http://journal.unoeste.br/index.php/ce/index
DOI: 10.5747/ce.2021.v13.n3.e369
ISSN on-line 2178-8332

Colloquium Exactarum

**Submetido: 29/06/2021 Revisado: 24/09/2021 Aceito:14/10/2021**

# EASY ASSISTANT: UMA FERRAMENTA DE AUXÍLIO PARA O DESENVOLVIMENTO DE CHATBOTS DE DOMÍNIO ESPECÍFICO

## EASY ASSISTANT: A help tool for specific domain chatbots development

Alexandre Moreira, Mário Augusto Pazoti, Robson Augusto Siscoutto

[1] Universidade do Oeste Paulista – UNOESTE, Faculdade de Informática de Presidente Prudente - FIPP, Presidente Prudente, sp.
E-Mail: alexandre_lopees@hotmail.com, mario@unoeste.br, robson@unoeste.br

**RESUMO –** Sistemas inteligentes estão ligados principalmente ao ato de trazer cada vez mais suporte às pessoas. Atualmente nota-se o uso crescente de elementos relacionados à capacidade de aprendizado de máquinas, ou seja, Inteligência Artificial (IA). Em IA, destacam-se os chatbots, agentes personalizados e virtuais com certo nível emocional e cognitivo, normalmente presentes em sistemas colaborativos, sociais ou de aprendizagem, a fim de oferecer algum serviço com interação persistente e fortemente relacional aos usuários. Desenvolver tal agente não é trivial pois deve fornecer uma comunicação consistente e imersiva, aplicando várias técnicas complexas de processamento, compreensão e geração de linguagem natural, bem como classificação de entidades, objetivos, auto recuperarão para evitar inconsistências de conversa e perda de contexto, dentre outros. Diante disso, este trabalho apresenta uma ferramenta web que permite ao usuário desenvolver, treinar e personalizar seu agente virtual de chatbot. Este agente faz uso de dois modelos de treinamento e obtenção de respostas baseado nos modelos de recuperação e generativo, construídos sob redes neurais que manipulam uma base de dados definida pelo usuário. A ferramenta foi testada e avaliada visando qualificar sua acurácia, a qual demonstrou atingir seus objetivos, bem como, permitir a criação de um chatbot personalizado.
**Palavras-chave:** Inteligência Artificial, Agentes Virtuais, Agentes Inteligentes.

**ABSTRACT –** Intelligent systems are mainly linked to the act of bringing more and more support to people. Currently, there is a growing use of elements related to the machine learning capacity, that is, Artificial Intelligence (AI). In AI, chatbots stand out, personalized and virtual agents with a certain emotional and cognitive level, usually employed in collaborative, social or learning systems, in order to offer some service with persistent and strongly relational interaction to users. Developing such an agent is not trivial. A chatbot that provides consistent and immersive communication requires several techniques, sometimes even complex,related to natural language processing, understanding and generation, which involve classification of entities, objectives, among others; also self-healing to avoid conversation inconsistencies and loss of context. Therefore, this work presents a web tool that allows the user to develop, train and customize their chatbot virtual agent. This agent uses two training and response models based on recovery and generative models, built on neurais networks that manipulate a database

defined by the user. The tool was tested and evaluated in order to qualify its accuracy, which proved to reach its goals, as well as allow the creation of a personalized chatbot.

**Keywords:** Chatbot, Chatbot Development, Artificial Intelligence, Virtual Agents, Intellectual Agents.

## 1. INTRODUÇÃO

Por meio da Inteligência Artificial (IA), pesquisadores buscam desenvolver tecnologias capazes de simular os pensamentos e comportamentos humanos, envolvendo o raciocínio, tomada de decisões, percepção e uma alta capacidade de solucionar problemas (ZHANG; THALMAN; ZHENG, 2016). O processamento e interpretação de linguagens naturais, aprendizado de máquina e aplicação de redes neurais para habilidades cognitivas são subáreas que estimulam o uso de recursos de interatividade entre homem e máquina (RUSSAN, 2014).

Dentro deste contexto, o uso de chatbots como uma plataforma em que há um agente virtual de aspecto personalizado já é uma realidade. Tal agente possui certo poder emocional e cognitivo em que, muitas vezes, são empregados em sistemas colaborativos, sociais ou de aprendizagem, cujo foco é proporcionar uma interação persistente e fortemente relacional (SVIATLANA; BUSEMANN; SCHOMMER, 2012).

O uso de chatbots por empresas, para um pré-atendimento ao consumidor em seus canais digitais, já está altamente presente no mundo dos negócios empresariais, devido aos seus benefícios, conforme indicado por Singh, Ramasubramanian e Shivam (2019). Além disso, o seu uso no entretenimento, em que se dialoga com internautas de forma livre ou em um jogo de perguntas e respostas, já é uma realidade.

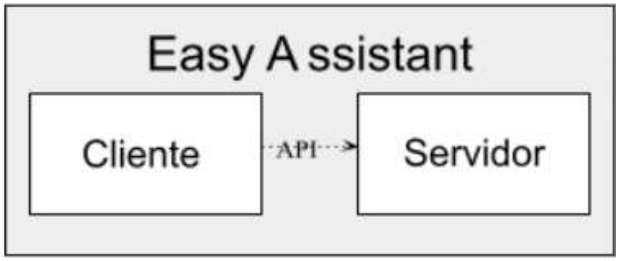
Segundo Cahn (2017), o desenvolvimento de um chatbot envolve classificar o nível de conhecimento sobre um domínio específico ou ilimitado, passando por tópicos particulares ou ter vastos conceitos e assuntos. Outro ponto crucial indicado por Lokman e Ameedeen (2019) é a escolha do modelo para a geração de respostas, que vão desde Modelos Baseados em Recuperação (recuperação de respostas pré-definidas em uma base de conhecimento) a Modelos Generativos (respostas geradas de acordo com um esquema de rede neural e tecnologias afins).

