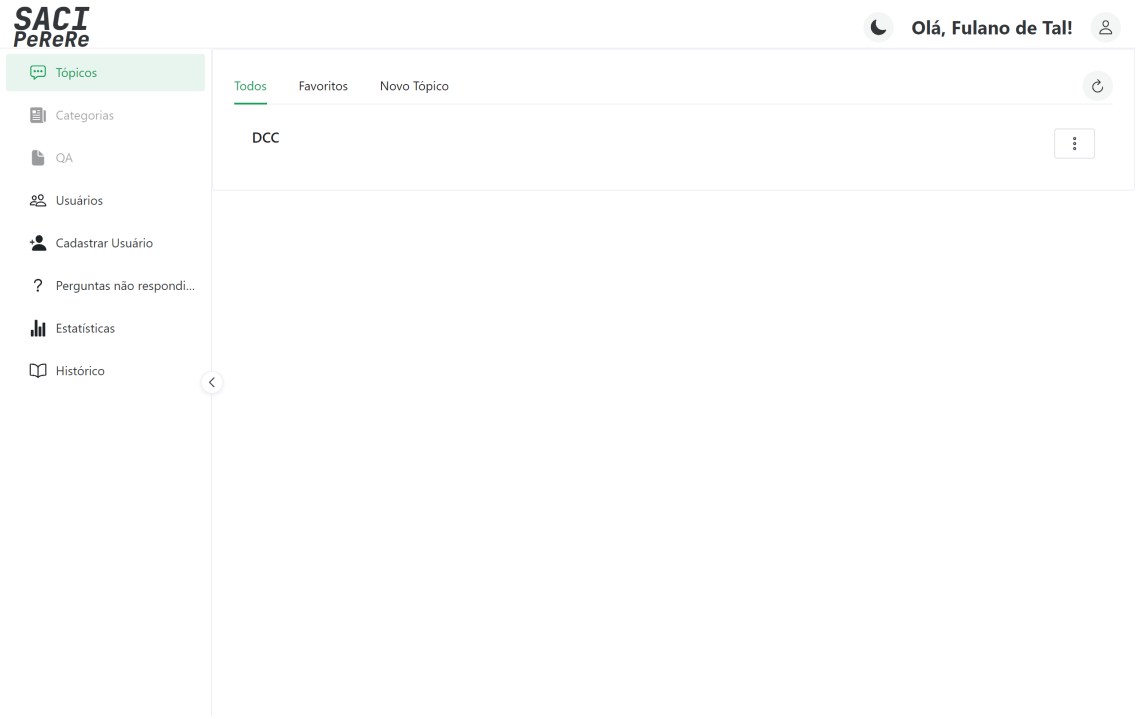

Figura 4.2: Página inicial do sistema administrativo

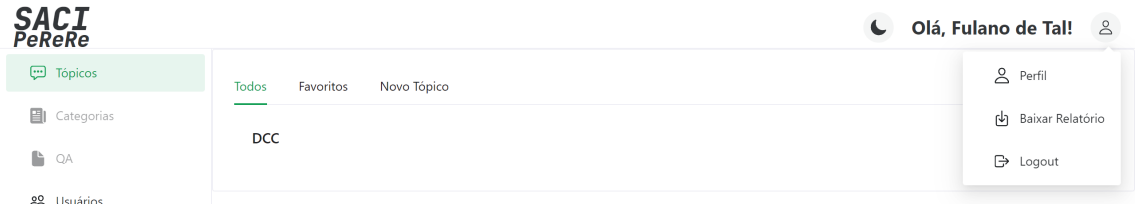

Figura 4.3: Menu suspenso do botão no cabeçalho do site

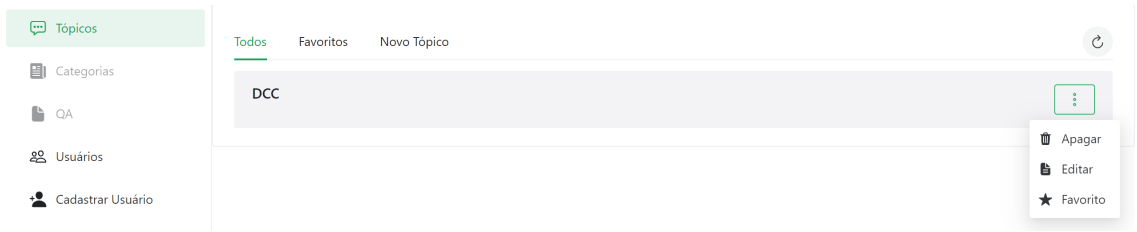

Figura 4.4: Menu suspenso do tópico

Ao selecionar um tópico, a parte central do site é praticamente a mesma para as categorias, o que muda é no menu lateral indicando que você está em uma categoria. Já ao selecionar uma categoria, a parte central das perguntas e respostas sofre uma leve alteração como é possível vê na figura 4.5. Agora, as três abas anteriores são

