CAIO NOBORU ASAI
RICARDO KEN WANG TSUZUKI

DESENVOLVIMENTO DE UM *CHATBOT* INFORMACIONAL SOBRE A COVID-19

São Paulo, São Paulo
2020

CAIO NOBORU ASAI
RICARDO KEN WANG TSUZUKI

DESENVOLVIMENTO DE UM *CHATBOT* INFORMACIONAL SOBRE A COVID-19

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

São Paulo, São Paulo
2020

CAIO NOBORU ASAI
RICARDO KEN WANG TSUZUKI

DESENVOLVIMENTO DE UM *CHATBOT* INFORMACIONAL SOBRE A COVID-19

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

Área de Concentração:
Inteligência Artificial

Orientador:
Fábio Gagliardi Cozman

São Paulo, São Paulo
2020

RESUMO

Este trabalho tem como objetivo implementar um *chatbot* que informe o usuário sobre a COVID-19. O *Watson Assistant* é utilizado como interface com o usuário e suas ferramentas de *Natural Language Processing*, o *scikit-learn* fornece os métodos de aprendizado de máquina para verificar a validade das informações e o *covid-daily* é utilizado para obter informações da COVID-19.

Palavras-Chave – *Chatbot*, *Watson Assistant*, *agente conversacional*, *fake news*.

ABSTRACT

This work aims to implement a *chatbot* that informs the user about COVID-19. Watson Assistant is used for its user interface and Natural Language Processing tools, scikit-learn provides machine learning methods to verify validity of information and covid-daily is used to obtain COVID-19 information.

Keywords – *Chatbot*, *Watson Assistant*, *Conversational agent*, *fake news*.

SUMÁRIO

PARTE I

INTRODUÇÃO

1 INTRODUÇÃO

1.1 Motivação

Agentes conversacionais são sistemas de diálogo entre usuário e computador que recebem informação e respondem no formato de linguagem humana. Com o advento da digitalização, o aumento da comunicação em forma de texto e a evolução do poder computacional, os agentes conversacionais, também conhecidos como *chatbots*, se destacaram como uma ferramenta de automatização no atendimento ao consumidor.

O interesse em inteligência artificial, que engloba agentes conversacionais, surgiu na década de 1950. Com a evolução do *machine learning*, novas arquiteturas surgiram e proporcionaram outros tipos de aplicações.

Chatbots são muito utilizados para geração de receita, aumento da produtividade e eficiência através da substituição do trabalho humano. Eles podem ser usados para a difusão de informações provenientes de fontes confiáveis, como assistente pessoal e em tratamentos médicos como por exemplo no auxílio no tratamento da depressão e sentimento de solidão comuns em idosos. Os *chatbots* proporcionam integração com várias APIs possibilitando a construção de sistemas cada vez mais inteligentes e abrangentes, capazes de interagir com o ser humano em sua própria linguagem.

O *Facebook Messenger* possui exemplos de sucesso na implementação de *chatbots* como soluções. Esses casos podem ser conferidos através de sua *webpage* oficial [1]. Alguns números ajudam a expressar o impacto desse tema. Globalmente existem cerca de 2,5 bilhões de usuários ativos mensais no *Facebook*. Diariamente o número é de 1,66 bilhões de usuários. Dentre todos os usuários do *Facebook*, 88% acessa o serviço através do celular. Hoje existem aproximadamente 2,53 bilhões de *smartphones* em uso no mundo. Além disso, 95,8% dos meios de comunicação social comerciantes estão usando a plataforma de anúncios do *Facebook*.

Do ponto de vista comercial pode-se citar vantagens como:

- **Acessibilidade:** Os *chatbots* são fáceis de acessar. O consumidor pode abrir o site e começar a fazer perguntas ou resolver suas consultas sem precisar discar um número e seguir incômodas etapas na "URA". Ele pode chegar rapidamente no objetivo a partir de um conjunto básico de informações.
- **Eficiência:** Os clientes podem sentar-se em sua cadeira no escritório ou em uma poltrona enquanto jogam um game e veem seus status em uma aplicação de cartão de crédito, pesquisam em que nível está seus pedidos de comida ou fazem uma reclamação sobre algum problema. Os bots ajudam a melhorar o negócio.
- **Disponibilidade:** Os *chatbots* ficam disponíveis 24 horas por dia, 7 dias por semana. Executam as mesmas tarefas ou novos serviços sempre com a mesma eficiência e desempenho.
- **Escalabilidade:** Um bot pode substituir vários funcionários. Ele consegue manipular várias consultas ao mesmo tempo sem se cansar. Não é necessário que os clientes esperem na fila até um atendente estar disponível.
- **Custo:** *Chatbots* são econômicos por substituírem o trabalho de vários funcionários, pela sua eficiência e baixa taxa de manutenção.
- ***Insights*:** Os bots podem usar técnicas mais recentes de *machine learning* e *data science* para fornecer *insights* que geralmente consultores não conseguem dar.

As Figuras 1 e 2 mostram a popularidade do uso dos *chabots* nos últimos 5 anos. Os números do eixo Y representam o interesse de pesquisa relativo ao ponto mais alto no gráfico de uma determinada região em um dado período. Um valor de 100 representa o pico de popularidade de um termo. Um valor de 50 significa que o termo teve metade da popularidade. Uma pontuação de 0 significa que não havia dados suficientes sobre o termo. A Figura 3 mostra o interesse por Estado da federação.

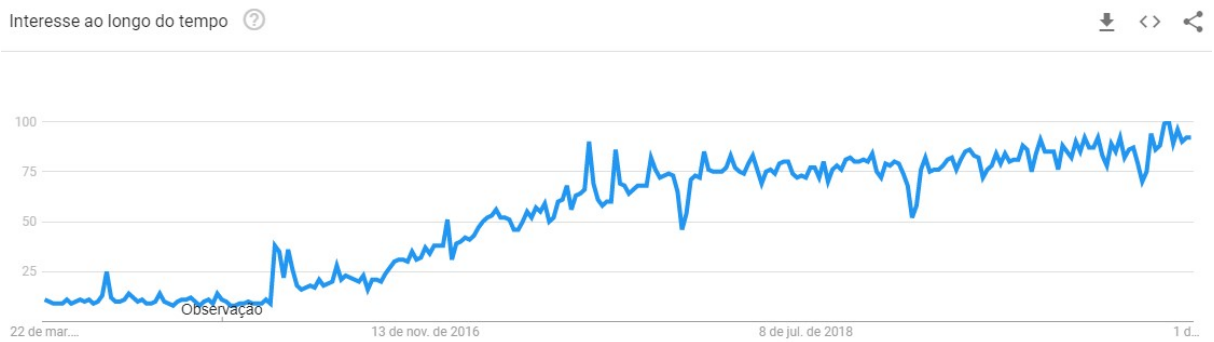

Figura 1: Popularização dos *chatbots* no mundo.
Fonte: https://trends.google.com/trends/

O pico mundial ocorreu entre os dias 26/01/2020 e 08/02/2020.

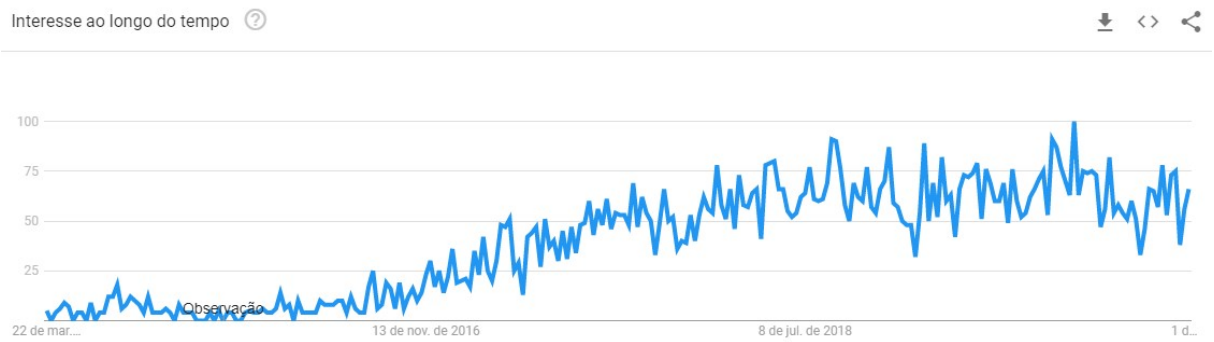

Figura 2: Popularização dos *chatbots* no Brasil.
Fonte: https://trends.google.com/trends/

O pico brasileiro ocorreu entre os dias 08/09/2019 e 14/09/2019.

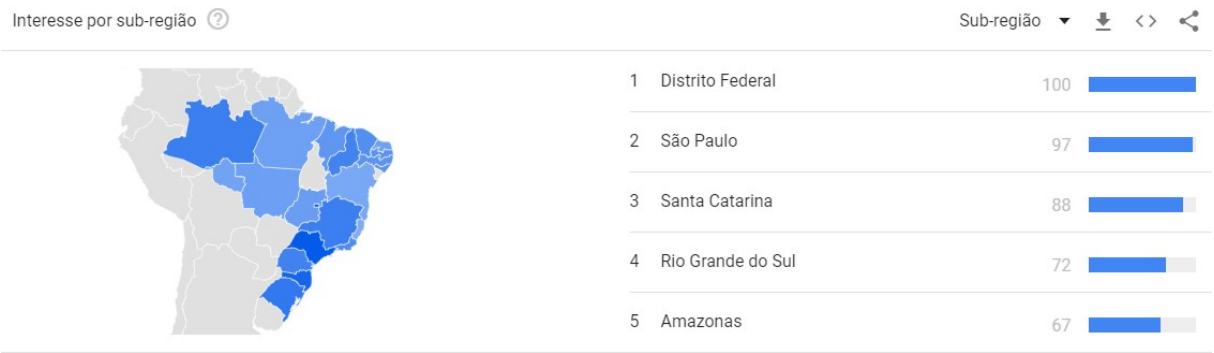

Figura 3: Popularização dos *chatbots* nos Estados da federação.
Fonte: https://trends.google.com/trends/

1.2 Contexto Histórico de *Chatbots*

A evolução histórica dos *chatbots* é descrita resumidamente nos seguintes momentos importantes. Essas informações foram compiladas e publicadas no livro "*Construindo Chatbots com Python: Usando Natural Language Processing e Machine Learning*" [2] e estão reproduzidas abaixo.

- **1950:** O teste de Turing foi desenvolvido por Alan Turing. Ele testava a habilidade de uma máquina exibir comportamento inteligente equivalente ao, ou indistinguível do, de um humano.
- **1966:** Eliza, o primeiro *chatbot*, foi criada por Joseph Weizenbaum e projetada para ser uma terapeuta. Ela simulava uma conversa usando uma metodologia de

"comparação de padrões"e substituição que dava aos usuários a impressão de compreensão por parte do bot.

- **1972:** Parry, um programa de computador do psiquiatra e cientista Kenneth Colby, modelava o comportamento de esquizofrenia paranoide.

- **1981:** O *chatbot* Jabberwocky foi criado pelo programador britânico Rollo Carpenter e lançado na internet em 1997. O objetivo do *chatbot* era "simular uma conversa humana natural de maneira interessante, agradável e bem-humorada".

- **1985:** O robô de brinquedo *wireless*, Tomy Chatbot, repete mensagens gravadas em sua fita.

- **1992:** Dr. Sbaitso, um *chatbot* criado pela *Creative Labs*, "conversava"com o usuário como se fosse um psicólogo em voz digitalizada.

- **1995:** A.L.I.C.E. (*Artificial Linguistic Internet Computer Entity*) foi desenvolvida pelo ganhador do Prêmio Nobel Richard Wallace.

- **1996:** Hex, desenvolvido por Jason Hutchens, foi baseado na Eliza e ganhou o Prêmio Loebner em 1996.

- **2001:** *Smarterchild*, um *bot* inteligente desenvolvido pela *ActiveBuddy*, foi amplamente distribuído em trocas de mensagens instantâneas globais e redes SMS. A implementação original evoluiu rapidamente para dar acesso instantâneo a notícias, previsão do tempo, informações do mercado de ações, horários de sessões de cinema, listagens de páginas amarelas e dados detalhados de esportes, além de fornecer várias ferramentas (assistente pessoal, calculadoras, tradutor etc.).

- **2006:** A ideia do *Watson* foi concebida em um jantar; ele estava sendo projetado para competir no programa de TV "*Jeopaydy*". Em sua primeira tentativa, só acertou cerca de 15% das perguntas, mas depois conseguiu vencer concorrentes humanos regularmente.

- **2010:** A *Siri*, uma assistente pessoal inteligente, foi lançada como um aplicativo do *iPhone* e depois integrada como parte do iOS. Ela é resultado de um esforço do Centro de Inteligência Artificial da *SRI International*. Seu mecanismo de reconhecimento de fala foi fornecido pela *Nuance Communications*; a *Siri* usa tecnologias avançadas de *machine learning*.

- **2012:** O *Google* lançou o *chatbot Google Now*. Originalmente seu codinome era "Majel"em homenagem a Majel Barrett, esposa de Gene Roddenberry e voz dos sistemas de computador da franquia *Star Trek*; Também recebeu o codinome "*Assistant*".

- **2014:** A *Amazon* lançou a *Alexa*.

- **2015:** *Cortana*, assistente virtual criada pela *Microsoft*. *Cortana* pode criar lembretes, reconhecer voz natural e responder a perguntas usando informações do mecanismo de busca *Bing*. Seu nome vem de uma personagem de inteligência artificial fictícia da série de videogame *Halo*.

- **2016:** Em Abril de 2016. o *Facebook* anunciou uma plataforma de bot para o *Messenger* que incluía APIs de construção de *chatbots* para a interação com os usuários. Melhorias posteriores incluíram *bots* que participam de grupos, telas de pré-visualização e recurso de leitura de QR por intermédio da funcionalidade de câmera do *Messenger* para levar os usuários diretamente para o *bot*.

  Em Maio de 2016, o *Google* revelou seu *bot* ativado por voz concorrente do *Amazon Echo* chamado *Google Home* na conferência de desenvolvedores da empresa.

- **2017:** O *Woebot* é um agente conversacional que nos ajuda a monitorar nosso humor, a nos conhecer melhor e a nos sentir bem. Ele usa uma combinação de técnicas de NLP (*Natural Language Processing*), habilidades psicológicas (terapia cognitivo-comportamental) e senso de humor para tratar a depressão.

1.3 COVID-19

Em Dezembro de 2019 houve uma aglomeração de casos de pneumonia na China. Pesquisas concluíram que a origem tratava-se de um vírus desconhecido, conhecido atualmente como "*2019 Novel Coronavírus*". Coronavirus é um grande grupo de vírus que consiste em um núcleo de material genético envolto por um envelope de picos de proteínas dando uma aparência de coroa, do latim "*corona*". Há diferentes tipos de coronavírus que geram sintomas respiratórios e gastrointestinais.

2 OBJETIVOS

O objetivo do presente trabalho é fornecer uma ferramenta informativa sobre doenças infectocontagiosas. A aplicação se restringe tendo como foco a doença causada pela 2019 *n-Cov*, entretanto pode-se extender o uso para doenças como sarampo e dengue.

A ferramenta é proposta como um *chatbot* conversacional que é capaz de reconhecer as intenções e entidades na fala do usuário, obter a informação solicitada e fornecê-la de forma apropriada. O *Watson Assistant* é a plataforma utilizada para a criação do agente conversacional.

Dados numéricos como número de pessoas infectadas e número de casos que vieram a óbito são atualizados segundo o *https://www.worldometers.info/*, permitindo procurar informações com linguagem natural. Utiliza-se a biblioteca *covid-daily*, a qual realiza o *web scraping* do *Worldometers* para extrair informações de gráficos, tabelas e textos. Os dados são tratados utilizando bibliotecas da linguagem *python* como o *pandas* e *numpy*.

Outra aplicação consiste no combate a *Fake News*, aplicando-se modelos que verificam se uma determinada notícia é verdadeira ou falsa. Para tal, criou-se um modelo de predição baseado em aprendizado supervisionado que foi treinado com um conjunto de notícias específicas sobre a COVID-19. O tratamento e a classificação dessas informações são aplicadas a um algoritmo de *machine learning* no *scikit-learn* para o aprendizado da aplicação. A aplicação desse tipo de sistema é abrangente, podendo ser utilizada para fins educacionais, sociais e econômicos.

PARTE II

ESTADO DA ARTE

3 TERMINOLOGIA E FERRAMENTAS SOBRE *CHATBOTS*

Atualmente os *chatbots* possuem limitações. Geralmente são aplicados em tarefas repetitivas que demandam análise ou busca de dados e que possam ser automatizadas e resolvidas. Existem vários *frameworks* utilizados para construção *chatbots* e seu uso para contrução de *chatbots* é analisado em [3]. Os mais conhecidos são o *Dialogflow* e o *Watson Assistant*. Assim, apresenta-se uma breve introdução dos termos de agentes conversacionais e uma breve comparação entre as características do *Dialogflow* e *Watson Assistant*.