A criação de chatbots envolve questões técnicas, que dificultam que qualquer pessoa possa construir seu próprio agente ou bot. Além disso, a criação do bot ficaria à mercê das capacidades de uma terceira pessoa, podendo sofrer inconsistência ou desvio, bem como o tempo alto gasto pelo profissional responsável pelo desenvolvimento, em compreender o tópico específico fornecido pelo cliente, o que pode, por consequência, não atender verdadeiramente a necessidade do cliente. Tal panorama demonstra um ponto a ser explorado, em que um indivíduo de baixo conhecimento no assunto possa desenvolver seu próprio chatbot.

Dentro deste contexto, este artigo apresenta a ferramenta Easy Assistant, que permite que indivíduos construam seu próprio agente chatbot com suas próprias características de interesse e mantendo a abrangência sobre tópicos específicos, abstraindo, desse modo, o desenvolvimento de chatbots e possibilitando, em um formato compreensível, englobar as principais características relacionadas à arquitetura de chatbots.

Na Figura 1 são mostrados os módulos do Easy Assistant e sua relação. Basicamente, o Easy Assistant é uma ferramenta composta pelos lados cliente e servidor, que se comunicam por meio de uma interface de programação de aplicação (API). No lado cliente é possível visualizar e manipular as informações que são internamente processadas no lado servidor.

**Figura 1.** Easy Assistant, módulo Cliente e servidor.




Fonte: Os autores.

As principais contribuições apresentadas neste artigo são:

- Um motor como arquitetura para desenvolvimento de chatbot via servidor pelo usuário;
- Uma aplicação para estruturação e manipulação do chatbot a partir de um diagrama de fluxo.

O restante deste artigo está organizado da seguinte forma: Na Seção 2 são apresentados trabalhos relacionados sobre o assunto; na seção 3 são descritos detalhes da ferramenta Easy Assistant referentes a arquitetura, seja do lado

servidor como também do lado cliente. Ele também descreve os testes de performance que foram realizados e os resultados obtidos; e, finalmente, as conclusões e trabalhos futuros deste trabalho são apresentados na Seção 4.

## 2. TRABALHO RELACIONADOS

Alguns trabalhos relacionados ao desenvolvimento de chatbots que buscam aprimorar a capacidade de comunicação dos mesmos são apresentados a seguir.

Dhingra *et al.* (2017) propõem um agente de diálogo de turno múltiplo como auxílio a pesquisas em Base de Conhecimento (BD) de forma flexível. É adotado um método de busca probabilística (baseado em crenças levantadas de acordo com entidades) para calcular a distribuição posterior do objetivo do usuário em uma base de conhecimento. Os autores comprovaram que tal recurso ajuda o agente a descobrir melhores políticas de diálogo, além de permitir o treinamento ponta a ponta das políticas de diálogo em configuração on-line.

Liu *et al.* (2018) propõe uma combinação de políticas de treinamento de chatbots de domínio específico, gerando um modelo híbrido de imitação e reforço. No estudo é definido um agente inicialmente treinado através da política de treinamento supervisionado (diálogos corpora); e, posteriormente, por meio da interação com usuários reais, o agente adquire conhecimento de acordo com o *feedback* obtido durante a conversa.

Song *et al.* (2018) propuseram a combinação dos modelos de recuperação e generativo. Com o acréscimo de um mecanismo de cópia, com o intuito de extrair palavras que compõem as respostas recuperadas sob um nível de probabilidade. O modelo de ranqueamento leva em consideração a similaridade entre os termos, entidades, tópico, tamanho da sentença, entre outros.

Yang *et al.* (2019) desenvolveram um modelo de treino e predição combinando modelo baseado em recuperação e generativo. Basicamente, a partir de uma frase de entrada $e$ transmitida pelo modelo de recuperação, é extraído um rótulo de intenção que permite consultar respostas predefinidas na base de conhecimento, com isso, tem-se $n$ respostas candidatas $r$ que serão consideradas no modelo de geração. O modelo generativo recebe como parâmetros de entrada $e$ e $r$ , gerando uma frase única de resposta $g$. Nesse momento, as frases candidatas $r + g$ passam por uma análise de ranqueamento para classificar a melhor candidata como resposta final. Segundo os autores, a combinação dos dois métodos trouxe melhor performance ao modelo de conversação.

## 3. EASY ASSISTANT

A ferramenta Easy Assistant procura trazer uma forma simplificada de lidar com características básicas no desenvolvimento de chatbots. O intuito inicial é permitir a construção de chatbots de domínio específico, mantendo um escopo de pouca abrangência. Para tanto, a ferramenta faz uso de aspectos voltados para o desenvolvimento de chatbots e relacionados à Inteligência Artificial (redes neurais, classificação de texto, similaridade entre termos). Esses aspectos permitem definir e treinar um chatbot por meio da própria ferramenta, além da possibilidade de interação.

A Seção 3.1 indica características gerais das funcionalidades definidas como essenciais para a ferramenta. Já na Seção 3.2 é apresentado um panorama geral da arquitetura da ferramenta, incluindo as tecnologias utilizadas. Nas Seções 3.3, 3.4 e 3.5 são apresentados mais detalhes das funcionalidades indicadas na seção 3.1, respectivamente. Por fim, na seção 3.6, são dados detalhes sobre os testes e resultados obtidos.