substituídas pelo nome da categoria, acompanhado por um botão para adicionar novas perguntas e respostas e o menu suspenso representado pelos três pontos tem apenas as opções de apagar ou editar as perguntas e respostas.

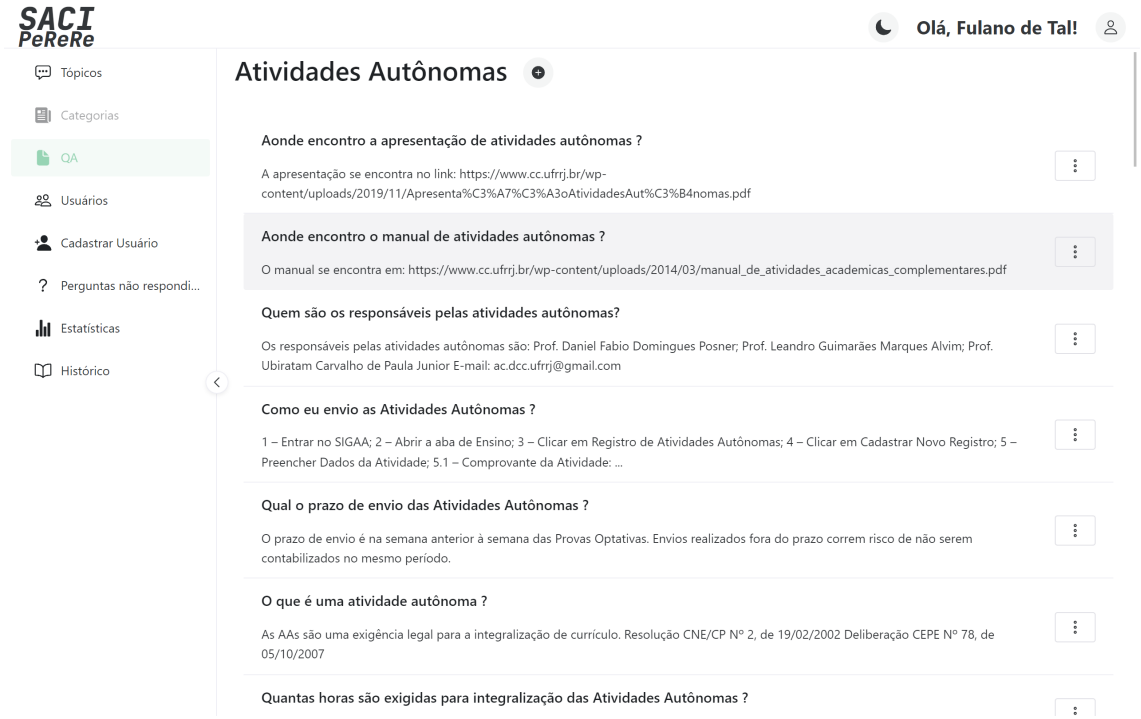

Figura 4.5: Página de perguntas e respostas da categoria Atividades Autônomas

A figura 4.6 retrata a página de estatísticas, na qual você pode visualizar informações relevantes, como o número total de perguntas feitas, o número de *feedbacks* fornecidos pelos usuários e a distribuição desses *feedbacks* em positivos e negativos. Além disso, são exibidos os *feedbacks* textuais fornecidos pelos usuários.