3.1 *Natural Language Processing*

O *Natural Language Processing* (NLP) é uma área da inteligência artifical que permite que os computadores analisem e entendam a linguagem humana. O *Natural Language Understanding* (NLU) expressa a habilidade de uma máquina entender a linguagem natural da forma fornecida pelos humanos. Alguns avanços em NLP são discutidos em [4].

Os desafios de desenvolver agentes conversacionais que se assemelham a humanos segundo [5] são grandes. O trabalho sugere que os agentes podem ser inspirados em diálogos humano-humano, mas não precisam necessariamente imitá-los. Podem haver áreas de aplicação específicas nas quais a conversa pode ser apropriada, se não essencial, entre seres humanos e agentes, particularmente em áreas como assistência médica e bem-estar, onde as nuances de contextos e dados demográficos precisam ser consideradas.

3.2 Intenção

Quando se cria uma experiência de conversa, a primeira tarefa é compreender sobre o que o usuário está falando. Intenção é a maneira de identificar o que o consumidor final quer. Cria-se uma intenção para qualquer coisa que o usuário possa solicitar como um pe-

dido de uma pizza, agendamento de consultas médicas, ou solicitação de uma informação em um banco de dados. Para cada intenção, é preciso providenciar as várias maneiras pelas quais o usuário irá comunicar seu desejo. Chama-se isso de frases de treinamento. Deve-se providenciar também como a aplicação irá responder a essas solicitações.

É possível criar agentes conversacionais que são capazes de compreender e responder a vários idiomas. Conforme as pessoas utilizam o agente, pode-se incorporar o que elas dizem a essas intenções como exemplos de treinamento, como um procedimento de aprendizado.

Para uma dada intenção, podem-se adicionar intenções *follow-Ups* que serão somente acionadas depois de a intenção inicial ter sido ativada. Pode-se usar isso para combinar sequências como Sim/Não numa resposta a uma pergunta específica. Fazendo-as específicas para cada intenção, evita-se combinações erradas de Sim/Não durante a conversa, auxiliando na organização da estrutura do diálogo.

As intenções *Fallback* são acionadas se a entrada do usuário não combina com alguma intenção existente. Pode-se usá-las para guiar o usuário na direção correta do diálogo.

3.3 Entidade

Quando um usuário expressa sua intenção, é comum ele querer que o agente realize ações com as informações especificadas. É importante identificar elementos como datas, horários, nomes e números. Entidades são utilizadas para extrair esse tipo de informação a partir da fala do usuário. Existe uma lista do próprio sistema com algumas entidades comuns como data, número e horário. É possível também criar uma lista própria de palavras ou frases que se assemelhem com um dado conceito como por exemplo um conjunto de siglas, ou acrônimos.

3.4 Controle da Conversa

Conversa é o processo em que duas pessoas negociam significado e compreensão através de sua interação. Há dois tipos de diálogos que é preciso considerar: diálogos lineares e não-lineares.

Diálogos lineares são usados no processo de coletar toda informação necessária para completar uma ação solicitada como agendar uma reunião ou finalizar um pedido.

Imagine que se solicita a um agente para agendar o reparo de uma bicicleta. Antes do agente conseguir ajudar o usuário, é preciso que aquele saiba algumas informações como o tipo do serviço solicitado, data e horário. É possível que o usuário providencie essas informações em uma única declaração. Entretanto, se o usuário colocar apenas algumas informações, é necessário um procedimento de validação.

Diálogo linear funciona somente quando há um conjunto específico de fatos que desejamos coletar. Diálogos não-lineares é algo mais próximo a um diálogo real em que o assunto pode variar dependendo da mudança do contexto da conversa.

Quando um usuário diz algo, isso é interpretado no contexto do que foi dito anteriormente. Suponha que durante a conversa há duas perguntas de Sim/Não que foram feitas. Para compreender qual pergunta está sendo respondida e em que ponto da conversa se está, o *chatbot* usa o conceito de contexto.

3.5 Contexto

Contextos são variáveis utilizadas para armazenar informações com o objetivo de auxiliar na distinção de ambiguidades e tornar o diálogo mais personalizado. Cada plataforma aborda contexto de formas diferentes. Alguns exemplos de contexto são: período da conversa (manhã, tarde, ou noite), nome do usuário e o assunto do diálogo em um determinado instante.

3.6 *Dialogflow*

Segundo a documentação no *Google Cloud Platform*, "O Dialogflow é uma plataforma de processamento de linguagem natural que facilita o *design* e a integração de uma interface do usuário conversacional com *apps* para dispositivos móveis, aplicativos da Web, *bots*, sistemas de unidade de resposta audível etc. Usando o *Dialogflow*, é possível fornecer maneiras novas e atraentes para os usuários interagirem com seu produto. O *Dialogflow* pode analisar vários tipos de entrada de seus clientes, incluindo entradas de texto ou áudio (como de um smartphone ou gravação de voz). Ele também é capaz de responder aos seus clientes de várias maneiras, seja por meio de texto ou com fala sintética" [6]. Por se tratar de um serviço que está há um tempo no mercado, seus algoritmos de processamento de linguagem natural não são os mais atualizados de acordo com a tecnologia atual disponível. Isso gera uma lentidão considerável no tempo de resposta e requisições

via protocolo HTTP.

O *Dialogflow* permite a marcação de entidades através de uma ferramenta chamada de *text-annotation*, conforme as Figuras 4 e 5.

Figura 4: Exemplo de *Text annotation* no *Dialogflow*.

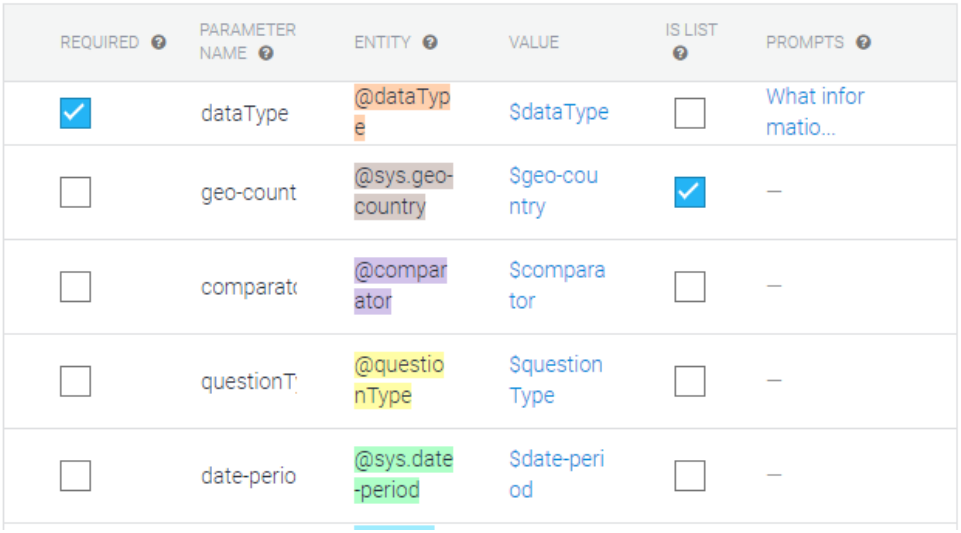

Figura 5: Entidades identificadas com *Text annotation* no *Dialogflow*.

O *Dialogflow* suporta mais de 20 idiomas e todos possuem esse recurso. Além disso, o *Dialogflow* possui três tipos de entidades:

- Entidade do Sistema: abrangem casos de uso comum como números, datas, quantidades com unidade, contatos, nomes e localidades.
- Entidade do Desenvolvedor: permitem definir uma própria entidade baseada numa lista de palavras no Console, na API, ou adicionada através de um arquivo CSV. Cada palavra da lista será reconhecida pela entidade. É possível incorporar sinônimos

se houver várias maneiras de se referir a um determinado tópico. Pode-se ainda criar entidades compostas que combinam várias entidades para descrever conceitos com vários atributos.

- Entidades de Sessão: podem ser definidas por uma seção específica do usuário. Elas permitem relacionar coisas que são transientes como detalhes de uma requisição anterior do usuário, ou uma lista das principais promoções de uma loja perto da sua localização. Essas entidades expiram após 10 minutos de conversa. Elas são criadas programando via API.

O *Dialogflow* também diferencia as entidades *@sys.number* (números cardinais) e *@sys.ordinal* (números ordinais). O serviço da *Google* também providencia uma entidade para localização, denominada *@sys.geo-country*.

Contextos no *Dialogflow* são semelhantes aos contextos de linguagem natural. Para interpretar uma sentença, é necessário conhecer informações anteriores. Contextos são muito utilizados para controlar o fluxo de uma conversa.

No *Dialogflow* existem dois tipos de contextos: entrada e saída. Os contextos de saída controlam contextos que estão ativos, ou seja, quando uma intenção é reconhecida e acionada, todos os contextos de saída correpondentes a ela são ativados. Os contextos de entrada controlam a correspondência da intenção. Enquanto os contextos estiverem ativos, o *Dialogflow* irá acionar intenções relacionadas com os contextos de entrada que pertençam ao conjunto de contextos ativos no momento.

Quando uma intenção com contexto de saída é correspondida, seus valores de parâmetro coletados podem servir como armazenamento temporário com referências de parâmetro para contextos ativos. Esses valores podem ser acessados durante a correspondência futura da intenção, enquanto o contexto permanecer ativo. Cada contexto ativo tem uma duração que define o número de interações em uma conversa para os quais o contexto permanece ativo. No *Dialogflow* os contextos expiram após 20 minutos ativados. Quando um contexto de saída é ativado por meio do acionamento de uma intenção, o tempo de expiração é redefinido.

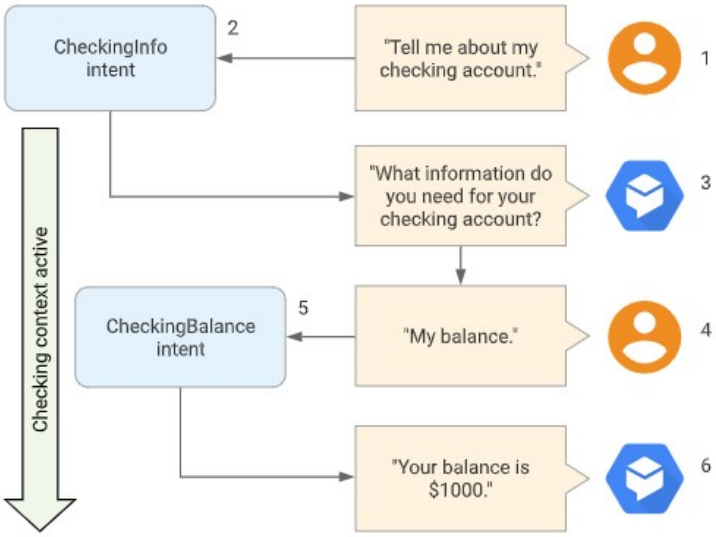

Figura 6: Diagrama do funcionamento de Contexto no *Dialogflow*.
Fonte: https://cloud.google.com/dialogflow/docs/contexts-overview

A Figura 6 ilustra o funcionamento de contexto. Quando o usuário solicita uma ação, a intenção correspondente é acionada. Essa intenção tem um contexto de saída, para que o contexto seja ativado. O agente solicita uma informação mais específica sobre a intenção, obtendo a respectiva resposta do usuário. Com essa informação, o *Dialogflow* aciona outra intenção que se relaciona ao contexto de entrada caracterizado pela informação que o usuário forneceu. Finalmente, o sistema executa as consultas necessárias no banco de dados e retorna a resposta desejada para o usuário.

O *Dialogflow* proporciona integração com várias plataformas entre elas, *Google Assistant*, *Slack*, *Telegram*, *Facebook Messenger*, *Twitter*, *Twilio* e *Skype*. Também possui canais de texto e fala e integração Web e *mobile*.

A versão para língua inglesa possui funcionalidades adicionais. A mais interessante é o recurso de *Knowledge Base*, capaz de adicionar arquivos *.csv*, *.json*, ou *.html* para criar uma FAQ. O *Dialogflow* também possui o recurso de *slots*, que são campos de preenchimento para coleta das informações necessárias para o *chatbot* realizar uma determinada ação. Um exemplo são as informações necessárias para um pedido de uma pizza como sabor, tamanho, quantidade, endereço, etc. A integração com outras aplicações pode ser realizada através de *Webhooks*.

Em Setembro de 2020 a *Google* anunciou uma versão mais atualizada desse serviço. O chamado *Dialogflow CX* funciona baseado no conceito de máquina de estados e possui as implementações mais atuais de processamento de linguagem natural. Porém, o *Dialogflow CX* existe apenas na versão inglesa e seus recursos são limitados quando comparados ao

serviço anterior ainda vigente.

3.7 *Watson Assistant*

O *Watson Assistant* é uma plataforma de IA de conversação que ajuda a fornecer aos clientes respostas rápidas, diretas e precisas às perguntas, em vários aplicativos, dispositivos ou canais. Ele possui uma maneira intuitiva de organizar o diálogo através de uma árvores de nós, conforme a Figura 7. Ele é um serviço alocado na *IBM Cloud*.

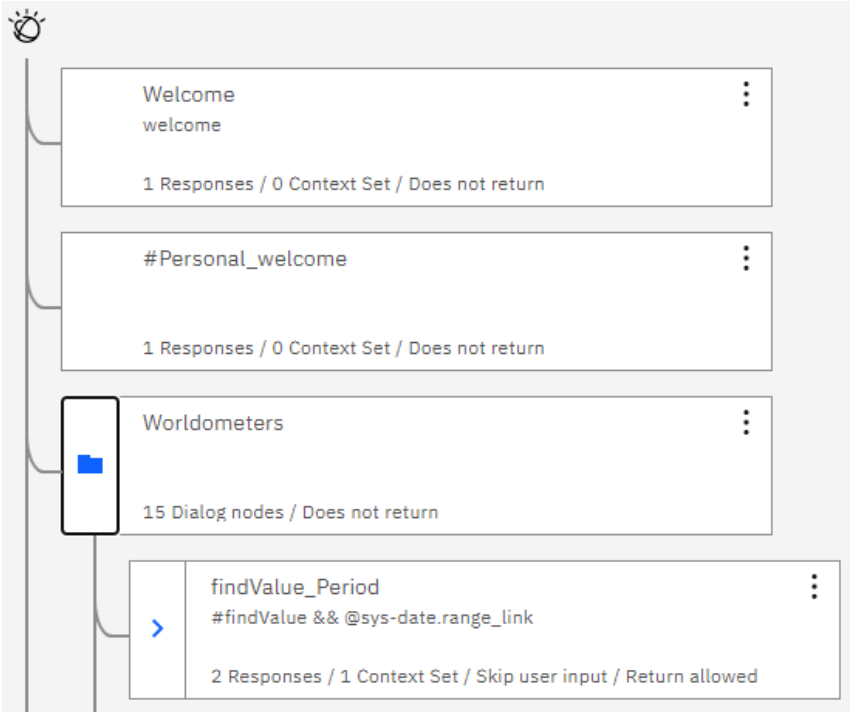

Figura 7: Exemplo de uma árvore de nós do *Watson Assistant*.

É possível agrupar as intenções que fazem parte de um mesmo assunto em pastas. A ferramenta de *text-annotation* está presente somente na língua inglesa. Entretanto, o algoritmo de processamento de linguagem natural do *Watson* possui menor tempo de resposta e apresenta as tecnologias mais atuais.

O *Watson Assistant* não possui intenções *follow-Ups*. O controle de diálogo pode ser realizado através da relação dos nós da árvore. Um determinado nó pode possuir nós filhos, conforme a Figura 8.

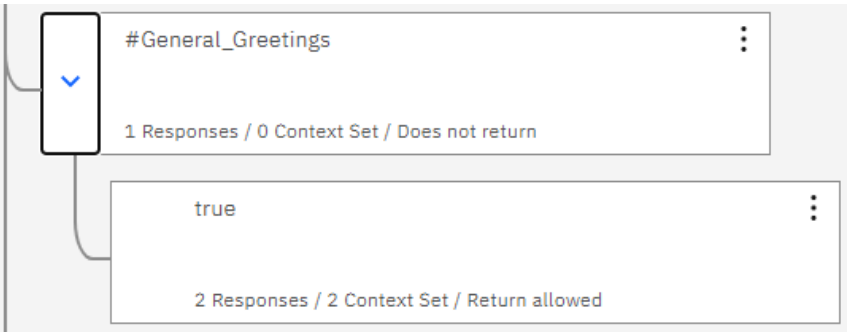

Figura 8: Relação entre nós da árvore no *Watson Assistant*.

Cada nó é ativado somente se atender a um requisito, podendo ser o reconhecimento de uma intenção, entidade, variável de contexto, ou expressões lógicas. Existe a possibilidade de direcionar o diálogo para um nó específico e implementar funcionalidades de *Webhook* e *Slots*.