### 3.1. Funcionalidades Essenciais do Easy Assistant

As funcionalidades essenciais da ferramenta, são:

a) Permitir o treinamento de uma base de dados, visando incorporar conhecimento ao chatbot e, dessa forma, dar a ele a capacidade de interpretar sentenças de entrada. Essa funcionalidade permite o treinamento do chatbot através da combinação de dois modelos de treino além de outras técnicas. Um dos modelos, baseado em recuperação, utiliza um classificador de texto que faz uso de rótulos de intenção previamente definidos na base de dados. O classificador é treinado levando em consideração os rótulos e alguns exemplos que se assemelham às sentenças de entrada do usuário. Outro modelo, baseado em geração, define uma rede neural treinada a partir de $n$ pares (texto de entrada e texto de saída alvejada) de sentenças.

b) Possibilitar a construção de um contexto de diálogo por meio de diagramas de fluxo. Com base nas intenções previamente definidas é possível mapear um diagrama que direciona o fluxo da conversa. Cada nó do diagrama está vinculado a uma intenção e possibilita o registro de respostas pré-definidas

que serão utilizadas para gerar a resposta final quando tal nó é identificado pelo chatbot. Os nós são interligados, hierarquicamente, direcionando o fluxo da conversa.

c) Permitir que o usuário possa fazer sugestões em casos de respostas inconsistentes. O usuário poderá avaliar a acurácia das respostas produzidas pelo chatbot a suas perguntas e, em caso de divergência, poderá sugerir respostas. A sugestão será combinada com o texto de entrada e será registrado na base de dados para posteriormente ser utilizado durante o treinamento do modelo generativo.

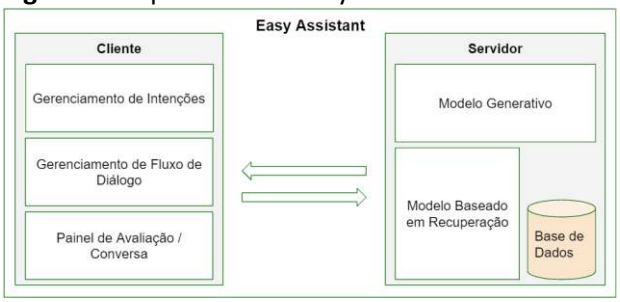
### 3.2. Arquitetura do Easy Assistant

Na Figura 2 é apresentada uma visão geral da arquitetura do Easy Assistant:

- O módulo "Cliente" trata do ambiente visível ao usuário para manipulação do chatbot. Nele estão presentes os cenários de modificação de rótulos de intenção e seus exemplos de entrada, bem como, a definição de fluxos de diálogos e um painel básico de bate-papo para avaliação do chatbot em treinamento.
- O módulo "Servidor" define as regras de funcionamento da ferramenta, basicamente, estão definidas nesta camada, chamadas para os modelos de treinamento e consultas ao chatbot. O módulo também engloba a base de dados que armazena informações sobre rótulos de intenção, exemplos de entradas, respostas predefinidas, conjunto de sugestões e fluxo de diálogo.

Com relação às tecnologias adotadas, no módulo "Servidor", foi utilizada, principalmente, a linguagem de programação Python (BORGES, 2014), incluindo recursos baseados em aprendizado de máquina, como é o caso das bibliotecas TensorFlow (ABADI *et al.*, 2016) e Keras (GULLI; PAL, 2017), bem como as bibliotecas para processamento de linguagem natural NLTK (*Natural Language Toolkit*) (LOPER; BIRD, 2002) e SpaCy (SRINIVASA-DESIKAN, 2018). Além disso, foi utilizada a biblioteca Flask (GRINBERG, 2018) para a construção de rotas de comunicação Web da API. O módulo "Cliente" faz uso de recursos da linguagem de marcação HTML5 (SILVA, 2019), em conjunto com a linguagem de estilos em cascata CSS (SILVA, 2019) e linguagem de programação JavaScript/React (BANKS, 2017), permitindo maior abrangência em sistemas computacionais.

**Figura 2.** Arquitetura do Easy Assistant.




Fonte: Os autores.

### 3.3. Treinamento e Incorporação de Conhecimento

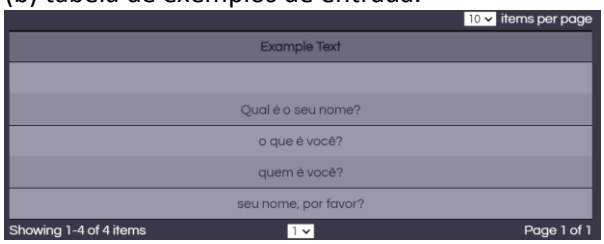
O usuário pode definir intenções e exemplos de entrada para reconhecimento de cada intenção. Duas tabelas personalizadas apresentadas na Figura 3 mostram as *tags* das intenções (Figura 3a) e os textos de entrada (Figura 3b), permitindo operações de consulta, inserção, alteração e exclusão. A segunda tabela exibe os textos de exemplo registrados para a *tag* selecionada na primeira tabela. Cada manipulação nessa tabela resulta em requisições ao servidor. Quando uma requisição é feita, se ela possui característica de alteração na base de dados, então, irá resultar em um treinamento paralelo do modelo baseado em recuperação.

**Figura 3.** Gerenciamento de Intenções.
(a) tabela de intenções.




Fonte: Captura no Easy Assistant pelos autores.

(b) tabela de exemplos de entrada.

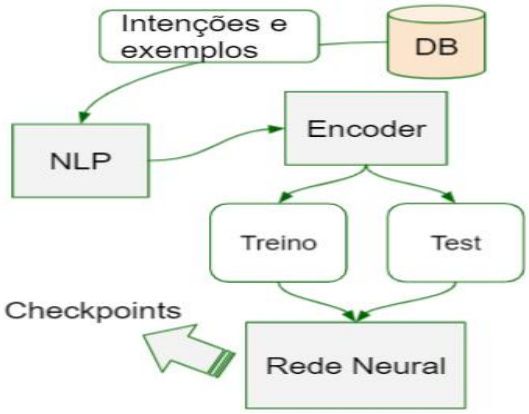



Fonte: Captura no Easy Assistant pelos autores.