Para uma melhor visualização, é incluído um gráfico que representa de forma clara e intuitiva a quantidade de *feedbacks* positivos e negativos recebidos. Esse gráfico fornece uma visão geral imediata do sentimento geral dos usuários em relação ao conteúdo apresentado pelo *bot* em relação a resposta esperada pelo usuário.

4.3.2 Bot para Discord

A figura 4.7 ilustra a interação entre o *bot* e um usuário quando este faz uma pergunta em um canal no Discord. Após receber a pergunta, o *bot* fornecerá uma

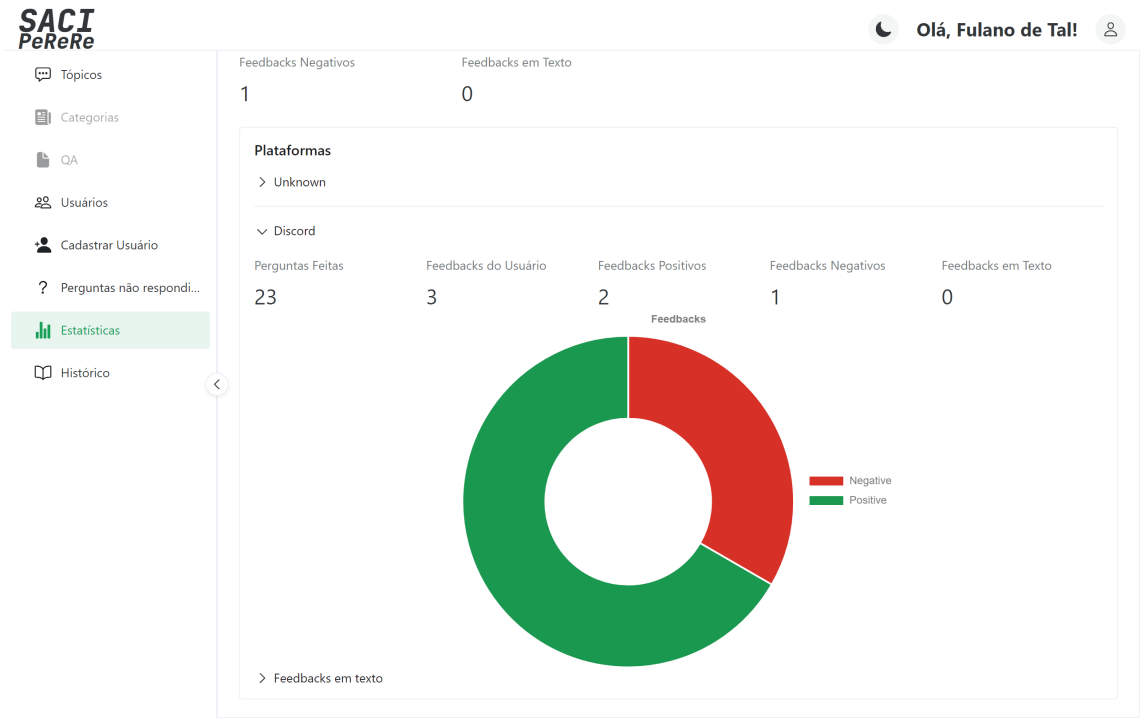

Figura 4.6: Página de estatísticas

resposta ao usuário. Nessa resposta, serão apresentados três botões de *feedback* para que o usuário possa clicar e enviar sua avaliação sobre a adequação da resposta fornecida pelo *bot*.

Esses botões de *feedback* permitem que o usuário expresse sua opinião de forma rápida e fácil. Eles podem indicar se a resposta foi útil ou não em relação à pergunta original, além de poder dar um *feedback* textual. Esses *feedbacks* serão valiosos para melhorar o desempenho e a precisão do *bot*, aprimorando sua capacidade de fornecer respostas relevantes e satisfatórias.

4.3.3 Outros Resultados

O código 4.1 ilustra um exemplo do relatório gerado pelo site administrativo. Esse relatório é gerado no formato JSON e baixado diretamente para a máquina do usuário. Esse arquivo pode ser útil para importar os dados em outro sistema, permitindo a integração e a utilização das informações de forma mais ampla e versátil.