O *Watson Assistant* organiza as entidades em duas categorias: as do sistema e as criadas pelo usuário. A solução da IBM não possui entidade para localização até o momento da conclusão do presente trabalho. Além disso, não existe diferença entre as entidades de número cardinal e ordinal, sendo que ambos estão inclusos na entidade *@sys-number* com a diferença de que cada uma possui um modificador diferente. Dessa forma, utilizando métodos de linguagem de expressão é possível determinar o tipo da variável de contexto dada [7].

Cada *bot* é chamado de *Assistant* no *Watson*. "Um assistente é um robô cognitivo para o qual você inclui qualificações que permitem que ele interaja com seus clientes de maneiras úteis" [7]. Cada *Assistant* apresenta apenas uma *Skill*, ou qualificação de diálogo. "Uma qualificação de diálogo é um contêiner para os artefatos que definem o fluxo de uma conversa que seu assistente pode ter com seus clientes" [7].

Contextos no *Watson Assistant* não expiram de acordo com as rodadas da conversa, ou tempo de ativação. Uma vez que são armazenados numa variável específica e determinada pelo usuário, o contexto pode ser aplicado em qualquer nó. Portanto, diferentemente do *Dialogflow*, no *Watson Assistant* não existe contextos de entrada e saída.

O *Watson Assistant* suporta um pouco mais de 10 idiomas, oferecendo canais de texto e fala e integrações Web, *mobile* e com outras aplicações como *Slack*,*Facebook Messenger* e *Wordpress*. Por se tratar de um serviço, o *Watson Assistant* possui três planos de assinatura.

4 *FAKE NEWS*

Fake News, definidas pelo *New York Times* como "uma história inventada com a intenção de enganar", muitas vezes para um ganho secundário, são sem dúvida um dos desafios mais sérios que a indústria da notícia enfrenta atualmente. Em uma pesquisa da *Pew Research* [8] em dezembro, 64% dos adultos norte-americanos disseram que "notícias inventadas"causaram "muita confusão"sobre os fatos dos eventos atuais. A Figura 9 mostra a procura pelo termo *fake news* nos últimos cinco anos. Os dois momentos de pico referem-se, respectivamente, à eleição presidencial para presidência da República do Brasil em 2018 e à disseminação do coronavírus ao redor do mundo.

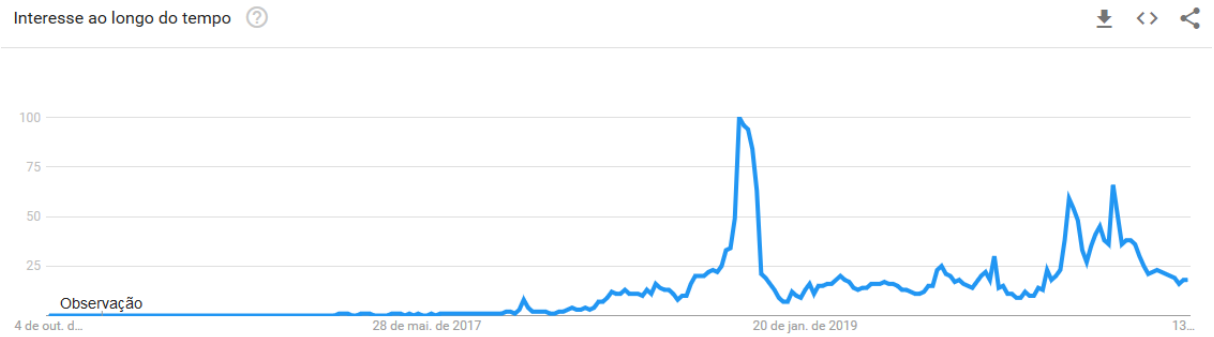

Figura 9: Procura pelo termo *fake news* no *Google*.
Fonte: https://trends.google.com/ **Acessado em: 28/09/2020**

A definição mais utilizada atualmente é a concebida por *Claire Wardle* [9], em que as disfunções da informação são separadas em três categorias:

1. ***Misinformation*:** ocorre quando uma notícia falsa é criada sem a intenção de causar dano. Geralmente é devido a erros na data e más traduções. Possui também uma origem social, que é a de quando uma sátira ou ironia é levada a sério por parte do leitor.

2. ***Mal-information*:** são notícias verdadeiras com o intuito de causar dano. Alguns exemplos são os dicursos de ódio e vazamentos de documentos confidenciais.

3. ***Desinformation*:** são notícias falsas com o propósito de prejudicar.

Dentre essas disfunções estão destacadas sete maneiras de manifestação:

	Misinformation	Desinformation	Mal-information
Sátira ou paródia	X		
Falsas conexões	X		X
Conteúdo enganoso		X	
Falso contexto	X		
Conteúdo impostor			X
Conteúdo manipulado		X	
Conteúdo fabricado		X	

4.1 Difusão de uma notícia

As notícias provenientes de fontes confiáveis passam por um processo de produção. Quando um fato ocorre, existe uma fonte, onde é realizada a apuração do evento. O resultado dessa apuração é convertido em um texto pelo jornalista e direcionado à fonte disseminadora, que são os jornais, revistas, blogs, etc. Nessa etapa ocorre a verificação das informações, refinamento através de figuras, gráficos e tabelas. Por fim, ocorre a publicação desse conteúdo nos veículos de comunicação como *websites* e redes sociais.

Segundo *Wardle* [9], as *fake news* passam por um processo com mais elementos. Inicialmente, a mensagem é criada, porém não provém de uma fonte confiável, ou foi apurada com rigor. Essa mensagem é transformada em um produto de mídia e, em seguida, distribuída, ou feita pública. Essas fases se conectam e interagem através de três elementos: agentes, mensagem e interpretador.

O agente é responsável pela etapa inicial de criação da mensagem, ou notícia. Dependendo da etapa, ele possui características diferentes. *Wardle* sugere as seguintes perguntas para compreendê-lo:

- Que tipo de ator ele é?
- Quais são suas motivações?
- Tem a intenção de enganar?
- Tem a intenção de causar dano?

- Qual audiência ele quer atingir?

A mensagem pode ser comunicada oralmente, através de textos, ou materiais de áudio ou visuais. Algumas características importantes são:

- Quão durável é a mensagem?
- Quão acurada é a mensagem?
- A mensagem está se passando por uma fonte confiável?
- Qual é o público para o qual a mensagem é direcionada?

O interpretador é responsável por entender a mensagem. Sua compreensão é influenciada por fatores como status sociocultural, posições políticas e experiências pessoais.

Segundo *Collier* [10], a exposição constante ao termo *fake news* resulta em níveis menores de confiança na mídia e um aumento na dificuldade do reconhecimento de notícias verdadeiras.

O processo de avaliação da veracidade de uma notícia pode ser dividido em etapas. Um primeiro passo para identificar notícias falsas é entender o que outras organizações de notícias estão dizendo sobre o assunto.

5 *PYTHON LIBRARIES*

5.1 *Web Scraping*

Web scraping é a maneira de extrair informação sem interagir com uma *API*. Em geral, um programa faz uma requisição de uma página da *web*, analisa a resposta e extrai a informação desejada. Isso permite extrair um grande volume de informação de várias páginas rapidamente. Para realizar o *web scraping* os dois pontos essenciais são a capacidade de realizar requisições *HTTP* e de manusear o conteúdo da página em *HTML* e *Javascript*.

5.1.1 *covid-daily*

A biblioteca *covid-daily* [11] fornece fácil manuseio das informações disponíveis no site *Worldometers* sobre o coronavírus. Como uma das fontes de informação popular sobre a doença, o *Worldometers* foi escolhido como ponto inicial de extração de informações e o *covid-daily* permite acessar as informações do site sem ter que se preocupar com mudanças constantes de estrutura que o site pode sofrer. A biblioteca permite acessar as informações gerais na tabela principal do *Worldometers* e também as informações individuais de cada país, o que atende às necessidades do projeto.

5.2 *scikit-learn*

O *scikit-learn* [12] é uma biblioteca de aprendizado de máquina para *Python* com suporte para aprendizado supervisionado e não supervisionado. Ele fornece ferramentas ajuste de modelo, pré-processamento de dados, avaliação de modelos e outras utilidades. Uma vantagem é a disponibilidade de diversos modelos prontos para classificadores como *Gaussian Naive Bayes*, *Logistic Regression* e *Support Vector Machine*. Ferramentas de processamento de texto também estão presentes na biblioteca como tokenização, *occurance counting* e *term frequency–inverse document frequency*. Assim, a biblioteca proporciona

um ambiente completo para classificadores de *fake news*.

5.3 *Pickle*

Pickle é uma biblioteca de *Python* que permite a serialização e desserialização de objetos *Python*. O objeto é transformado em *byte stream* e é uma maneira de salvar um classificador treinado.

6 ALGORITMOS DE CLASSIFICAÇÃO

6.1 Regressão Logística

A regressão logística é um algoritmo que lida com problemas de classificação, ou seja, é utilizado para categorizar uma variável em diversas classes. A base da regressão logística reside nos conceitos de probabilidade e estatística. Essa regressão mede a relação entre a variável dependente categórica e uma ou mais variáveis independentes, estimando as probabilidades usando uma função logística. Os resultados da análise ficam contidos no intervalo entre 0 e 1. Esse algoritmo é muito utilizado, por exemplo, em previsão de riscos. Há vários tipos de regressão logística: binomial, ordinal e multinomial.

- **Regressão logística binomial:** os objetos são classificados em dois grupos ou categorias;
- **Regressão logística ordinal:** trabalha com o conceito de categorias ordenadas, ou seja, os objetos são classificados de acordo com classes que possuem um ordem pré-determinada;
- **Regressão logística multinomial:** os objetos são classificados em categorias que não possuem ordem entre si.

A regressão logística possui vantagens como facilidade para lidar com variáveis independentes categóricas, fornecimento de resultados em termos de probabilidade, requerimento de pequeno número de suposições e alto grau de confiabilidade.

6.1.1 Função logística

Na regressão logística, a probabilidade de ocorrência de um evento pode ser estimada diretamente. No caso particular da regressão binomial e de haver um conjunto de p

variáveis independentes X_1,X_2, ..., X_p o modelo de regressão logística pode ser descrito da seguinte forma:

$$P(Y = y) = \frac{1}{1 + e^{-g(x)}} \tag{6.1}$$

onde,

$$g(x) = B_0 + B_1 X_1 + \cdots + B_p X_p \tag{6.2}$$

Os coeficientes B_0, ..., B_p são estimados a partir do conjunto de dados utilizando o método da máxima verossimilhança, em que se encontra uma combinação de coeficientes que maximiza a probabilidade da amostra ter sido observada. A curva da regressão logística possui a forma de uma sigmoide, conforme a Figura 10. Ela possui, portanto, as seguintes propriedades:

$$\lim_{g(x)\to+\infty} P(Y = y) = 1$$

$$\lim_{g(x)\to-\infty} P(Y = y) = 0$$

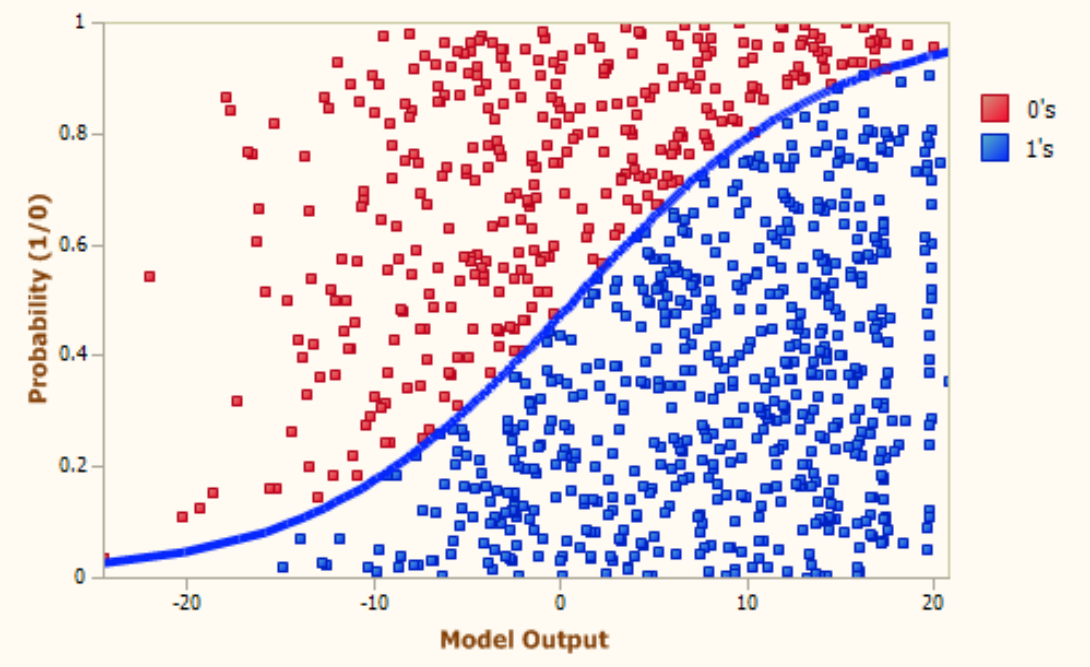

Figura 10: Curva sigmoide de Regressão Logística.

6.2 *Naive Bayes*

O *Naive Bayes* é um algoritmo que permite replicar a lógica do classificador de Bayes quando não é conhecida a distribuição dos parâmetros e rótulos. A distribuição é estimada assumindo a hipótese de que dado o rótulo, as probabilidades de cada par de parâmetros são independentes. Ele é um algoritmo popular que obteve sucesso na classificação de documentos e no filtro de *spam*. A descrição a seguir do algoritmo foi baseada na documentação do classificador no *scikit-learn*.

De acordo com o teorema de Bayes:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n \mid y)}{P(x_1, \ldots, x_n)} \tag{6.3}$$

Utilizando a hipótese de independência dado o rótulo:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n \mid y)}{P(x_1, \ldots, x_n)} \tag{6.4}$$

Para todo i, a equação pode ser escrita como:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n \mid y)}{P(x_1, \ldots, x_n)} \tag{6.5}$$

Como $P(x_1, \ldots, x_n)$ é constante dado a entrada, a regra de classificação fica:

$$\begin{gathered} P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y) \\ \Downarrow \\ \hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y), \end{gathered} \tag{6.6}$$

e utiliza-se estimação *maximum a posteriori* (MAP) para estimar $P(y)$ e $P(x_i \mid y)$ com $P(y)$ a frequência relativa de y nos dados de treinamento.

Há diferentes tipo de classificadores *Naive Bayes* de acordo com a distribuição assumida para $P(x_i \mid y)$. Como exemplo, para o classificador *Naive Bayes* gaussiano

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right) \tag{6.7}$$

e os parâmetros σ_y e μ_y são estimados com máxima verossimilhança.

6.3 *Support Vector Machine*

Support Vector Machines (SVMs) são modelos de aprendizado supervisionado que também são populares para o problema de classificação. O SVM constrói um hiperplano ou conjunto de hiperplanos que pode ser usado para dividir o espaço entre os rótulos. Quanto maior a distância entre o hiperplano e os pontos de treinamento mais próximos à margem, menor o erro de generalização. A descrição a seguir do algoritmo foi extraída da documentação do classificador no *scikit-learn*.

O problema de classificação pode ser descrito como, dado vetores de treinamento $x_i \in \mathbb{R}^p$, i=1,...,n com dois rótulos e um vetor $y \in \{1, -1\}^n$, o objetivo é achar $w \in \mathbb{R}^p$ e $b \in \mathbb{R}$ de tal maneira que a predição dada por $\text{sign}(w^T\phi(x) + b)$ esteja correta para a maioria das amostras.

O *Support Vector Classifier* (SVC) resolve o problema

$$\begin{aligned} \min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i \\ \text{sujeito a } y_i(w^T\phi(x_i) + b) \geq 1 - \zeta_i, \\ \zeta_i \geq 0, i = 1, ..., n \end{aligned} \tag{6.8}$$

A margem é maximizada ao minimizar $||w||^2 = w^T w$, ao passo que uma penalidade é imposta quando uma amostra é classificada errado ou dentro da margem da fronteira. Para o caso em que os rótulos não são perfeitamente separáveis por hiperplanos, há uma tolerância de distância de ζ_i da margem de fronteira. O termo C controla o peso da penalidade e serve como uma parâmetro de regularização inverso.

Um problema que acompanha o anterior é

$$\begin{aligned} \min_{\alpha} \frac{1}{2} \alpha^T Q \alpha - e^T \alpha \\ \text{sujeito a } y^T \alpha = 0 \\ 0 \leq \alpha_i \leq C, i = 1, ..., n \end{aligned} \tag{6.9}$$

onde e é o vetor de uns e Q é uma matrix positiva semidefinida n por n, $Q_{ij} \equiv y_i y_j K(x_i, x_j)$, onde $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ é o *kernel*. Os termos α_i são chamados de coeficientes duais e possuem limite superior C.