O modelo baseado em recuperação foi implementado sob um modelo de rede neural LSTM (*Long short-term memory* ou memória de longo prazo) referente a uma RNN (*Recurrent neural network* ou rede neural recorrente) aprimorada, que busca manter seus estados ao longo do tempo (GREFF *et al.*, 2017; MIKOLOV *et al.*, 2010; GRAVES *et al.*, 2013; KALCHBRENNER *et al.*, 2013). Nesse modelo foi elaborado um classificador de texto fazendo uso da LSTM bidirecional, capaz de contribuir para a identificação de intenções com base em textos de entrada (ZHOU *et al.*, 2015).

Conforme pode ser visualizado na Figura 4, o processo de treinamento do modelo baseado em recuperação tem início com a recuperação dos dados de treino da base de dados. Tais dados são enviados para o processamento de linguagem natural (NLP) (NADKARNI *et al.*, 2011) que visa tratar a informação e facilitar o entendimento a nível computacional da linguagem natural. Na sua implementação foram utilizados recursos do NLTK e SpaCy, como tokenização e remoção de palavras de baixa relevância (*stopwords*). Em seguida, os dados são codificados (*encoder*) para valores inteiros com base em um vocabulário e na divisão em dois conjuntos (treino e teste) para dar continuidade às etapas de treino do classificador de texto. Cada etapa que tenha um nível de perda menor do que o já encontrado durante o treinamento, irá gerar um checkpoint para recuperação desse melhor estágio do modelo da rede neural.

**Figura 4.** Esquema geral do processo de treinamento do modelo baseado em recuperação.




Fonte: Os autores.

Enquanto todo o treinamento ocorre, é ideal que a ferramenta ainda esteja totalmente acessível ao usuário, sendo assim, caso ocorra uma nova manipulação que exija o treinamento do chatbot, enquanto um outro treino está ocorrendo, então o treino atual será interrompido e irá iniciar o novo treinamento. Para a interrupção basta apenas alterar um parâmetro de *flag* fornecido pela biblioteca TensorFlow.

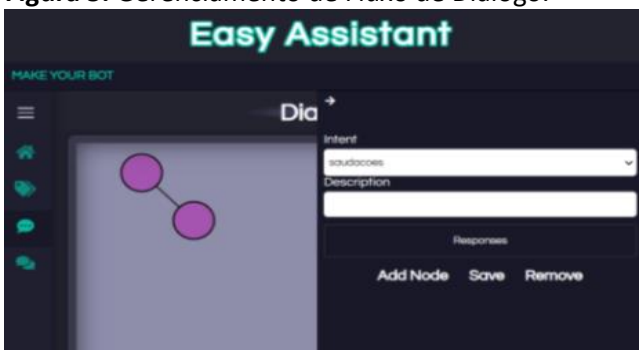
### 3.4. Fluxo de Diálogo

O fluxo de diálogo é definido de acordo com a esquematização do diagrama composto por nós interligados. A apresentação do diagrama é feita através do componente Canvas do HTML5.

A combinação de recursos do JavaScript/React e HTML5 permitiram que os dois tipos de elementos essenciais da representação, nó e ligação, sejam mapeados e desenhados para contribuir com a interação e manipulação do fluxo de diálogo. O componente básico "nó" é representado por círculos, enquanto que as ligações são representadas por linhas. Na Figura 5 é apresentado um exemplo do diagrama de Fluxo.

No painel expansível lateral da funcionalidade de fluxo de diálogo existe uma tabela personalizada para as respostas predefinidas e campos de entrada para definir a intenção e descrição de cada nó. Além disso, existem botões que permitem acrescentar e excluir nós do diagrama.

**Figura 5.** Gerenciamento de Fluxo de Diálogo.




Fonte: Captura no Easy Assistant pelos autores.

Fazendo uso do mouse diretamente no Canvas, é possível selecionar e arrastar os nós, além de interligá-los através de conectores. Cada nó permite o vínculo de uma intenção, além da inclusão de $n$ respostas predefinidas que serão recuperadas no momento de gerar a resposta final do modelo baseado em recuperação.

O usuário seleciona um dos nós existentes, fornece a informação para o novo nó (intenção e descrição) e o adiciona através do botão "Add Node", que pode ser visto na Figura 5. Nesse momento, um vínculo é criado entre o novo nó e o anteriormente selecionado. Se nenhum nó

existente foi selecionado antes de adicionar um novo nó, então esse novo nó será adicionado como o estado inicial de um fluxo (ponto de entrada).

A interligação entre nós é possível a partir do momento em que o botão esquerdo do mouse é mantido pressionado sobre um nó. Nesse momento, um conector se manterá visível, possibilitando arrastar o conector até o nó de destino.

Para a exclusão, basta selecionar o nó alvo e utilizar o botão “Remove”, isso fará com que as ligações existentes vinculadas ao nó recém-removido deixem de existir, inclusive o nó alvo.

### 3.5. Sugestão de Respostas

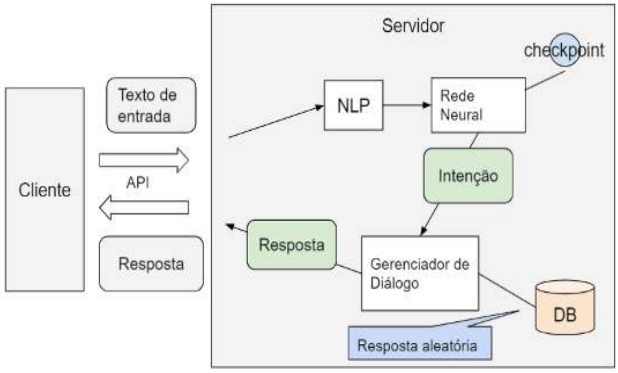
O Easy Assistant permite consultar o chatbot em treinamento, dessa forma é possível avaliar a acurácia das respostas. Basicamente, o recurso para esta avaliação se trata de uma representação de canal de bate-papo, salvo a possibilidade de definir o modo de conversa e também sugerir respostas.