A figura 4.8 ilustra um exemplo de fluxo de integração e entrega contínua no

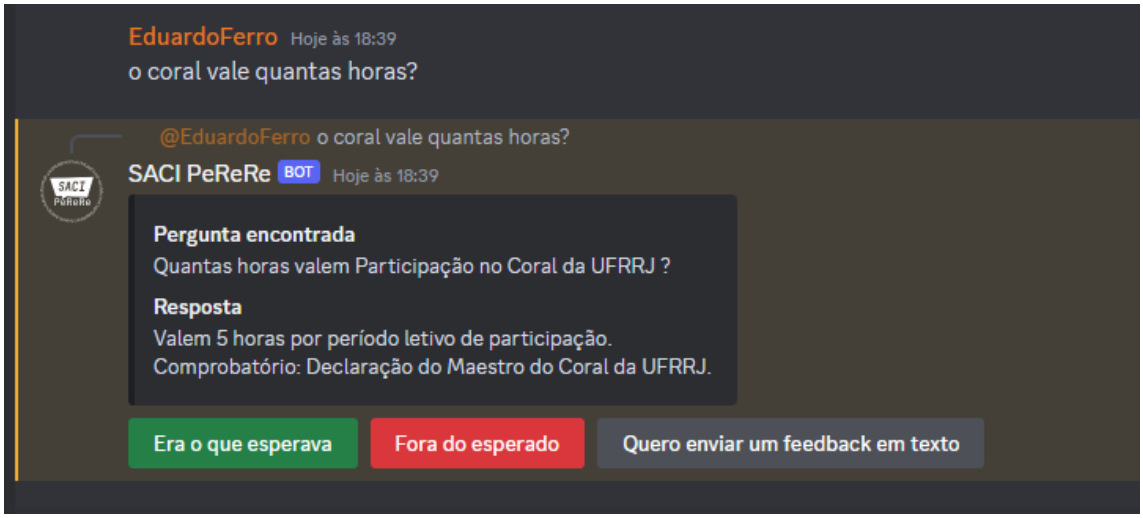

Figura 4.7: Exemplo de resposta do *bot* no Discord

GitHub. Esse fluxo é configurado para realizar automaticamente a execução de testes a cada vez que uma alteração é enviada para o repositório como mencionado anteriormente.

Figura 4.8: Pipeline do GitHub Actions

Código 4.1: Relatório em JSON

```
1 [
2     {
3         "topic_id": 1,
4         "topic_name": "DCC",
5         "topic_description": null,
6         "categories": [
```

```
            {
                "category_id": 1,
                "category_description": null,
                "category_name": "...",
                "topic_id": 1,
                "questions": [
                    {
                        "question_id": 8,
                        "question": "...",
                        "category_id": 1,
                        "answers": [
                            {
                                "answer_id": 8,
                                "answer": "..."
                            }
                        ]
                    }
                ]
            }
        ]
    }
]
```

Capítulo 5

Conclusão

Neste capítulo, serão abordadas as conclusões derivadas do desenvolvimento completo do sistema de chatbot. Será apresentado um resumo dos resultados obtidos, as principais contribuições do trabalho e as considerações finais sobre as limitações encontradas durante a implementação do projeto. Além disso, serão discutidas as oportunidades para futuros trabalhos, com o propósito de aprimorar ainda mais o sistema e explorar novas funcionalidades.

5.1 Considerações Finais

Considerando a necessidade de agilizar o acesso a informações e aliviar a carga de trabalho dos professores e funcionários universitários, como mencionado anteriormente, este trabalho desempenha um papel fundamental como uma ferramenta auxiliar que visa contribuir positivamente para a rotina desses profissionais.

Portanto, a implementação deste sistema de chatbot alcançou seu objetivo de oferecer suporte para a redução da sobrecarga de perguntas repetitivas. Ao disponibilizar respostas automatizadas e instantâneas para as dúvidas mais comuns, permitiu que os servidores públicos das universidades dediquem mais tempo a atividades acadêmicas e administrativas essenciais.