Ao resolver o problema de otimização, a saída da função de decisão para uma amostra

x é

$$\sum_{i \in SV} y_i \alpha_i K(x_i, x) + b, \tag{6.10}$$

e o sinal corresponde ao rótulo predizido. Somente é necessário somar os vetores de suporte (amostras no inteior da margem), pois os coeficientes duais α_i são nulos para as demais amostras.

6.4 *Random Forest*

O modelo de *Random Forest* é um outro modelo popular para a classificação de texto. Ele é composto por diversas árvores de classificação, que por sua vez, são conjuntos de *splits*, ou decisões de como separar os dados. Após passar pelos *splits* e chegar na folha, o rótulo do dado é determinado.

A maneira de criar a árvore é com estimação do melhor *split* local com o cálculo de entropia, ganho de informação e razão de ganho ou com a utilização do *Gini index*. Os *splits* são definidos até que os dados de treino nos últimos nós da árvore sejam homogêneos ou sejam pequenos. Uma árvore de classificação apresenta *overfitting* para os dados de treinamento e para melhorar o seu desempenho foram concebidas as *bagging trees* e o *random forest*.

A *bagging tree* é um modelo em que são treinadas diversas árvores de classificação e o rótulo é escolhido com a média das árvores, o que reduz a variância. Ainda, se para a escolha dos *splits* somente um subconjunto das *features* for considerado, o resultado é uma *random forest*. Como os modelos anteriores, o *scikit-learn* possui um modelo de *random forest*.

PARTE III

METODOLOGIA

7 METODOLOGIA

Foi definido o *Watson Assistant* como a plataforma para a construção do *chatbot*, uma vez que possui o menor tempo de resposta, maior facilidade com integrações com outros serviços da *IBM Cloud*, o algoritmo de processamento de linguagem natural mais acurado e atualizado, uma interface intuitiva e apresenta maior flexibilidade no manuseio das intenções, entidades, fluxo de conversa, contextos e chamadas de *webhook*.

O *chatbot* é composto pela integração do *Watson Assistant* com aplicações em *Python*. O *Watson Assistant* recebe o *input* do usuário e identifica a intenção que mais se assemelha à frase do usuário e as entidades presentes. Em casos em que essas informações precisam ser processadas, elas são enviados para uma aplicação em *Python*, através do *webhook*, onde ocorre o seu tratamento. Os casos em que isso ocorre são dois, o usuário pode requisitar uma informação pontual sobre o coronavírus ou verificar a veracidade de uma notícia. Em relação às outras funcionalidades, o *Watson Assistant* é capaz de responder sem necessidade de aplicações de suporte.

Para a a situação em que o usuário requisita uma informação pontual sobre o coronavírus, a aplicação deve consultar o *covid-daily* e extrair as informações necessárias para responder o usuário. Essas informações possuem natureza quantitativa como números de casos, mortes e casos ativos até o momento. Para cada informação quantitiva presente no *Worldometers* foi criada uma intenção. Isso auxília na localização da informação desejada pelo usuário e permite a formulação da resposta a ser devolvida para o *Watson Assistant*.

Para a situação em que o usuário deseja verificar a veracidade de uma notícia, uma aplicação que possui um algoritmo de classificação de notícias é chamado. O usuário insere o título e o *corpus* de uma notícia na web e o algoritmo retorna a classificação da mesma. Como atributos, utilizam-se o título da notícia e o *corpus* do texto no modelo de aprendizado supervisionado.

7.1 Arquitetura

Um *bot* especialista possui conhecimento sobre um domínio específico e pode responder perguntas relacionadas a ele. No *Watson Assistant*, cada *bot* especialista é equivalente a uma *skill*. Se um usuário desejar ter uma conversa entre domínios, ele terá que alternar entre os diferentes *bots*. Existem cenários em que um usuário deseja ter uma conversa envolvendo vários domínios como por exemplo durante um planejamento de uma viagem. Nessa situação o usuário pode querer consultar a previsão do tempo e reservar um voo. Para permitir a mudança de domínio durante a conversa, propõem-se a arquitetura de multi-agente, inspirada em [13], conforme a Figura 11.

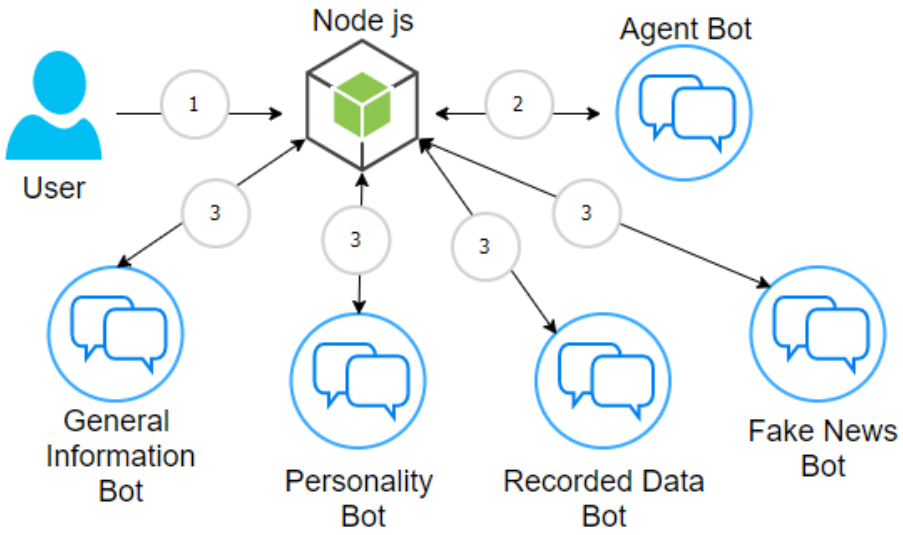

Figura 11: Arquitetura de *Bot* Multi-Agente.

A arquitetura consiste em um *bot* de interface, chamado de *Agent Bot*, e outros *bots* de domínio específicos, chamados de *skills*. O funcionamento ocorre nas seguintes etapas:

1. O usuário acessa a aplicação e envia uma mensagem.

2. O aplicativo Nodejs envia a mensagem do usuário para o *Agent Bot*. Em seguida, o *Agent Bot* determina a intenção da mensagem e responde com os detalhes específicos da *skill* para a qual a mensagem precisa ser redirecionada

3. O aplicativo Nodejs envia mensagem para a *skill* específica. Ela responde para o aplicativo Nodejs, imprimindo a resposta.

7.2 Skills

7.2.1 *Agent Bot*

O *Agent Bot* apresenta todas as frases de treinamento das *skills*, armazenadas nas respectivas intenções. Dessa forma, maximiza-se a correspondência do *input* do usuário com a *skill* responsável pela intenção. Em cada nó da árvore do *Agent Bot* que aloca uma *skill* é setado duas variáveis de contexto. A primeira é responsável por informar o nome da *skill* para a qual a mensagem do usuário está sendo direcionada. A segunda é uma variável booleana que verifica se a mensagem deve ser redirecionada para outra *skill* ou não.

7.2.2 *Personality*

A *skill* de personalidade foi criada para aproximar o comportamento do *CovidBot* ao de uma pessoa. Ela é encarregada de cumprimentar o usuário, se despedir, descrever o próprio *bot* e outras funcionalidades para preencher a conversa, como contar piadas, consolar o usuário, agredecer elogios. Para cada situação foram criadas as intenções e diálogos apropriados, sem a necessidade de uso de entidades.

As intenções da *skill* de personalidade são 21 ao todo. Para cada intenção devem ser definidos o nome dela e as frases de treinamento. Isso é suficiente para que, quando um usuário escrever um *input* similar a uma das frases de exemplo, a intenção correta seja selecionada.

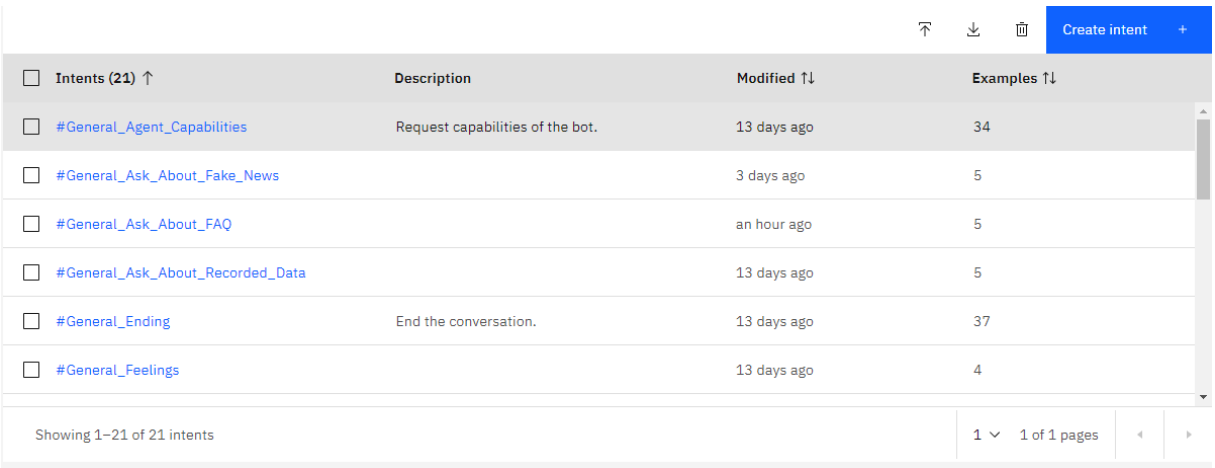

Figura 12: Intenções da *Personality skill*.

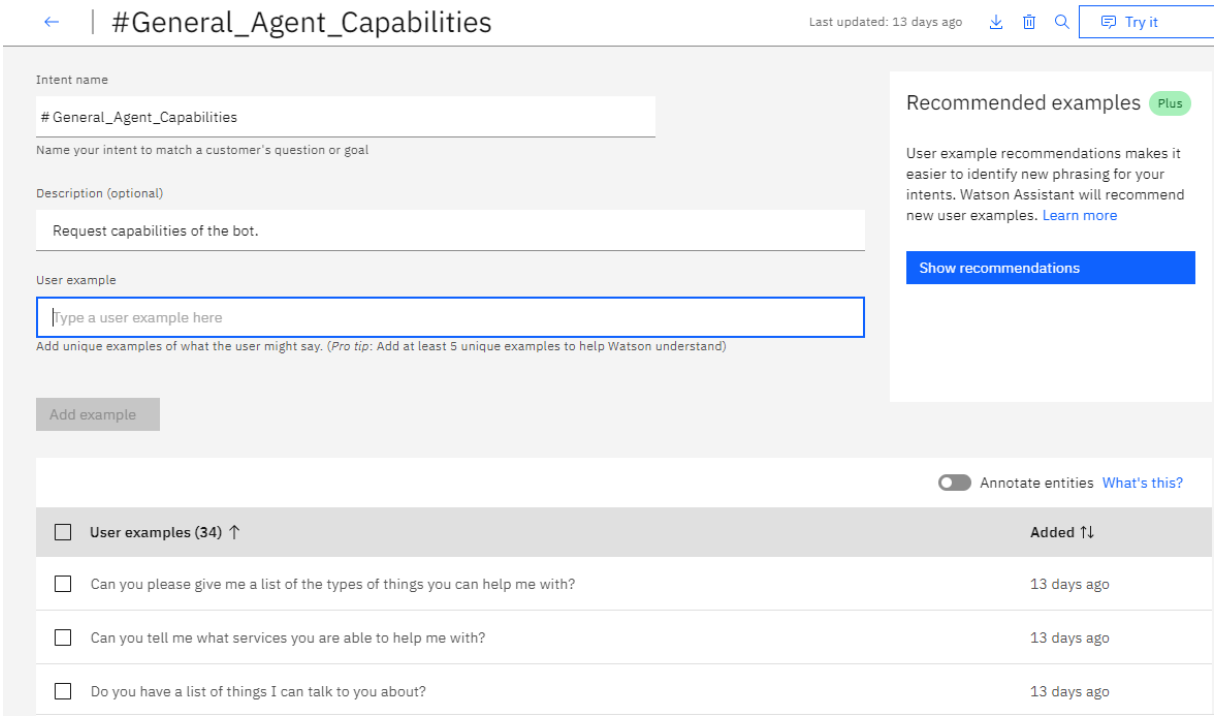

Figura 13: Interface de criação e modificação de intenção na *personality skill*.

Para finalizar a *skill* de personalidade é necessário criar os nós do *dialog*. Para cada intenção é criado um nó com condição de entrada, contexto e resposta. Para a personalidade, a condição de entrada é a identificação da intenção correspondente. Em seguida, é atribuido o valor "agent"para a variável de contexto *destination bot*, o que atribui a responsabilidade de receber o próximo *input* do usuário para o *Agent Bot*. Finalmente, a resposta a ser devolvida é definida.

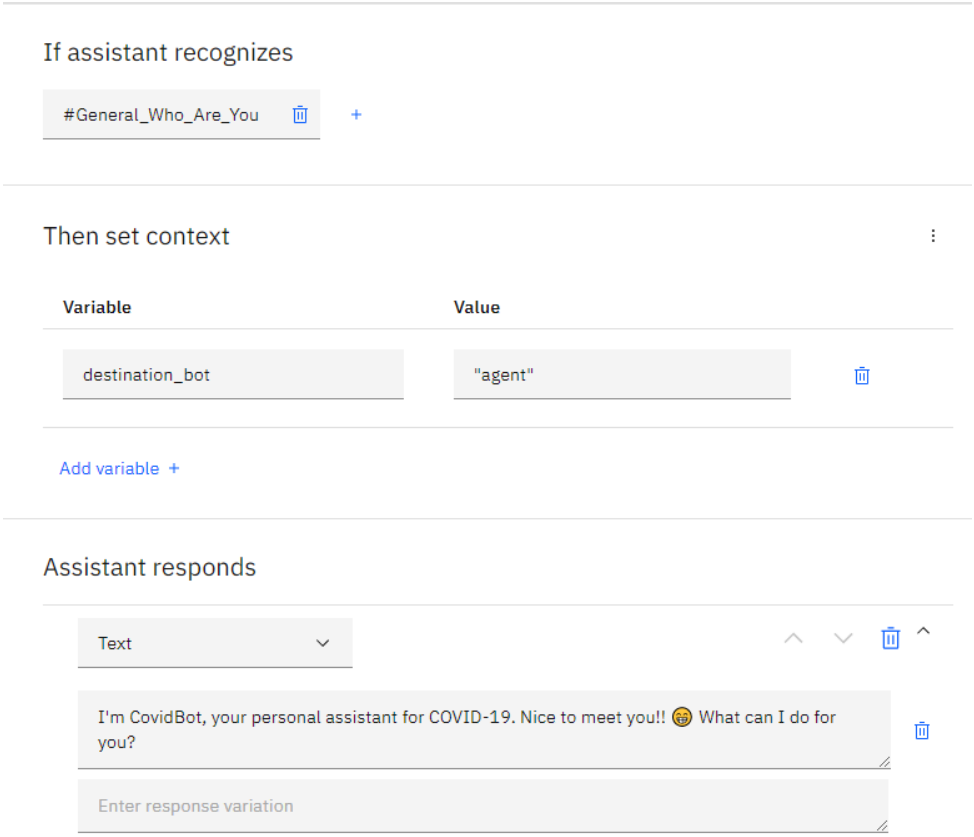

Figura 14: Interface de criação e modificação de nó do diálogo na *personality skill*.

Isso encerra a implementação da personalidade do *CovidBot*. Para verificar o funcionamento dessa *skill* foi utilizado o o console de desenvolvimento.

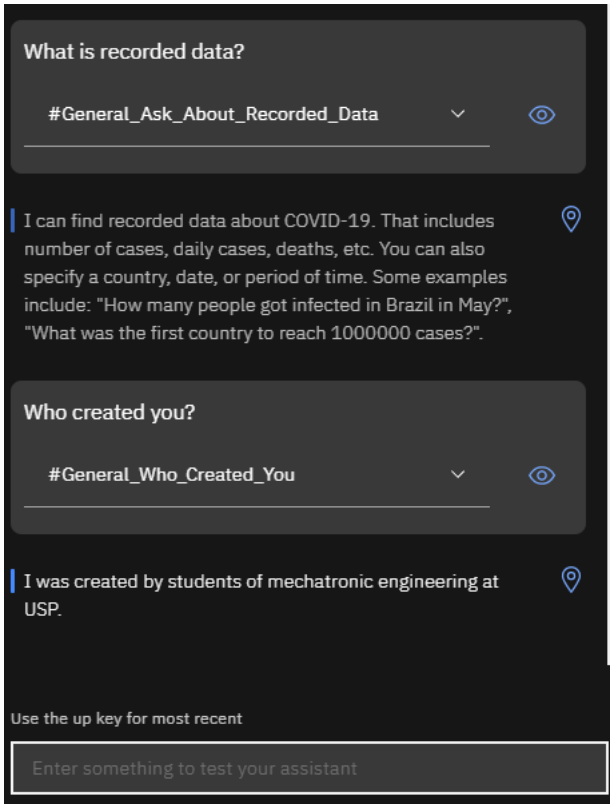

Figura 15: Conversa no console de desenvolvimento na *personality skill*.