O usuário pode escrever a mensagem em um campo de entrada de texto e encaminhá-la ao chatbot por meio do botão enviar. As mensagens são exibidas em um painel com rolagem para acompanhá-las.

Estão disponíveis dois modos de conversa: (1) segue o fluxo ativo definido pelo usuário e utiliza somente o modelo baseado em recuperação; (2) modo livre que mantém uma interação de “maior liberdade” fazendo uso do modelo baseado em recuperação e do modelo generativo.

Conforme a Figura 6, quando o modo de fluxo está ativo, cada mensagem do usuário (texto de entrada) é encaminhada ao servidor via API para consulta de uma resposta do chatbot. Chegando ao modelo baseado em recuperação, já no servidor, o primeiro passo é carregar o melhor modelo da rede neural classificador de texto (checkpoint definido durante o treino). Em seguida realiza a tratativa da mensagem do usuário através do NLP, passa pelo classificador de texto e obtém uma intenção. Posteriormente, com base em estados interligados (representação do fluxo de diálogo definido pelo usuário que fora apresentado na Subseção 3.4), será avaliado se a intenção reconhecida pertence a um estado subsequente do estado atual, se não for, irá gerar uma resposta que sugere a próxima mensagem que o usuário pode transmitir, caso contrário, irá simplesmente recuperar uma resposta aleatória predefinida, que é referente à intenção reconhecida, adotar o novo estado subsequente e retornar a resposta ao usuário.

**Figura 6.** Esquema geral da obtenção de resposta pelo modelo baseado em recuperação.

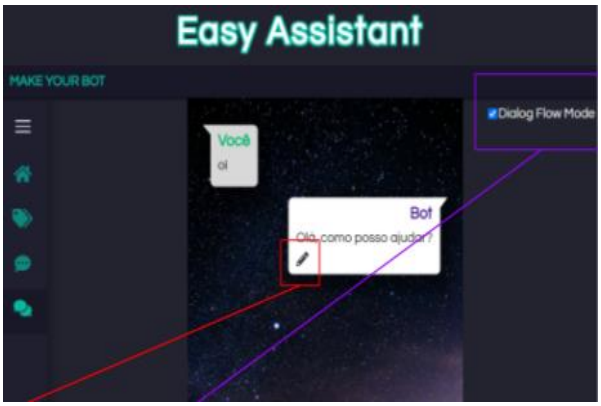



Fonte: Os autores.

Já no modo livre, o processo descrito no parágrafo anterior também ocorre, com a distinção de que a intenção não será avaliada pelos estados de fluxo e a resposta final gerada pelo modelo baseado em recuperação não é diretamente retornada ao usuário. Ainda será consultado o modelo generativo que, inicialmente, também carrega seu melhor modelo de rede neural. Em seguida, é feita a tratativa da mensagem de entrada, passando pela rede neural e gerando uma resposta. Ambas as respostas (predefinida recuperada e gerada automaticamente) passaram por um esquema de classificação para determinar qual delas tem maior relevância com a mensagem de entrada do usuário.

A sugestão de respostas surge a partir do entendimento do usuário de que a resposta final, gerada pelo chatbot, não foi adequada à mensagem de entrada. Então, abaixo da resposta existe um ícone em formato de lápis que habilita um campo para sugerir uma resposta, conforme Figura 7. A sugestão é combinada com a mensagem de entrada que gerou a possível resposta inconsistente. Essa combinação será registrada como um exemplo de diálogo para o treinamento do modelo generativo. A cada sugestão definida, o treinamento do modelo generativo será disparado.

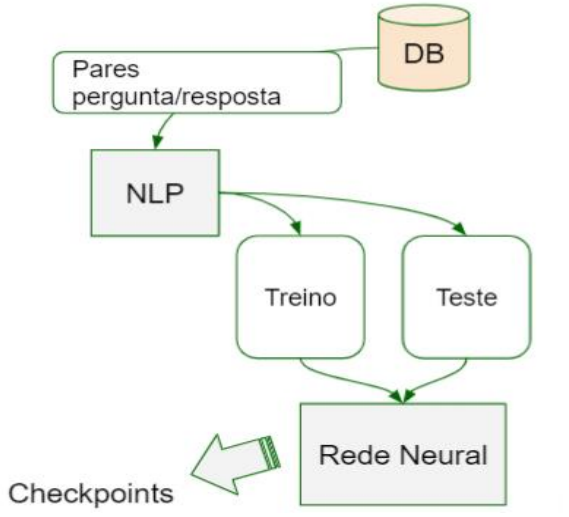
**Figura 7.** Visão geral do painel de avaliação do chatbot.




Fonte: Captura no Easy Assistant pelos autores.

O modelo generativo implementado é composto por um esquema de tradução automática neural Seq2Seq (LUONG *et al.* 2015; SUTSKEVER *et al.* 2014). Esse modelo define uma rede neural de dois LSTM, sendo um deles responsável por codificar o texto de entrada conforme o vocabulário incorporado e gerar estados ocultos e de saída, e o outro responsável por decodificar a informação com base nos estados recebidos do codificador e seus próprios estados. Então, durante o processo de treinamento (Figura 8), na primeira etapa é recuperado o conjunto de pares de pergunta e resposta, em seguida, é realizado o NLP. Logo em seguida, é dividido em conjuntos de treino e teste e, finalmente, as etapas de treino em lotes da rede neural de tradução automática. Assim, como no treinamento da rede neural de classificação de texto, aqui também foi definido para gerar o *checkpoint* dos melhores estados da rede.

**Figura 8.** Esquema geral do processo de treinamento do modelo generativo.




Fonte: Os autores.