Essa ferramenta será um complemento valioso na vida desses profissionais, pro-

porcionando um ambiente acadêmico mais ágil, eficiente e produtivo. Ao simplificar o processo de busca por informações, o chatbot torna-se um aliado para melhorar a experiência dos usuários, otimizando o fluxo de trabalho e permitindo um atendimento mais rápido e preciso.

5.2 Resumo dos Resultados

Foi desenvolvido um sistema administrativo que desempenha um papel essencial no controle das perguntas e respostas. Além disso, possibilita a inclusão de novos usuários no sistema administrativo e apresenta uma visão abrangente do histórico e das estatísticas das respostas fornecidas pelo modelo, com base na avaliação do usuário final. O sistema também tem a capacidade de mostrar as perguntas que o modelo não conseguiu responder ou não encontrou na base de dados.

Na página de estatísticas, são apresentadas informações relevantes, como o total de perguntas feitas, *feedbacks* dos usuários e sua distribuição em positivos e negativos. Ademais, são apresentados os *feedbacks* textuais dos usuários. Um gráfico claro e intuitivo exibe a quantidade de *feedbacks* positivos e negativos, proporcionando uma visão imediata da satisfação geral dos usuários em relação ao conteúdo do *bot* em relação às respostas esperadas.

Para integrar com a base de dados e o modelo, um *bot* para Discord foi criado, permitindo que usuários façam perguntas e recebam respostas automaticamente. Após receber a resposta, o usuário tem três botões de *feedback* disponíveis para avaliar a adequação da resposta. Esses botões permitem uma avaliação rápida e fácil, permitindo que o usuário indique se a resposta foi útil ou não em relação à pergunta original, além de fornecer um *feedback* textual. Essas avaliações são valiosas para aprimorar o desempenho e a precisão do *bot*, tornando-o capaz de oferecer respostas mais relevantes e satisfatórias.

5.3 Principais Contribuições

- *Uso de inteligência artificial para responder FAQs em português brasileiro*
  Este trabalho contribuiu como material de estudo sobre a utilização de inteligência artificial (IA) na resposta a perguntas frequentes em português brasileiro. Mais especificamente, foi explorado o uso de similaridade semântica para comparar perguntas feitas em linguagem natural com as perguntas cadastradas em uma base de dados, com o objetivo de identificar a pergunta mais semelhante e, consequentemente, fornecer a resposta correspondente de forma precisa e eficiente.

- *Desenvolvimento de um sistema administrativo Web*
  Uma outra contribuição importante deste trabalho está relacionada ao desenvolvimento de um sistema administrativo Web, que serve como referência para projetos futuros. Neste estudo, são apresentadas soluções que desempenham bem, enquanto outras podem servir como aprendizados para aprimoramentos futuros.

5.4 Limitações e Trabalhos Futuros

Durante o desenvolvimento deste trabalho, ficou evidente a sua magnitude, o que nos levou a identificar algumas limitações e áreas que podem ser aprimoradas em trabalhos futuros. A partir da nossa perspectiva, uma abordagem dividida em duas partes pode contribuir para um desenvolvimento mais completo e eficiente da ferramenta. Essas duas partes consistiram em uma pesquisa aprofundada das principais tecnologias utilizadas no desenvolvimento de chatbots, seguida pelo foco exclusivo no desenvolvimento do sistema Web.

A primeira sugestão para um trabalho futuro envolveria uma investigação detalhada das tecnologias e abordagens mais recentes no campo de chatbots. Isso incluiria uma análise de diferentes *frameworks*, bibliotecas e algoritmos disponíveis, bem como estudos comparativos sobre a eficácia e escalabilidade dessas tecnologias. Essa pesquisa forneceria uma base sólida para a seleção das melhores ferramentas e

estratégias a serem adotadas no desenvolvimento do chatbot.

A nossa segunda sugestão seria dedicada ao desenvolvimento do sistema Web em si. Com base nas descobertas da pesquisa anterior, seria possível realizar uma implementação mais refinada, considerando as melhores práticas e as necessidades específicas da universidade. Isso poderia incluir a criação de uma interface intuitiva e amigável, além do aprimoramento do modelo de redes neurais para garantir respostas mais precisas e eficientes.