7.2.3 *FAQ*

A *skill* de *FAQ* tem como objetivo responder perguntas gerais sobre o covid-19. Ela está encarregada de informar sobre tópicos como o que é o COVID-19, sintomas da doença, uso de máscara e a existência de vacinas. A informação utilizada para a implementação dessa *skill* foi proveniente de seções de FAQ de diversas fontes confiáveis como a faculdade de medicina da universidade Johns Hopkins [14], a Organização das Nações Unidas [15] e a Organização Mundial de saúde [16].

Para cada par de pergunta e resposta selecionado foi necessário criar uma intenção na *skill* de *FAQ*. Ao todo foram implementadas 52 intenções, que abrangem tópicos como contagio por pessoas sem sintomas, risco ao pedir comida por entrega, diferença entre COVID-19 e SARS, contaminação por animais e a origem da doença. O foco foi em abranger diversos interesses e apresentar informações confiáveis.

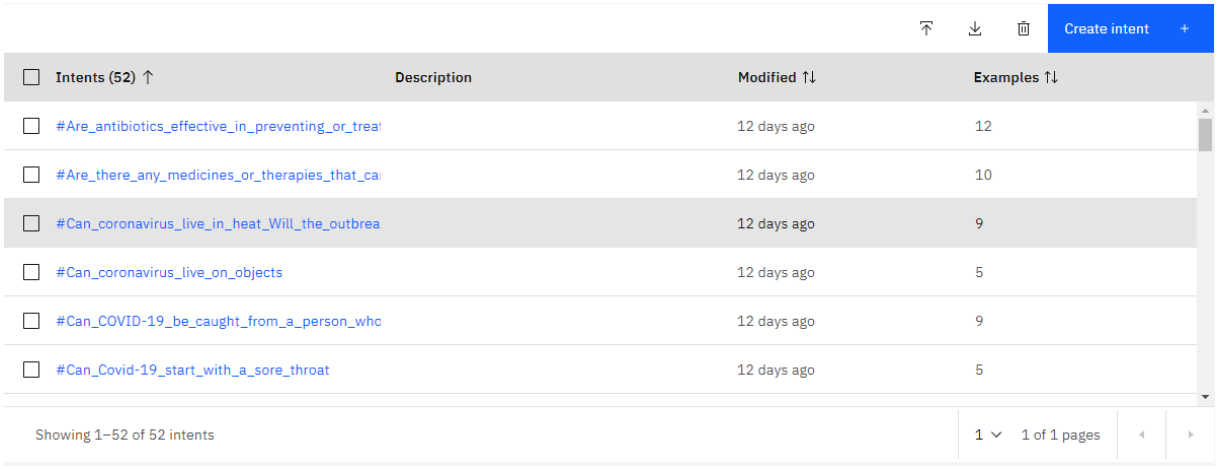

Figura 16: Intenções da *FAQ skill*.

Com as intenções prontas, foi criado um nó de diálogo para cada uma. Assim como para a *personality skill*, a condição de entrada foi a identificação da intenção correspondente. Também foi atribuido o valor "agent"para a variável de contexto *destination bot* para direcionar a conversa para o *Agent Bot*. Por último, a resposta a ser devolvida para o usuário é a resposta encontrada na *FAQ* com a pergunta correspondente.

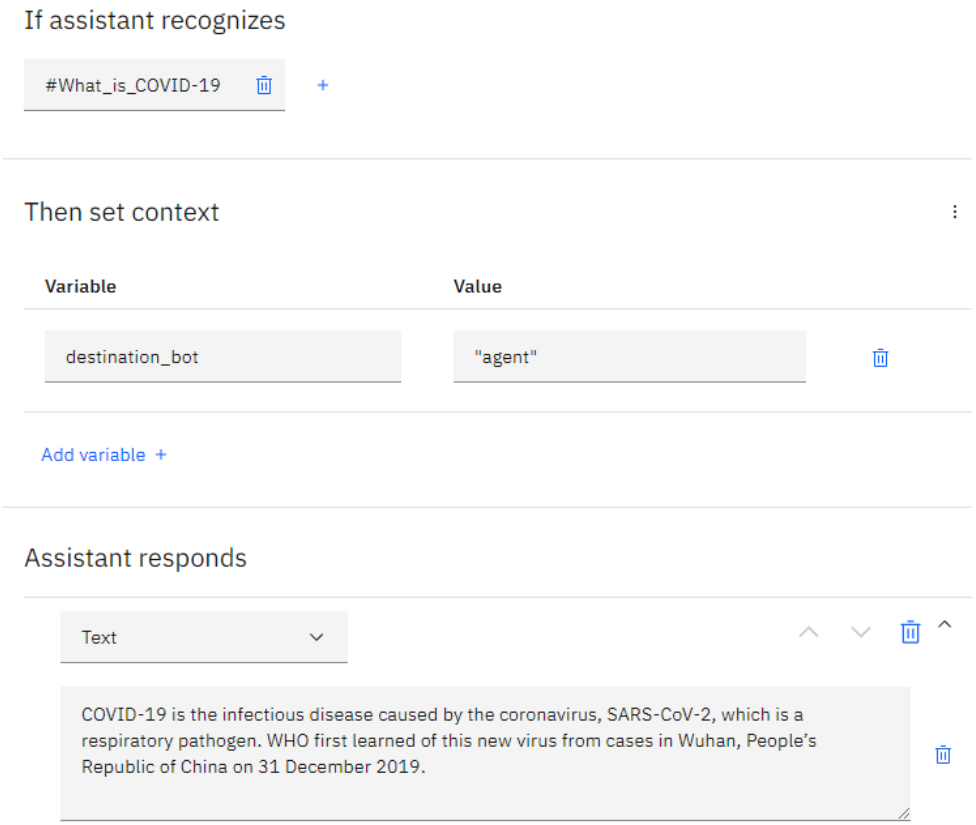

Figura 17: Interface de criação e modificação de nó do diálogo na *FAQ skill*.

Para verificar o funcionamento esperado da *skill* foi utilizado o console de desenvol-

vimento.

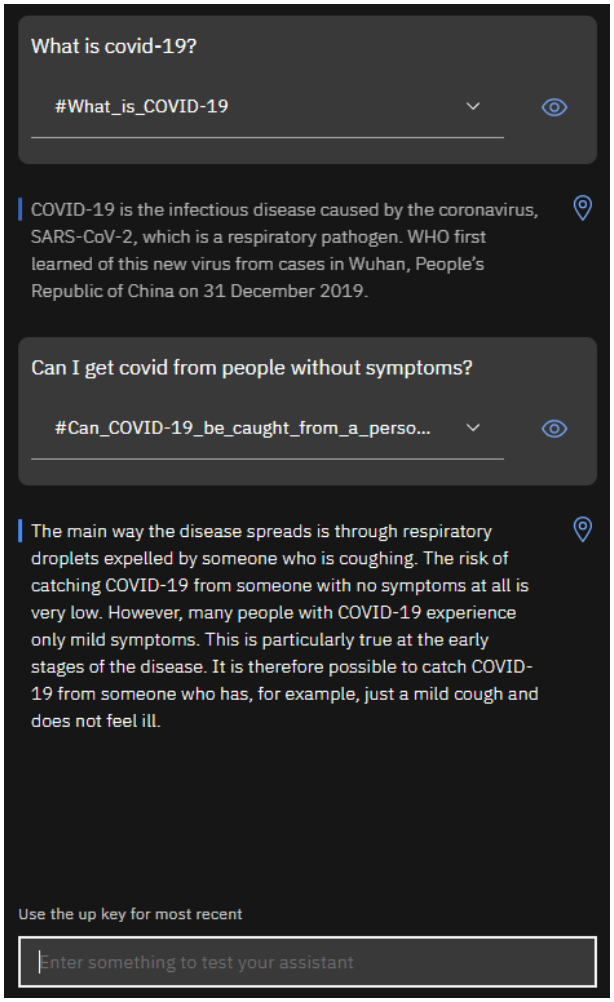

Figura 18: Conversa no console de desenvolvimento na *FAQ skill*.

7.2.4 *Recorded data*

A *skill Recorded data* está encarregada de responder perguntas sobre informações numéricas relacionadas ao progresso do covid nos países. Essas informações estão disponíveis do *site Worldometers* e foram acessadas através da biblioteca *covid-daily* [11].

Os dados estão divididos nas categorias: casos totais, casos diários, mortes, mortes diárias, casos ativos e recuperações diárias. Também é possível selecionar o dado de acordo com o país e a data. A informação é atualizada diariamente e os dados do dia atual estão disponíveis.

O *chatbot* deve responder a perguntas de vários tipos. Inicialmente, restrigindo-se à funcionalidade de obter informações, é possível classificá-las de acordo com o tipo de dado a ser retornado. Para perguntas do tipo "Quando a Itália superou 300 casos con-

firmados?", a resposta é dada por um tempo específico representado pela identificação do advérbio "quando". Os casos de uso foram divididos da seguinte forma, no que diz respeito à obtenção de informações:

1. Resposta é dada por um valor numérico
   Exemplo: "How many Spanish have died of coronavirus?"

2. Resposta é dada por um tempo específico
   Exemplo: "When did China surpass 100 deaths?"

3. Resposta é dada por uma localização
   Exemplo: "Which country has the most cases?"

4. Resposta é dada por um período de tempo
   Exemplo: "How long did it take Brazil to reach 100 cases?"

5. Respostas de Sim/Não
   Exemplo: "Does Spain have more than 1000 deaths?"

A *Recorded Data* recebe uma variável dicionário que apresenta as seguintes informações:

- *intent*: intenção relacionada
- *number*: número de referência
- *ordinal*: número de referência ordinal
- *data_Type*: categoria de dado
- *date*: data específica
- *date_begin*: data específica de início quando um período é reconhecido
- *date_end*: data específica de fim quando um período é reconhecido
- *location*: país
- *comparator*: expressão de comparação
- *superlative*: indicador de comparação superlativa

O *chatbot* não consegue reconhecer as categorias de dados. Criou-se então entidades específicas, listadas anteriormente. No caso de informações ausentes no *input* do usuário, o *Watson Assistant* atribui valor ***None***. Além disso, o *Watson* não consegue reconhecer os períodos de uma forma eficaz. O algoritmo de linguagem natural não possui raciocínio, ou seja, períodos informados como "desde Março de 2020", "até Abril de 2020"e "de Maio até Agosto"devem ser tratados de forma diferente. Para isso, o *Watson Assistant* faz uso do *entity-modifier*, o qual especifica a palavra modificadora da entidade período no caso. Alguns exemplos de *entity-modifier* são "desde", "de"e "até". Dessa forma, basta verificar a presença dessas palavras dentro do período informado para tratar a entidade de forma mais acurada.

Na aplicação *python* foi criada uma variável chamada *mapper_value* com a finalidade de organizar o fluxo durante o tratamento dos dados. Ela é calculada de acordo com o seguinte código;

```
def get_mapper_value(self,req):
    mapper_value = 0
    if (req.get('date') != None) and (req.get('period_modifier') ==
        ↪ None):
        mapper_value += 1
    if (req.get('date_end') != None) and (req.get('date_begin') !=
        ↪ None):
        mapper_value += 2
    elif (req.get('period_modifier') != None) and (req.get('date') !=
        ↪ None):
        mapper_value += 2
    if req.get('comparator') != None:
        mapper_value += 4
    if req.get('number') != None:
        mapper_value += 8
    if req.get('superlative') != None:
        mapper_value += 16
    if req.get('ordinal') != None:
        mapper_value += 64
    return mapper_value, req
```

Dessa forma, garante-se a unicidade do valor do *mapper_value* e direciona-se a requisição para o trecho responsável tendo certeza das informações que foram recebidas. A combinação do *mapper_value* com a intenção permite identificar a informação desejada pelo usuário e localizá-la para formulação da resposta.

O *Worldometers* apresenta os dados para um dado país de acordo com gráficos. Ao realizar uma busca para uma determinada data específica, observa-se um alto custo computacional, uma vez que o *crawler* deve buscar comparar as informações na *webpage* de cada país, no respectivo gráfico e percorrer até a data solicitada. Uma forma otimizada de contornar o problema é gerar arquivos *.csv* que contêm as categorias de dados para uma determinada data específica. Esses arquivos foram gerados, tratados e estão disponíves na pasta *./data* do projeto.

O *covid-daily* possui funções para a obtenção dos dados. Os dados tabelados representam as informações do dia atual. Na mudança de dia, os dados tabelados são armazenados nos respectivos gráficos de cada país. Portanto, os dados dos gráficos apresentam as informações no passado.

Inicialmente, realiza-se um filtro de acordo com o tipo de pergunta, ou seja, se o usuário deseja uma resposta de data, local, dado numérico, etc. Em seguida, filtra-se de acordo com as informações fornecidas no *input* do usuário através do valor do *mapper_value*. Assim, se for um dado que é encontrado através de uma tabela, o algoritmo coleta a informação no arquivo *python* específico da tabela. De modo análogo, o mesmo ocorre para dados de gráficos. No caso da categoria do dado não estiver disponível, padronizou-se que a função retorne um valor *None*. Essa exceção é tratada no momento de retornar a resposta para a aplicação Nodejs.

Flask é um *micro-framework* que possibilita que uma aplicação em *Python* escute requisições da rede em uma porta local. O *Ngrok* permite criar um túnel de conexão segura a partir do *localhost* e publicá-lo na internet. Ambos são utilizados para desenvolver a aplicação que recebe o *webhook* enviado pelo *Watson Assistant*. Dessa forma, é possível expor a aplicação em um endereço público.

O funcionamento da *skill* encontra-se na Figura 19

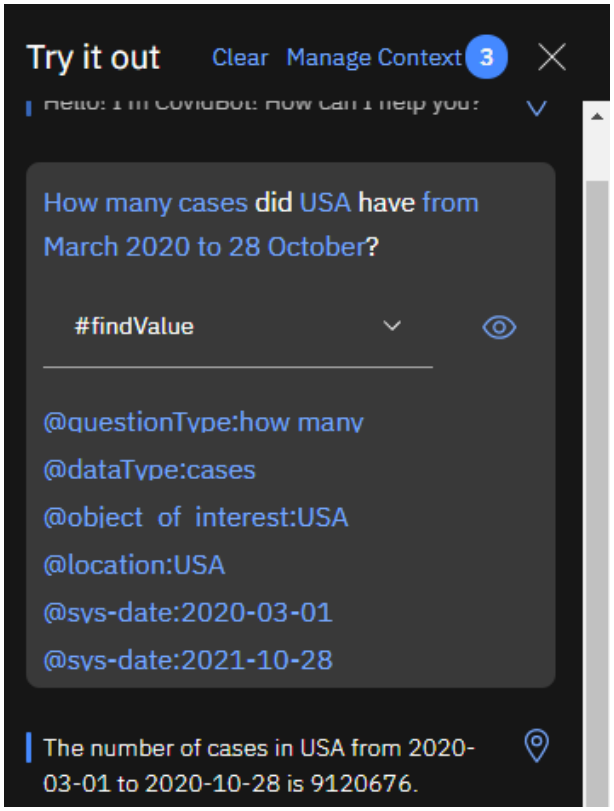

Figura 19: Conversa no console de desenvolvimento na *Record Data skill*.

7.2.5 Fake News

A *skill* de *fake news* verifica se um texto fornecido pelo usuário pode ser uma *fake news*. O objetivo é auxiliar o usuário com uma opinião secundária sobre a veracidade de notícias. O *chatbot* recebe o título e corpo da notícia e os envia para um programa em *Python* que rotula a notícia.

Para a criação de um classificador foi necessário a utilização de um conjunto de dados com título e corpo de notícias e com rotulos de falso ou verdadeiro. O conjunto de dados utilizado está disponível em [17]. Ele é composto por 586 notícias classificadas como verdaderias e 578 notícias classificadas como falsas, todas coletadas até o mês de Maio. Como a própria criadora do *dataset* comentou, foi desconsiderado o uso da fonte da notícia por causa da possibilidade de introdução de *bias* no treinamento. Para a utilização dos dados foi aplicado um pré-processamento que consiste na remoção de *html tags* e pontuação e tranformação para letras minúsculas.

Para a utilização dos dados, foi necessário transformar os textos em valores numéricos. O *Tf-idf vectorizer* foi escolhido para fazer essa transformação. Ele conta a quantidade de vezes que uma palavra se repete no texto e cria um vetor de números que representa

essa contagem. Outras funcionalidades do *Tf-idf vectorizer* é a contagem com a raíz morfológica das palavras com a ajuda de um *Porter-Stemmer* e a transformação dos pesos de frequência para reduzir a influência de termos comuns na classificação do texto.