Para simplificar o entendimento, suponha que se tenha *n* pares (pergunta e resposta, 1:1) definidos. Os *n* pares passaram pelo NLP, em que haverá o processo de tokenização. Também há a inserção de novos *tokens* (somente no modelo generativo) e, posteriormente, a definição do vocabulário. Os *tokens* são <PAD> (preenchimento das sentenças para normalização do comprimento), <UNK> (*tokens* não conhecidos do vocabulário), <START> (indicação do início da sentença) e <END> (indicação do fim da sentença). Em seguida, os *n* pares são convertidos para duas matrizes distintas (perguntas e respostas) em que cada célula se tratam de um token convertido em um índice referente a sua presença no vocabulário, gerando algo semelhante ao apresentado na Figura 9.

**Figura 9.** Tokenização, inserção de novos *tokens* e criação de matrizes.




Fonte: Os autores.

Essas matrizes são divididas nos conjuntos de treino e teste e encaminhadas às etapas de treino da rede neural até obter o melhor resultado (menor valor de perda).

### 3.6. Testes e Discussões

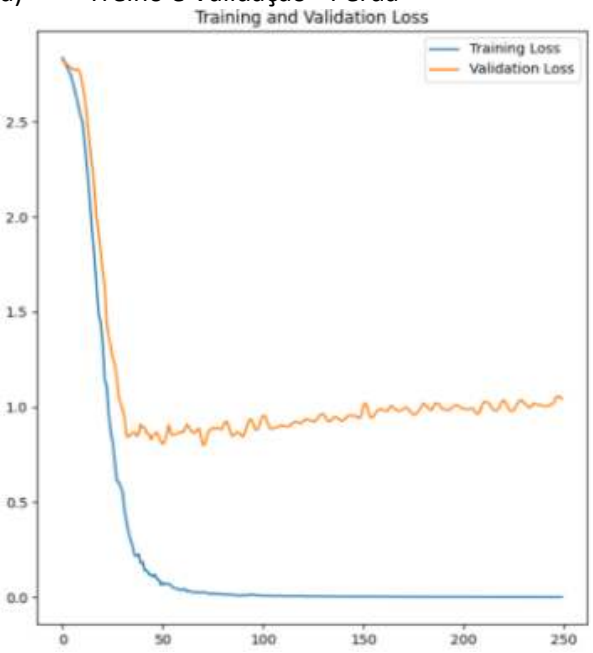
O modelo baseado em recuperação foi avaliado a fim de estimar sua acurácia. Na Figura 10 são apresentadas as estatísticas de perda e acurácia da etapa de validação durante o treinamento de uma base de dados que lida com diálogos de consultoria sobre o esporte Tênis. O gráfico (a) da Figura 10 indica o valor de perda ao longo do treinamento da rede neural de classificação de texto, o valor de perda nada mais é do que a capacidade da rede neural em se adaptar ao conjunto de dados de treino, para alcançar um modelo ideal. Como pode-se ver no gráfico (a), ao longo do treinamento (linha azul) o valor de perda segue em queda por conta da normalização dos pesos da rede neural, ou seja, conforme há distinção entre os resultados esperados e os resultados obtidos, o modelo segue definindo novos pesos na rede conforme a função de minimização de perda da rede, através de uma

política de gradiente. O processo de validação (linha laranja) avalia como os dados não incluídos nos lotes de treino se sobressaem ao modelo da rede neural com pesos recém atualizados, portanto seus valores de perda se mantêm um pouco acima dos obtidos pelo lote de treinamento (linha azul).

No gráfico (b) da Figura 10 é apresentada a taxa de acerto ao longo do processo de treinamento e durante a validação obteve uma acurácia média em torno de 76.40%. Como pode-se ver na área delimitada do gráfico (b), é notada uma estabilização no modelo em obter resultados esperados. Em um comparativo entre a acurácia dos dados não incluídos nos lotes de treinamento (linha laranja) com a acurácia dos lotes utilizados como parâmetros do treino (linha azul), conclui-se o quanto o modelo se torna altamente adaptado aos lotes de treinamento.

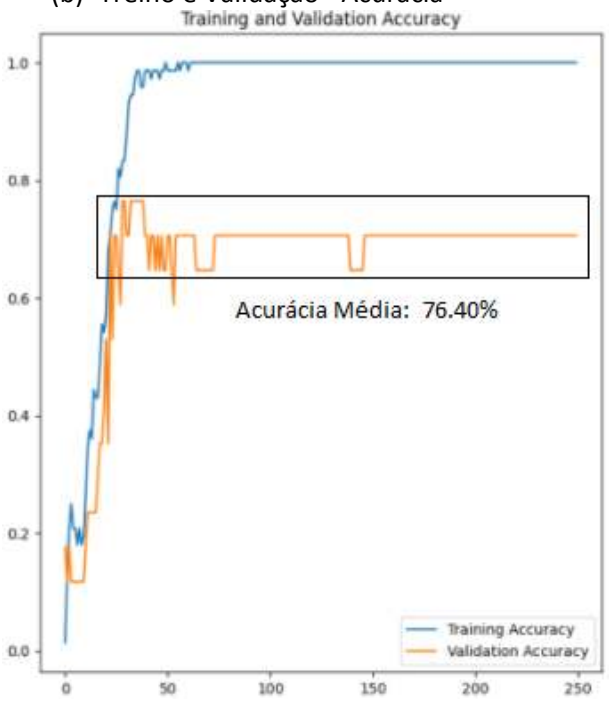
**Figura 10.** Perda e Acurácia das etapas de treino e validação.

(a) Treino e Validação - Perda




Fonte: Os autores.

(b) Treino e Validação - Acurácia




Fonte: Os autores.