Ao dividir o trabalho em duas partes distintas, seria possível aprofundar a pesquisa e se concentrar em cada etapa do desenvolvimento do chatbot. Dessa forma, o resultado final seria uma ferramenta mais robusta, baseada em tecnologias atualizadas e adaptada às necessidades específicas da universidade.

No entanto, é importante ressaltar que essa abordagem é apenas uma sugestão e que outras oportunidades de melhoria e refinamento podem surgir ao longo do processo. O objetivo final é desenvolver um sistema de chatbot que realmente cumpra seu propósito de agilizar o acesso a informações e auxiliar os profissionais da universidade de forma eficaz.

Além das sugestões e áreas para melhorias mencionadas anteriormente, existem algumas limitações específicas identificadas durante o desenvolvimento deste trabalho. Uma dessas limitações está relacionada ao modelo utilizado para o chatbot. Embora o modelo empregado tenha mostrado um desempenho satisfatório, é importante reconhecer que existem outros modelos disponíveis que apresentam um desempenho ainda melhor. Um exemplo é a pesquisa em andamento Rodrigues, Tanti e Agerri (2023)[1], que realizou um fine-tunning em modelos amplamente reconhecidos e obteve resultados promissores em textos escritos em português brasileiro. Considerar a adoção de um modelo mais avançado poderia potencialmente melhorar a precisão e a eficácia das respostas do chatbot.

Outra limitação identificada está relacionada ao *framework* utilizado para a API do sistema administrativo. Atualmente, o Express foi utilizado, porém, considerar a

[1] https://github.com/ruanchaves/eplm

utilização de um *framework* mais robusto e opinado, como o NestJS[2] ou Laravel[3], poderia trazer benefícios significativos para a manutenção do código. Ao adotar um *framework* mais robusto, seria possível aprimorar a estrutura do sistema e facilitar a implementação de novas funcionalidades no futuro.

Uma limitação importante identificada no sistema atual é a restrição do sistema de permissões. No momento, o sistema de permissões é bastante limitado e só tem utilidade para bloquear usuários não administradores de manipular os dados cadastrados. Melhorar o sistema de permissões, permitindo uma configuração mais personalizada, seria essencial para garantir o controle das ações de cada usuário.

Por fim, outra limitação identificada diz respeito à API que executa o modelo do chatbot. Atualmente, a API pode enfrentar problemas de desempenho quando há um grande número de usuários simultâneos. Para lidar com essa limitação, seria necessário aprimorar a API, considerando a implementação de algum tipo de fila de execução. Uma fila de execução permitiria o processamento ordenado das requisições, garantindo que o sistema não seja sobrecarregado quando vários usuários estiverem interagindo com o chatbot ao mesmo tempo. Isso resultaria em uma experiência mais estável e evitaria possíveis problemas de desempenho.

[2]https://nestjs.com/
[3]https://laravel.com/

Referências

ALLAM, A.; HAGGAG, M. The question answering systems: A survey. *International Journal of Research and Reviews in Information Sciences*, v. 2, p. 211–221, 09 2012.

ALVES, T. C. P. O uso de chatbots para responder dúvidas frequentes de um campus universitário: O caso do cefet/rj. In: . [S.l.: s.n.], 2021.

BERNERS-LEE, T. The world wide web-past, present and future. 1997.

BIRD, S.; KLEIN, E.; LOPER, E. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly Media, 2009. ISBN 9780596555719. Disponível em: <https://books.google.com.br/books?id=KGIbfiiP1i4C>.

BROMLEY, J. et al. Signature verification using a"siamese"time delay neural network. *Advances in neural information processing systems*, v. 6, 1993.

CHOPRA, S.; HADSELL, R.; LECUN, Y. Learning a similarity metric discriminatively, with application to face verification. In: IEEE. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. [S.l.], 2005. v. 1, p. 539–546.

DEVLIN, J. et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019.