Com os dados prontos para serem alimentados em um classificador, foram avaliados diversos modelos populares para classificação de texto: *random forest*, *linear SVM*, *logistic regression* e *multinomial naive bayes*. Para os modelos *random forest*, *linear SVM* e *multinomial naive bayes* foi necessário procurar os valores ótimos para os hiperparâmetros. Foram utilizados o *RandomizedSearchCV* e o *GridSearchCV* do *sklearn* para testar as combinações de hiperparâmetros. A diferença entre os dois é que o primeiro usa uma combinação aleatória dos valores disponíveis com um limite de iterações e o segundo testa todas as combinações. Assim, o *RandomizedSearchCV* permite buscas mais abrangentes e o *GridSearchCV* permite realizar buscas mais refinadas.

Com o método de busca definido, para o modelo de *random forest*, os hiperparâmetros a serem considerados foram: número de árvores, número de *features*, máxima profundidade, números mínimo de amostras para criação de *split*, número mínimo de amostras em uma folha e utilização de *bootstrap*. Após a aplicação de *RandomizedSearchCV* e *GridSearchCV*, a melhor combinação de hiperparâmetros encontrada foi de 300 árvores, sem profundidade máxima, número de *features* igual à raíz do número de *features*, mínimo de 3 amostras para criação de um *split*, mínimo de 1 amostra em cada folha e sem utilização de *bootstrap*. Essa combinação atingiu precisão de 91,7% com uma divisão do *dataset* de 3 amostras de treino para 1 amostras de teste.

```
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 50, stop = 500,
    ↪ num = 10)]
# Number of features to consider at every split
max_features = ['auto', 'log2', None]
# Number of samples considered
max_samples = [int(x) for x in np.linspace(start = 10, stop = 100, num
    ↪  = 10)]
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(1, 10, num = 10)]
max_depth.append(None)
# Minimum number of samples required to split a node
```

```
min_samples_split = [2, 5, 10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4]
# Method of selecting samples for training each tree
bootstrap = [True, False]

# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_samples': max_samples,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}
```

Para o modelo de *linear SVM*, chamado de *linear support vector classifier* ou *LinearSVC* no *sklearn* , os hiperparâmetros a serem considerados foram: penalidade, função de perda e parâmetro de regularização, o C. Aplicando os métodos de busca, a combinação encontrada foi penalidade 'l2', que é a padrão, função de perda *hinge*, que também é padrão, e parâmetro de regularização igual a 10. A precisão atingida foi de 97,7% com a divisão de 3 para 1 entre dados de treino e de teste.

```
# Norm used in the penalization
penalty = ['l1', 'l2']
# Loss function
loss = ['hinge', 'squared_hinge']
# Regularization Parameter
C = [0.1, 1, 10, 100, 1000]

# Create the random grid
random_grid = {'C': C,
               'penalty': penalty,
               'loss': loss}
```

Para o modelo de *multinomial naive bayes*, o único hiperparâmetro a ser considerado é o alpha. O melhor valor de alpha encontrado para os dados disponíveis foi 0,05. Essa

configuração atingiu uma precisão de 93,1% com mesma divisão entre dados de treino e teste dos modelos anteriores.

Finalmente, para o *linear regression* foi utilizado o *LogisticRegressionCV* do *sklearn*. Esse modelo já seleciona os melhor valores de hiperparâmetro por validação cruzada. Assim, só foi necessário aplicar o treinamento no modelo. A precisão atingida foi de 92,1% com a divisão 3 para 1 de dados de treino e teste.

Ao comparar a precisão resultante dos 4 modelos, o *linear SVM* teve o melhor desempenho e foi o modelo utilizado no projeto para a classificação de notícias. O modelo treinado apresentou desempenho parecido para classificação de notícias verdadeiras e falsas. A figura a seguir detalha o desempenho do *linear SVM* para cada rótulo.

	precision	recall	f1-score	support
FAKE	0.977	0.976	0.977	578
TRUE	0.976	0.978	0.977	586

Figura 20: Valor de *precision*, *recall* e *f1-score* e a quantidade de exemplos para cada rótulo.

Para a *skill* de *fake news* no *Watson Assistant*, a única itenção criada foi a de *classify news*. Em *dialog*, foi criado um nó com 3 nós filhos. A condição de entrada do primeiro é a identificação da intenção *classify news* e ele pede para o usuário escrever o título da notícia. O segundo nó guarda o título e pede o corpo da notícia. O terceiro nó guarda o corpo da notícia e atribui o valor "agent"para a variável de contexto *destination bot*. O último nó manda o título, corpo e intenção como um dicionário para o *webhook* configurado e aguarda a resposta para responder o usuário.

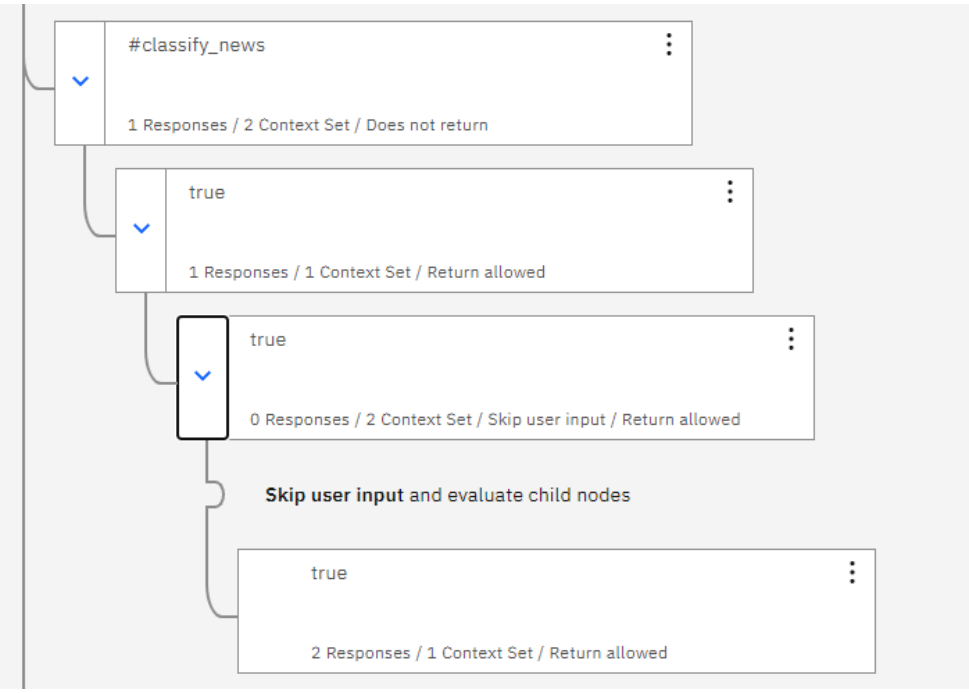

Figura 21: Nó de diálogo para a classificação de notícias.

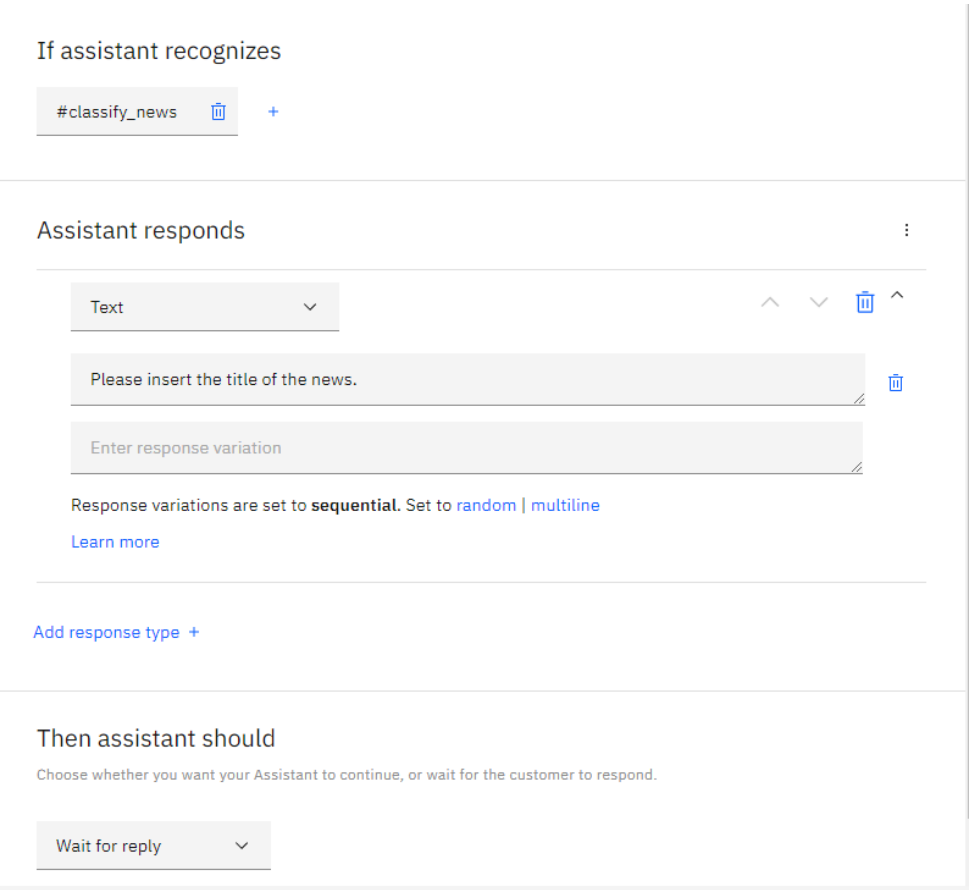

Figura 22: Primeiro nó para a classificação de notícias.

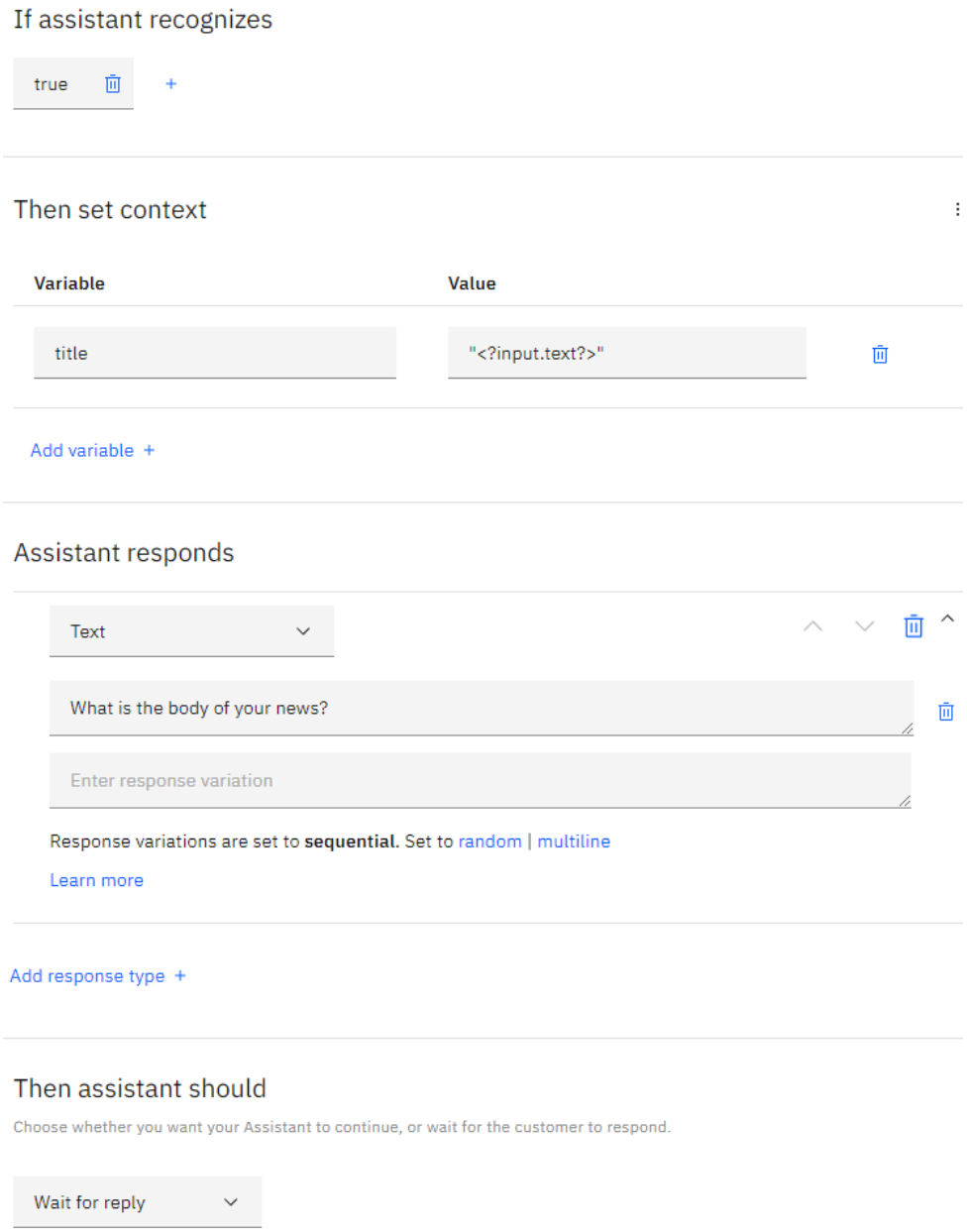

Figura 23: Segundo nó para a classificação de notícias.

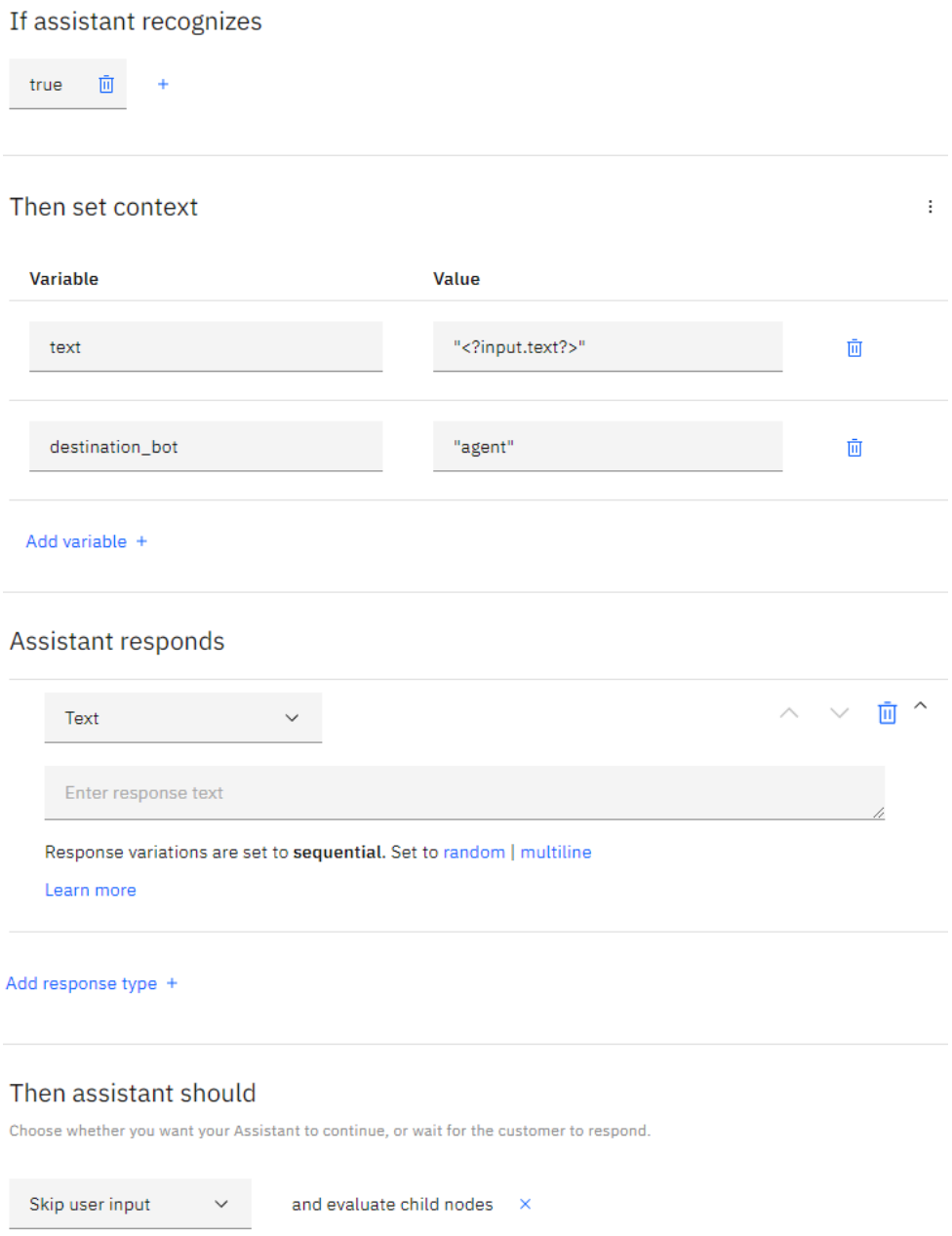

Figura 24: Terceiro nó para a classificação de notícias.