Os testes foram auxiliados pela biblioteca Matplotlib (HUNTER, 2007), que permite utilizar o histórico de perda e acurácia gerado ao longo do treinamento da rede neural como conjunto de informações para representação gráfica. Tudo pôde ser feito no mesmo ambiente do servidor.

Apesar de se ter alcançado uma acurácia média de 76,40% é preciso compreender que não é um índice ideal pois, como discutido na Seção 3.5, se o modo de fluxo está ativo, então a não correspondência de fluxo entre o estado atual e novo estado a ser definido fará com que o chatbot sugira qual o próximo passo que o usuário deve adotar, portanto, a possibilidade de não haver correspondência da intenção levaria a um comportamento inconsistente em que o chatbot não procura contornar a situação e surpreender com uma boa resposta e recuperação do contexto da conversa em nosso modelo, mas sim passar a sugerir incansavelmente qual a mensagem para atingir o próximo estado do fluxo.

Muito provavelmente o que contribuiu para a taxa razoável de acurácia do modelo foi a definição de uma base de dados (quantidade de intenções e exemplos de entrada) curta para o treinamento suficiente do modelo, contudo, para a ferramenta, a quantidade de informações, em hipótese alguma, foi constatada como inviável, já que um chatbot personalizado deve seguir os conceitos de seu proprietário.

Na Tabela 1 são apresentados alguns testes de interação por meio da ferramenta. Nela se pode

notar uma perda de reconhecimento da intenção relacionada a intenção "*top_player*" em que mesmo com a mensagem "Melhor Jogador", semelhante aos exemplos de entrada ("Quais são os melhores jogadores no tênis", "Liste os melhores jogadores", entre outras) o modelo não foi capaz de gerar o resultado esperado.

**Tabela 1.** Testes de troca de mensagens através da ferramenta.

| Intenção | Exemplos de entrada | Entrada | Reconheceu a intenção ? |
|---|---|---|---|
| start_conversation | Oi<br>Olá<br>Ei | Oi | Sim |
| options | Como você pode me ajudar?<br>O que você pode fazer?<br>O que você sabe? | Pode me ajudar? | Sim |
| | | Preciso de ajuda | Sim |
| top_players | Quais são os melhores jogadores no tênis?<br>Liste os melhores jogadores<br>Melhores jogadores?<br>Você sabe sobre os melhores jogadores?<br>Quem é o melhor jogador de tênis? | Quem é o melhor jogador | Sim |
| | | Melhor jogador | Não |
| general_rules | Quais são as regras gerais do tênis?<br>Você sabe sobre as regras gerais?<br>Me diz as regras do tênis? | Regras do tênis | Sim |
| | | Quais as regras do tênis? | Sim |

Fonte: Os autores.

Isso indica que o modelo, por mais que permita a construção do chatbot, ainda carece de melhorias que viabilizem o treinamento e obtenção de respostas a partir de um modelo de rede neural bem elaborado.

Com relação ao modo livre, notou-se que a classificação entre as respostas produzidas pelo modelo baseado em recuperação e o modelo generativo, provocou uma certa confusão, quando estava sendo avaliada a similaridade entre as respostas recuperadas pelo modelo baseado em recuperação e a resposta automática gerada pelo modelo generativo. Tal confusão ocorreu por conta de se utilizar apenas a similaridade como fator de classificação entre as respostas do bot e a mensagem de entrada do usuário. Consequentemente, resultando na falta de contextualidade com a mensagem de entrada do usuário, não sendo considerado ideal, pois provoca inconsistência do chatbot quanto às intenções e respostas definidas pelo usuário.

Apesar de algumas incoerências encontradas em alguns casos de testes, a acurácia de 76,40% nos testes citados, indicam que a ferramenta está atingindo seus objetivos, bem como, pode-se ressaltar a facilidade na definição do chatbot customizado pelo próprio usuário.

## 4. CONSIDERAÇÕES FINAIS

Este artigo apresentou a ferramenta Easy Assistant, que viabiliza o desenvolvimento de chatbots personalizados com abrangência de tópicos específicos. Para tanto, sua estrutura é composta por um ambiente apresentado em navegadores web que permite definir intenções e exemplos de mensagens de entrada que reconhecem o objetivo da conversa com o chatbot, além de definir fluxo de diálogo e permitir a sugestão de respostas. De um outro lado, a presença de um servidor que manipula a base de dados diretamente e fica a cargo de processar os modelos do chatbot (baseado em recuperação e generativo).

Com base nos resultados obtidos e na utilização da ferramenta, pode-se concluir que ela tem a capacidade de suprir o desenvolvimento customizado de chatbots. O modelo baseado em recuperação com a rede neural classificador de texto permitiu o treinamento das intenções e consulta de respostas, por consequência é possível definir um objetivo ao chatbot.

O modo de fluxo, quando habilitado, permite que o chatbot se comunique e se guie com base em estados interligados. Contudo, procurar manter o estado, considerando uma situação em que o usuário esteja empenhado em não seguir o fluxo do diálogo, tornará o comportamento do chatbot inconsistente, ou seja, ele passará a sempre sugerir qual o próximo estado do fluxo sem se adaptar à conversa e contornar a situação, por consequência perdendo o contexto da conversa.

Conclui-se que futuramente é preciso aprimorar os modelos de treino e obtenção de resposta, inclusive um outro formato de combinação entre o modelo baseado em recuperação e o modelo generativo estudado, basicamente, as respostas predefinidas do primeiro modelo se tornam parâmetros de entrada para o segundo modelo (ZOPH; KNIGHT, 2016), contudo houve complicações de incompatibilidade durante a implementação do método de treino e avaliação. Além disso, é preciso aprimorar a usabilidade da ferramenta, agregando melhores recursos de visualização e manipulação no lado cliente, inclusive a implementação de perfis para o treinamento de distintos modelos de chatbot.