HARABAGIU, S. M.; PAsCA, M. A.; MAIORANO, S. J. Experiments with open-domain textual question answering. In: *Proceedings of the 18th Conference on Computational Linguistics - Volume 1*. USA: Association for Computational Linguistics, 2000. (COLING '00), p. 292–298. ISBN 155860717X. Disponível em: <https://doi.org/10.3115/990820.990863>.

HAYKIN, S. *Redes Neurais: Princípios e Prática*. Bookman Editora, 2001. ISBN 9788577800865. Disponível em: <https://books.google.com.br/books?id=bhMwDwAAQBAJ>.

IBGE - Instituto Brasileiro de Geografia e Estatística. *Censo da Educação Superior 2021*. Rio de Janeiro: IBGE, 2022.

JÚNIOR, S. R. J. d. S.; BARBOSA, Y. d. A. M. Um chatbot para responder faqs. *Trabalho de conclusão de curso apresentado ao curso de bacharelado em sistemas de informação*, Universidade Federal da Paraíba, 2018.

KEMP, S. *Digital 2023: Global Overview Report*. 2023. <https://datareportal.com/reports/digital-2023-global-overview-report>. Acesso em: 08 jul. 2023.

KOWSARI, K. et al. Text classification algorithms: A survey. *Information*, v. 10, n. 4, 2019. ISSN 2078-2489. Disponível em: <https://www.mdpi.com/2078-2489/10/4/150>.

KUROSE, J.; ROSS, K. *Computer Networking: A Top-down Approach*. Pearson, 2013. (Always learning). ISBN 9780132856201. Disponível em: <https://books.google.com.br/books?id=fWg5pwAACAAJ>.

LE, Q. V.; MIKOLOV, T. *Distributed Representations of Sentences and Documents*. 2014.

LEE, K. et al. Can chatbots help reduce the workload of administrative officers? - implementing and deploying faq chatbot service in a university. In: STEPHANIDIS, C. (Ed.). *HCI International 2019 - Posters*. Cham: Springer International Publishing, 2019. p. 348–354. ISBN 978-3-030-23522-2.

LIN, T. et al. A survey of transformers. *CoRR*, abs/2106.04554, 2021. Disponível em: <https://arxiv.org/abs/2106.04554>.

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

MIKOLOV, T. et al. *Efficient Estimation of Word Representations in Vector Space*. 2013.

MIKOLOV, T. et al. *Distributed Representations of Words and Phrases and their Compositionality*. 2013.

MITCHELL, T. *Machine Learning*. McGraw-Hill Education, 1997. (McGraw-Hill international editions - computer science series). ISBN 9780070428072. Disponível em: <https://books.google.com.br/books?id=xOGAngEACAAJ>.

NECULOIU, P.; VERSTEEGH, M.; ROTARU, M. Learning text similarity with siamese recurrent networks. In: *Proceedings of the 1st Workshop on Representation Learning for NLP*. [S.l.: s.n.], 2016. p. 148–157.

PENNINGTON, J.; SOCHER, R.; MANNING, C. GloVe: Global vectors for word representation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, 2014. p. 1532–1543. Disponível em: <https://aclanthology.org/D14-1162>.

RADFORD, A. et al. Language models are unsupervised multitask learners. 2019.

REIMERS, N.; GUREVYCH, I. *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. 2019.

RODRIGUES, R. C.; TANTI, M.; AGERRI, R. *Evaluation of Portuguese Language Models*. 2023. Disponível em: <https://github.com/ruanchaves/eplm>.

RUSHKOFF, D. *Present Shock: When everything happens now*. Nova York: Current, 2013.

RUSSELL, S.; NORVIG, P.; DAVIS, E. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2010. (Prentice Hall series in artificial intelligence). ISBN 9780136042594. Disponível em: <https://books.google.com.br/books?id=8jZBksh-bUMC>.

VASWANI, A. et al. Attention is all you need. *CoRR*, abs/1706.03762, 2017. Disponível em: <http://arxiv.org/abs/1706.03762>.

ZHANG, W.; YOSHIDA, T.; TANG, X. A comparative study of tf*idf, lsi and multi-words for text classification. *Expert Systems with Applications*, v. 38, n. 3, p. 2758–2765, 2011. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417410008626>.