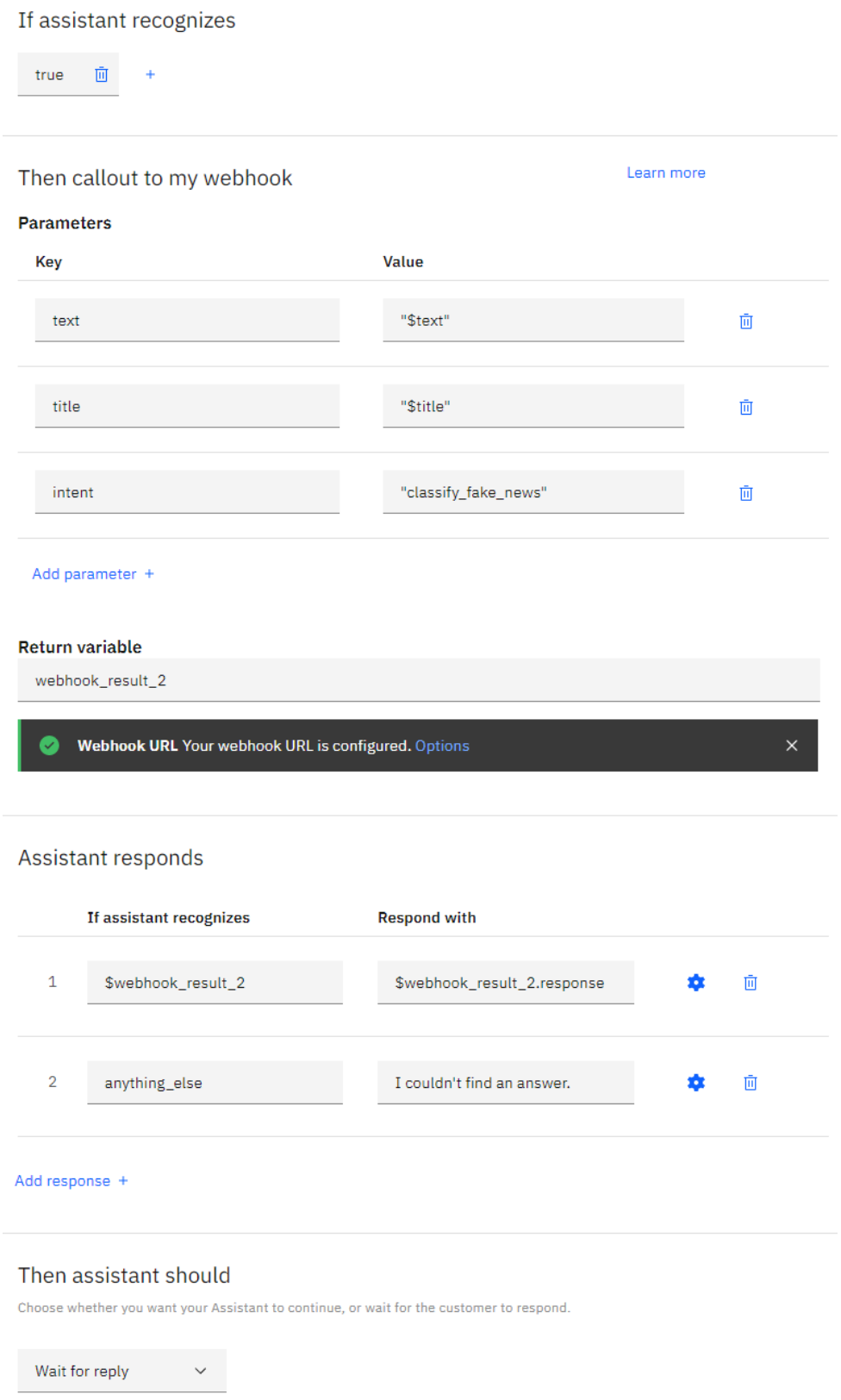

Figura 25: Quarto nó para a classificação de notícias.

Com o classificador, intenção e diálogo prontos, a última etapa é configurar o *webhook*. A *url* do *webhook* deve ser configurada no *menu* opções da *skill*. Para que uma *url* pública

direcione para o programa de classificação em *Python*. Foi utilizado uma combinação de *flask*, que permite que um programa escute requisições HTTP em uma porta local, e ngrok, que faz uma ponte entre uma *url* pública e uma porta local.

Para testar o funcionamento foi utilizado o console de desenvolvimento.

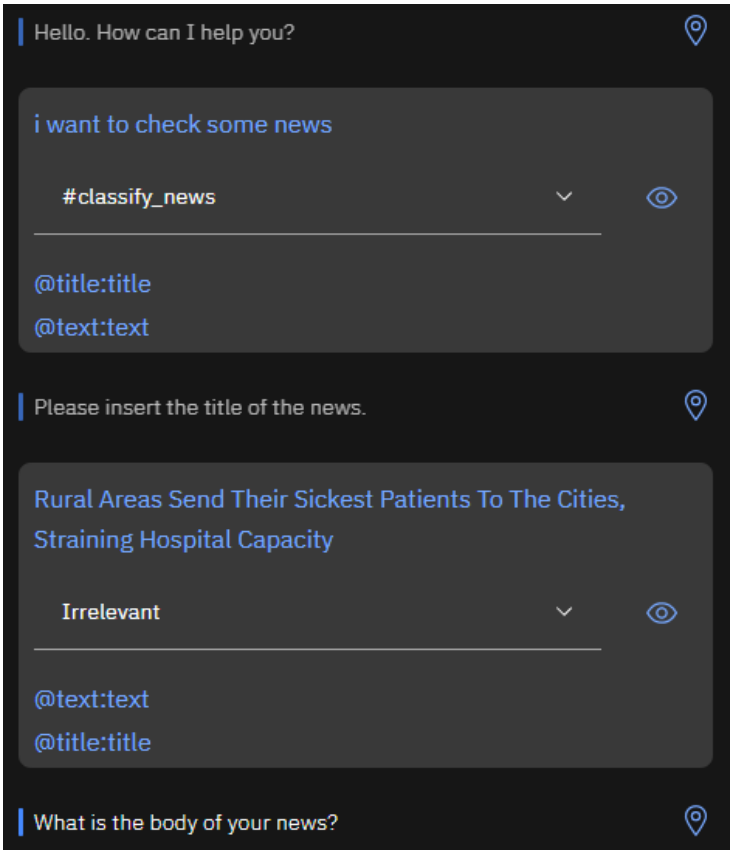

Figura 26: Conversa no console de desenvolvimento mostrando os dois primeiros nós do diálogo sobre classificação de *fake news*.

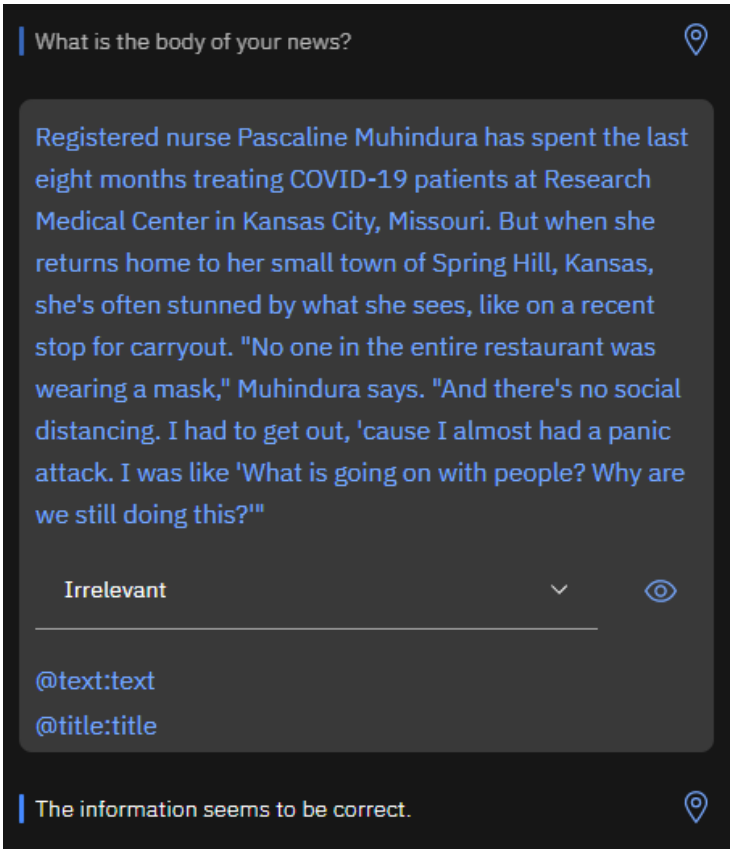

Figura 27: Conversa no console de desenvolvimento mostrando os dois últimos nós do diálogo sobre classificação de *fake news*.

7.3 Validação

O método de validação escolhido para avaliar o sucesso do projeto foi disponibilizar o *chatbot* para voluntários seguido de um formulário com diversas perguntas sobre a experiência. Os pontos mais importantes foram a sua capacidade educacional, a sua usabilidade e a coerência das respostas. Também foi avaliado cada *skill* do *chatbot*.

O formulário possui perguntas abertas e fechadas. Entre as perguntas fechadas, foi pedido para o usuário avaliar de 1 a 5 se o *chatbot* trouxe alguma informação nova, a coerência das respostas e a facilidade de utilizá-lo. No outro tipo de pergunta fechada, foi pedido para o voluntário selecionar quais *skills* foram utilizadas e quais delas foram experiências positivas. Cada pergunta fechada foi acompanhada de uma pergunta aberta para que o usuário pudesse descrever com mais detalhes o motivo da resposta.

PARTE IV

RESULTADOS

8 RESULTADOS

O *chatbot* foi disponibilizado para 13 pessoas. A primeira pergunta foi se o *chatbot* trouxe alguma informação nova para o usuário. Das respostas recebidas, 9 foram positivas, 3 foram neutras e 1 foi negativa . Dos usuários, 10 mencionaram ter aprendido com a *skill* de *Recorded Data*, 4 mencionaram ter aprendido com a *skill* de *FAQ* e 1 mencionou a *skill* de *Fake News*. As avaliações recebidas estão na figura 28.

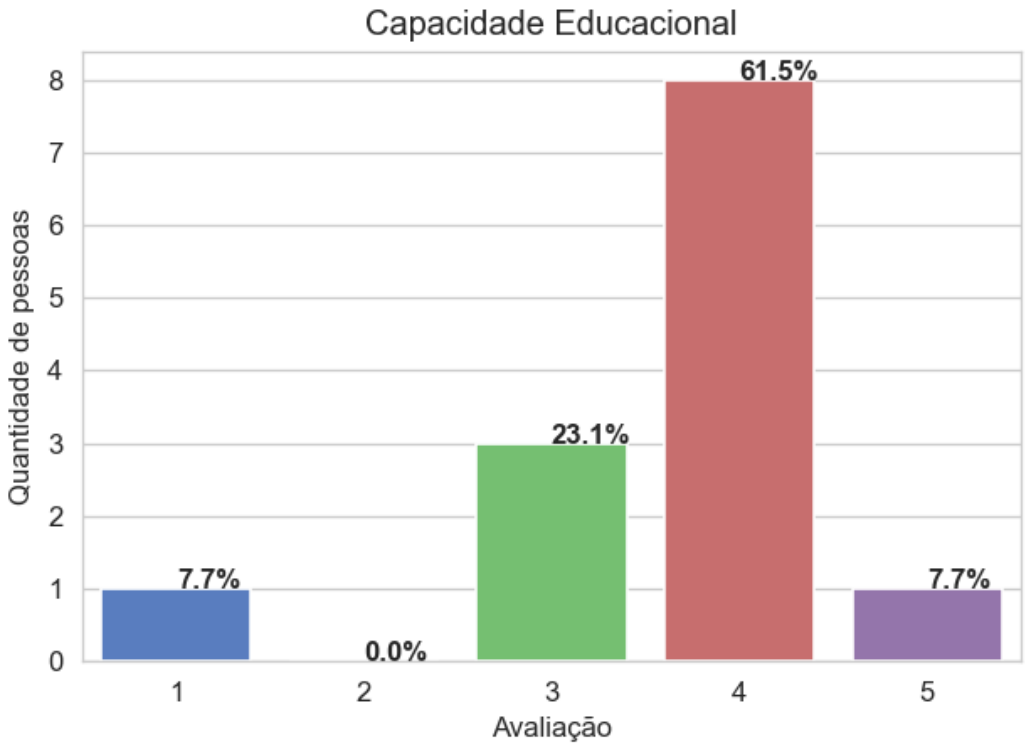

Figura 28: Resultado da pesquisa sobre capacidade educacional do *CovidBot*.

A segunda pergunta foi o quão condizente foram as respostas para as perguntas. Das respostas, 9 foram positivas, 3 foram neutras e 1 foi negativa. Todos os voluntários mencionaram pelo menos uma pergunta em que o *chatbot* teve dificuldade de compreensão dela e não conseguiu respondê-la ou respondeu uma pergunta diferente. Entre as perguntas que não foram respondidas com coerência foram registradas: "When is the pandemic goind

to end?", "Which country dealt best with the pandemic?", "What is the letality rate?", "What is lockdown?"e "How many vaccines COVID has?". As avaliações recebidas estão na figura 29.

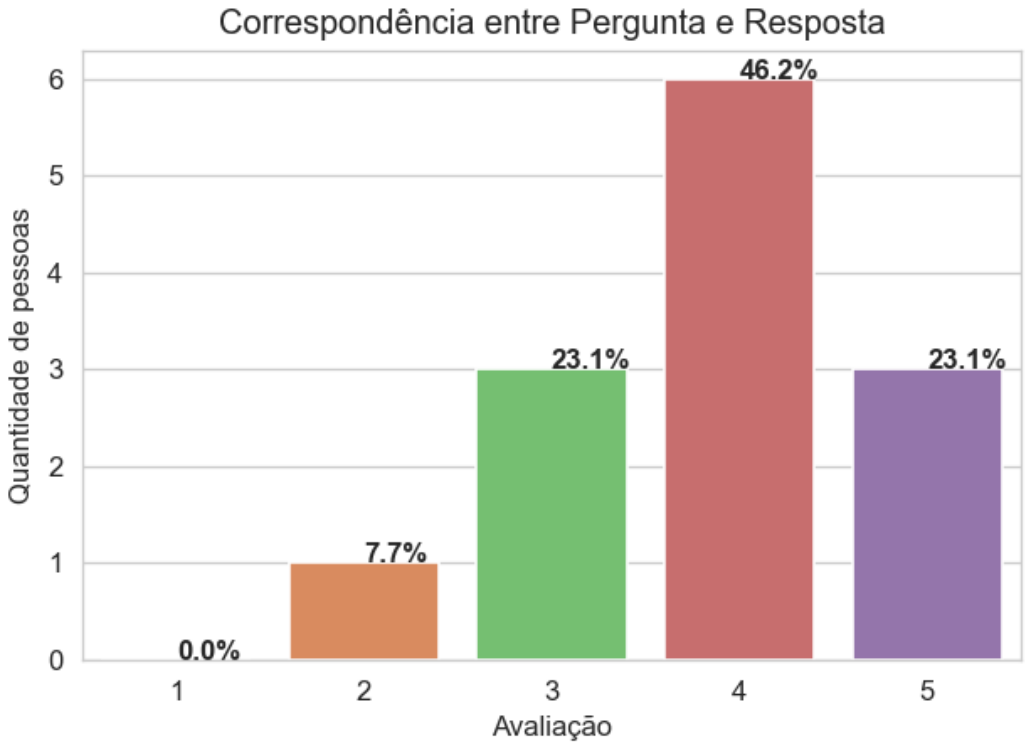

Figura 29: Resultado da pesquisa sobre correspondência entre perguntas e respostas.

A terceira pergunta foi o quão fácil foi conversar com o *chatbot*. Das respostas, 9 foram positivas, 3 foram neutras e 1 foi negativa. Foi mencionado que o *chatbot* foi fácil de usar e as respostas foram objetivas. As avaliações recebidas estão na figura 30.

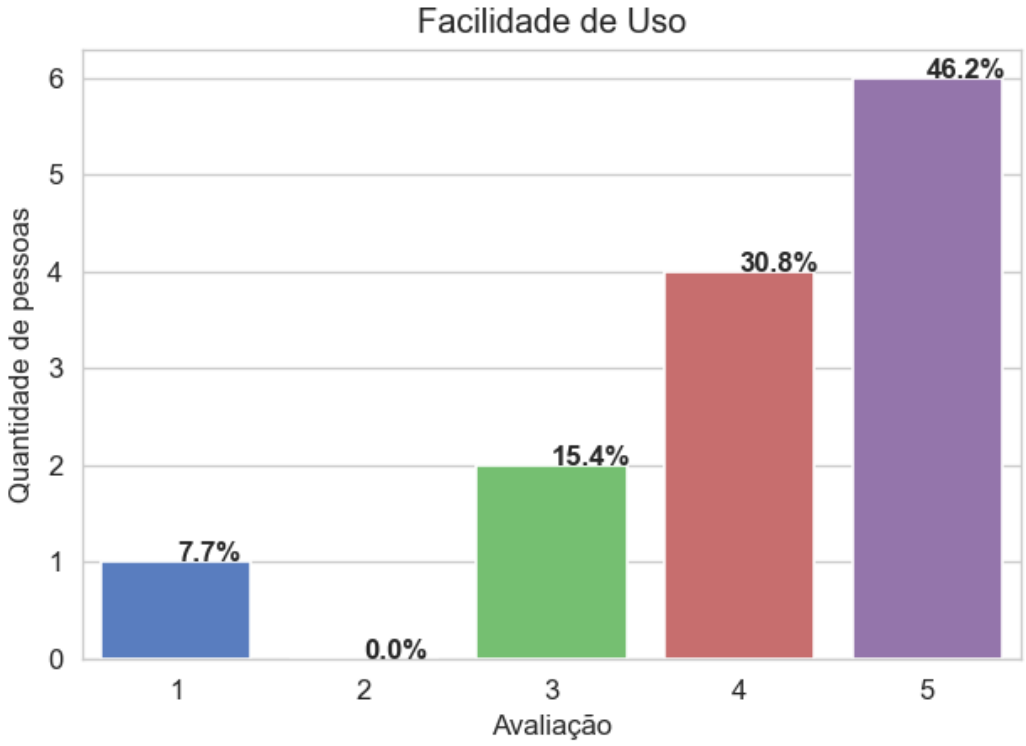

Figura 30: Resultado da pesquisa sobre facilidade de uso do *CovidBot*.

As perguntas quatro e cinco foram relacionadas. A quatro pede para indicar quais foram as *skills* que o usuário conseguiu utilizar e a cinco pede para indicar quais delas proporcionaram uma experiência positiva. Para a *skill* de *Personality*, 10 pessoas conseguiram utilizá-la e todas descreveram a experiência como positiva. Para a *skill* de *FAQ*, 12 conseguiram utilizá-la e 8 descreveram a experiência como positiva. Para a *skill* de *Recorded Data*, 13 conseguiram utilizá-la e 12 descreveram a experiência como positiva. Para a *skill* de *Fake News*, 10 conseguiram utilizá-la e 7 descreveram a experiência como positiva. Alguns dos comentários recebidos foram de que foi necessário entender como estruturar a pergunta e que as respostas foram feitas para responder perguntas pontuais. Os resultados estão nas figuras 31 e 32

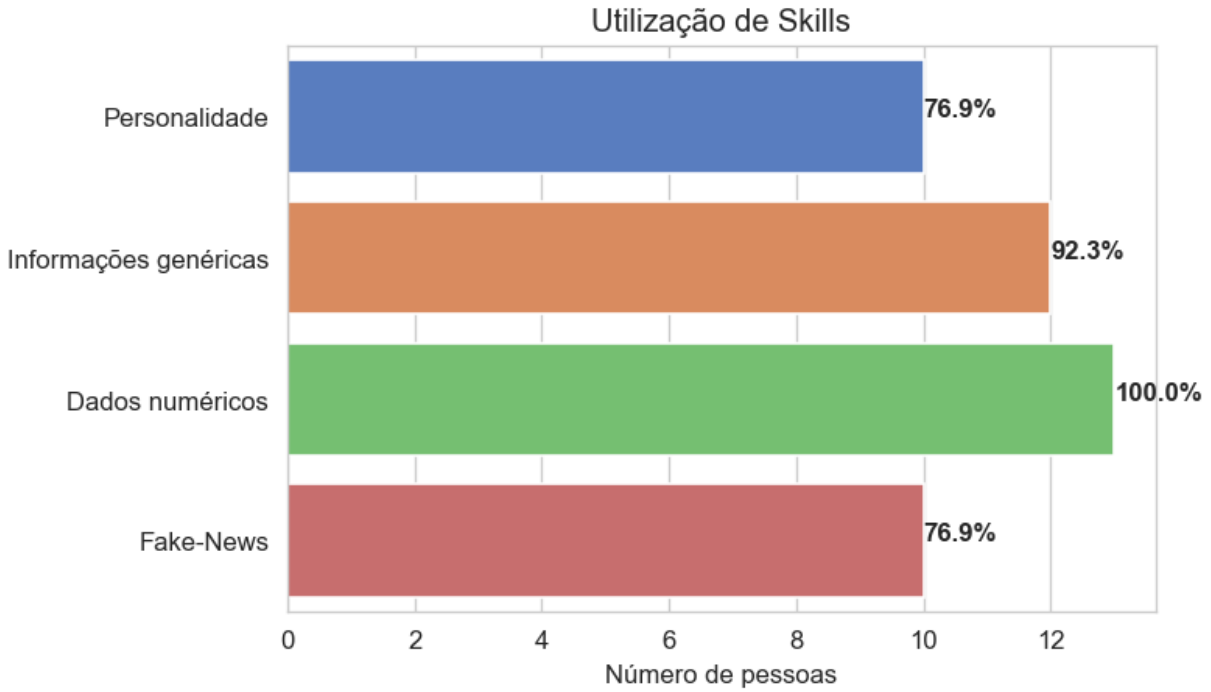

Figura 31: Resultado da pesquisa sobre utilização das *skills* do *CovidBot*.

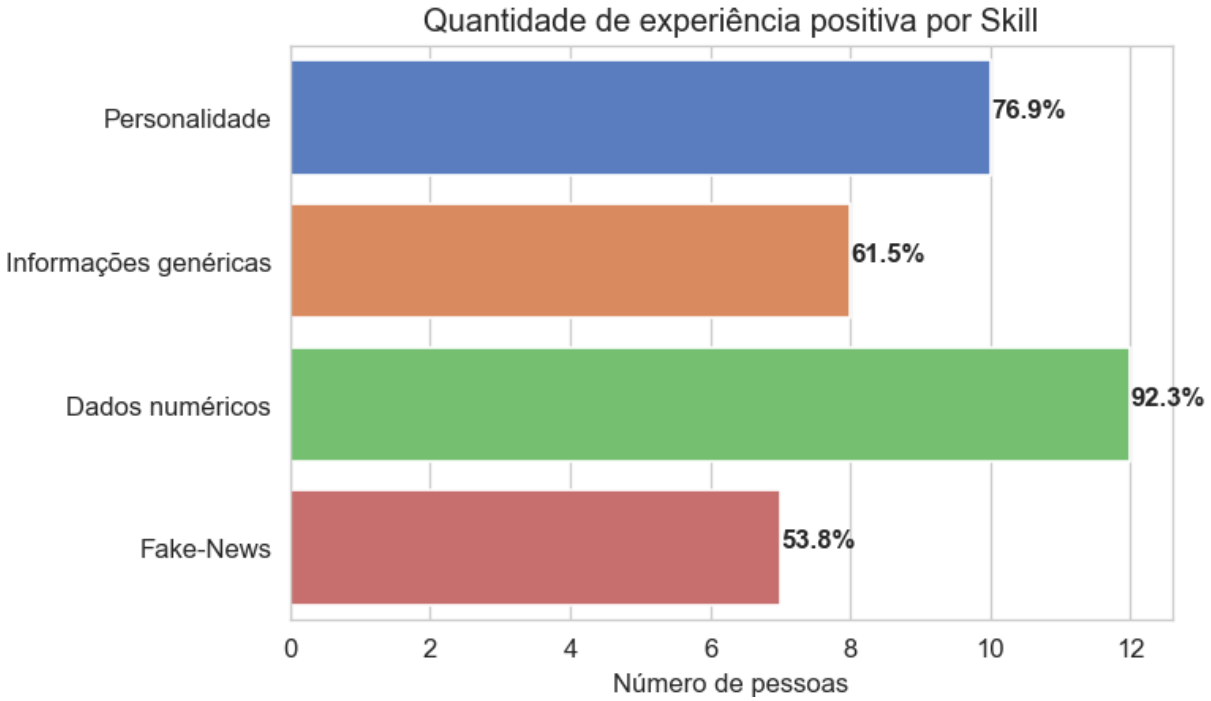

Figura 32: Resultado da pesquisa sobre a experiência do usuário com relação ao *CovidBot*.

A última pergunta foi se o usuário utilizaria o *chatbot* novamente. A figura 33 mostra as respostas recebidas.

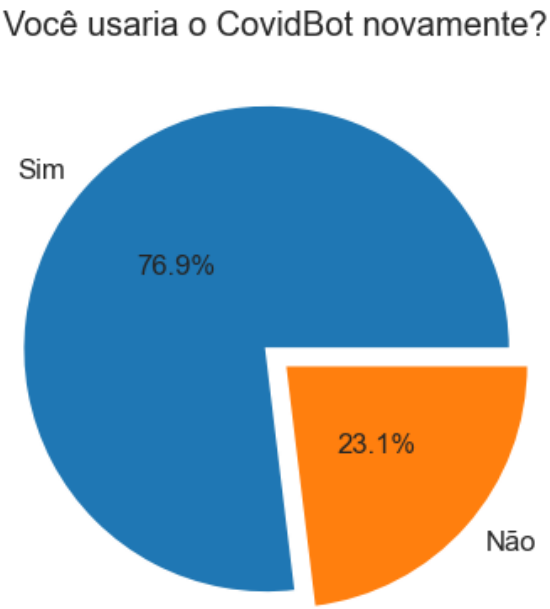

Figura 33: Conversa no console de desenvolvimento mostrando os dois últimos nós do diálogo sobre classificação de *fake news*.

9 DISCUSSÃO

Com as respostas de campo aberto para a primeira pergunta, foi observado que a maioria das pessoas aprenderam com a *skill* de dados numéricos de casos e mortalidades. Isso pode ser consequência do fato de esses dados mudarem diariamente durante a pandemia, o que cria a necessidade constante de acompanhá-los. O *chatbot* se apresentou como uma ferramenta para realizar esses acompanhamentos. A *skill* de *FAQ* também mostrou potencial de trazer informações novas para o usuário. Um ponto mencionado foi que o escopo das perguntas dessa *skill* foi limitado, o que pode explicar a quantidade menor de usuários que aprenderam com essa funcionalidade. A *skill* de *Fake News* teve baixa capacidade de trazer informações novas para o usuário, com menção por apenas um deles. A *skill* de *Personality* não tinha objetivo educacional, então o fato de nenhum usuário ter descrito ela como capaz de trazer informações novas era esperado.

Sobre os resultados da segunda pergunta, foi possível notar que várias perguntas relacionadas ao COVID-19 não conseguiram ser respondidas pelo *chatbot*. Um dos motivos foi que os usuários perguntaram sobre informações que não foram consideradas para a *FAQ*, como "What is lockdown?"e "How many vaccines does COVID have?". Um outro motivo para perguntas que não tiveram respostas coerentes foi a falta de treinamento para responder perguntas mais genéricas como "What should I do if I have symptoms?". Os dois problemas podem ser resolvidos com uma maior quantidade de informações de *FAQs*, notícias e artigos, ou com um refinamento maior das frases de exemplo para cada intenção. O primeiro foco com a *skill* de *FAQ* foi responder as perguntas mais comuns que servem de orientação no momento inicial da contaminação na pandemia. Isso permitiu menor preocupação em manter as informações atualizadas com a evolução dos estudos sobre o COVID-19 e da pandemia. A segunda prioridade foi de fornecer informações confiáveis ao usuário. Por isso, a abrangência das perguntas e respostas da *skill* de *FAQ* foi limitada.

As respostas da terceira pergunta mostram que o *chatbot* apresentou facilidade de uso. Essa foi a características com avaliações mais altas. Isso mostra que a possibilidade de conversa em linguagem natural melhorou a experiência do usuário.

Sobre as respostas das peguntas quatro e cinco, as *skills* de *Personality* e de *Recorded Data* tiveram sucesso ao proporcionar experiências positivas para os usuários que interagiram com elas. Já as *skills* de *FAQ* e *Fake News* apresentaram uma diferença maior entre a quantidade de usuário que as utilizaram e a quantidade deles que gostou da experiência. Isso pode ser sinal de que essas funcionalidades são mais limitadas do que esperado pelos usuários. Como mencionado anteriormente, a limitação do escopo das perguntas e respostas para a *skill* de *FAQ* foi intencional. As limitações da *skill* de *Fake News* consistem em dois fatores. O primeiro é a necessidade de ter uma notícia disponível e de extrair manualmente o título e corpo dela. O segundo é a resposta ser apenas uma afirmação de que a notícia pode ser falsa ou não.

A última pergunta foi proposta para avaliar o *chatbot* como um todo. O resultado de que 10 dos 13 voluntários gostariam de usar o *chatbot* novamente mostra que há interesse em *chatbots* informativos.

10 CONCLUSÃO

Diante dos resultados e da discussão apresentada, conclui-se que o *CovidBot* é uma ferramenta promissora para fornecer informações e estatísticas básicas sobre a COVID-19. A capacidade educacional, coerência das respostas e facilidade do uso do *bot* receberam avaliações positivas, com destaque para a terceira característica. Com esse resultado, foi atingido o objetivo de criar um *chatbot* acessível ao público que informe sobre a COVID-19.

Sobre as *skills* desenvolvidas, a de *Personality* e de *Recorded Data* se mostraram promissoras e bem sucedidas com os usuários. Em relação às *skills* de *FAQ* e *Fake News* deve-se avaliar como melhorar a recepção dos usuários para essas funcionalidades.

REFERÊNCIAS

[1] FACEBOOK. *Facebook Business - Casos de Sucesso*. 2020. Accessed: 2010-03-16. Disponível em: <https://www.facebook.com/business/success/categories/messenger>.

[2] RAJ, S. *Building Chatbots with Python: Using Natural Language Processing and Machine Learning*. [S.l.]: Apress, 2020.

[3] WEI, C.; YU, Z.; FONG, S. How to build a Chatbot: Chatbot framework and its capabilities. In: *ACM International Conference Proceeding Series*. [S.l.]: Association for Computing Machinery, 2018. p. 369–373. ISBN 9781450363532.

[4] HIRSCHBERG, J.; MANNING, C. D. Advances in natural language processing. *Science*, American Association for the Advancement of Science (AAAS), v. 349, n. 6245, p. 261–266, 2015. ISSN 0036-8075.

[5] CLARK, L. et al. What Makes a Good Conversation? In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems - CHI '19*. [S.l.]: ACM Press, 2019.

[6] LLC, G. *Dialogflow Documentation*. 2020. Accessed: 2020-03-16. Disponível em: <https://cloud.google.com/dialogflow/docs>.

[7] CLOUD, I. *Watson Assistant IBM Cloud Service*. 2020. Accessed: 2020-11-14. Disponível em: <https://cloud.ibm.com/docs/assistant?topic=assistant-getting-started>.

[8] CENTER, P. R. *Many Americans Believe Fake News Is Sowing Confusion*. 2016. Accessed: 2020-04-10. Disponível em: <https://www.journalism.org/2016/12/15/many-americans-believe-fake-news-is-sowing-confusion/>.

[9] DERAKHSHAN H., W. C. Information disorder: Toward an interdisciplinary framework for research and policy making. *Council of Europe*, 2017.

[10] DUYN E., C. J. V. Priming and fake news: The effects of elite discourse on evaluations of news media. p. 29–48, 2018.

[11] CANTO, A. B. del. *covid-daily - COVID-19 Daily Data from Worldometers with Python*. GitHub, 2020. Disponível em: <https://github.com/alvarobartt/covid$_d$aily¿.

[12] PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011.

[13] CHAVAN, M. *watson-assistant-multi-bot-agent*. 2020. Accessed: 2020-10-3. Disponível em: <https://github.com/IBM/watson-assistant-multi-bot-agent>.

[14] MEDICINE, J. H. *Coronavirus Symptoms: Frequently Asked Questions*. 2020. Accessed: 2020-10-31. Disponível em: <https://www.hopkinsmedicine.org/health/conditions-and-diseases/coronavirus/coronavirus-symptoms-frequently-asked-questions>.

[15] UNIDAS, O. das N. *General FAQs*. 2020. Accessed: 2020-10-31. Disponível em: <https://www.un.org/en/coronavirus/covid-19-faqs>.

[16] SAúDE, O. M. de. *QAs on COVID-19 and related health topics*. 2020. Accessed: 2020-10-31. Disponível em: <https://www.who.int/emergencies/diseases/novel-coronavirus-2019/question-and-answers-hub>.

[17] LI, S. *COVID Fake News Detection with a Very Simple Logistic Regression*. 2020. Accessed: 2020-11-02. Disponível em: <https://towardsdatascience.com/covid-fake-news-detection-with-a-very-simple-logistic-regression-34c63502e33b>.