Como avanços futuros, é necessário promover melhorias no chatbot personalizado,

abstraindo as complexidades do processo de definição dos contextos de diálogos. Desta forma, irá permitir a contribuição para a área educativa, considerando que educadores poderão definir seus próprios modelos para instruir determinados tópicos relacionados a disciplinas de estudo.

## REFERÊNCIAS


ABADI, M. *et al.* **Tensorflow: A system for large-scale machine learning**. *In*: Symposium on operating systems design and implementation, 12., 2016. Georgia USA. **12. USENIX [...]**. USENIX Association, 2016. p. 265-283.

BANKS, Alex; PORCELLO, Eve. **Learning React: functional web development with React and Redux.** O'Reilly Media, Inc., 2017.

BORGES, Luiz Eduardo. **Python para desenvolvedores: aborda Python 3.3.** Novatec Editora, 2014.

CAHN, Jack. **CHATBOT: Architecture, design, & development**. University of Pennsylvania School of Engineering and Applied Science Department of Computer and Information Science, 2017.

DHINGRA, Bhuwan; LI, Lihong; LI, Xiujun; *et al.* **Towards End-to-End Reinforcement Learning of Dialogue Agents for Information Access.** arXiv preprint arXiv:1609.00777, 2017. https://doi.org/10.18653/v1/P17-1045

GRAVES, Alex; MOHAMED, Abdel-rahman; HINTON, Geoffrey. **Speech recognition with deep recurrent neural networks**. *In*: International conference on acoustics, speech and signal processing, 2013, Vancouver BC Canada. **2013 IEEE [...]**. IEEE, 2013. p. 6645-6649. https://doi.org/10.1109/ICASSP.2013.6638947

GREFF, K. *et al.* **LSTM: A Search Space Odyssey. IEEE Transactions on Neural Networks and Learning Systems**, v. 28, n. 10, p. 2222–2232, 2017. https://doi.org/10.1109/TNNLS.2016.2582924

GRINBERG, Miguel. **Flask web development: developing web applications with python**. O'Reilly Media, Inc., 2018.

GULLI, Antonio; PAL, Sujit. **Deep learning with Keras.** Packt Publishing Ltd, 2017.

HUNTER, John D. **Matplotlib: A 2D graphics environment. IEEE Annals of the History of Computing**, v. 9, n. 03, p. 90-95, 2007. https://doi.org/10.1109/MCSE.2007.55

KALCHBRENNER, N.; BLUNSOM, P. **Recurrent continuous translation models**. *In:* Conference on empirical methods in natural language processing, 2013, Seattle Washington USA. **Proceedings [...]**. Seattle: Association for Computational Linguistics, 2013. p. 1700-1709.

LIU, B.; LANE, Ian. **An End-to-End Trainable Neural Network Model with Belief Tracking for Task-Oriented Dialog**. Interspeech 2017, p. 2506–2510, 2017. https://doi.org/10.21437/Interspeech.2017-1326

LIU, B. *et al.* **Dialogue Learning with Human Teaching and Feedback in End-to-End Trainable Task-Oriented Dialogue Systems.** arXiv preprint arXiv:1804.06512, 2018. https://doi.org/10.18653/v1/N18-1187

LOKMAN, A. S.; AMEEDEEN, M. A. **Modern Chatbot Systems: A Technical Review.** *In*: Future Technologies Conference (FTC), 2018, Vancouver Canada. **Proceedings [...]**. Cham: Springer, 2018. p. 1012–1023. https://doi.org/10.1007/978-3-030-02683-7_75

LOPER, E.; BIRD, S. **Nltk: The natural language toolkit.** arXiv preprint cs/0205028, 2002. https://doi.org/10.3115/1118108.1118117

LUONG, Minh-Thang; PHAM, Hieu; MANNING, Christopher D. **Effective Approaches to Attention-based Neural Machine Translation.** arXiv preprint arXiv:1508.04025, 2015. https://doi.org/10.18653/v1/D15-1166

MIKOLOV, T. *et al.* **Recurrent neural network based language model**. *In*: Interspeech, 2010, Chiba Japan. **Interspeech [...]**. p. 1045-1048. https://doi.org/10.21437/Interspeech.2010-343

NADKARNI, P. M.; OHNO-MACHADO, L.; CHAPMAN, W. W. **Natural language processing: an introduction**. Journal of the American Medical Informatics Association, p. 8.

RUSSELL, S.J.; NORVIG, P. **Artificial intelligence: a modern approach.** 3. ed. Boston Columbus Indianapolis: Pearson, 2016.

SILVA, M. S. **Construindo sites com CSS e (X) HTML: sites controlados por folhas de estilo em cascata**. São Paulo: Novatec, 2007.

SILVA, M. S. **HTML5: a linguagem de marcação que revolucionou a web.** [São Paulo]: Novatec, 2019.

SINGH, A.; RAMASUBRAMANIAN, K.; SHIVAM, S. **Building an Enterprise Chatbot: Work with Protected Enterprise Data Using Open Source Frameworks.** Berkeley, CA: Apress, 2019. https://doi.org/10.1007/978-1-4842-5034-1

SONG, Y. *et al.* **An Ensemble of Retrieval-Based and Generation-Based Human-Computer Conversation Systems**. In: International joint conference on artificial intelligence, 27., 2018, Stockholm Sweden. **Proceedings [...]**. AAAI Press, 2018. p. 4382-4388. https://doi.org/10.24963/ijcai.2018/609

SRINIVASA-DESIKAN, Bhargav. **Natural Language Processing and Computational Linguistics: A practical guide to text analysis with Python, Gensim, spaCy, and Keras**. Packt Publishing, 2018.

SUTSKEVER, I.; VINYALS, O.; LE, Q. V. **Sequence to sequence learning with neural networks**. *In*: International Conference on Neural Information Processing Systems, 27., 2014, Montreal Canada. **Proceedings [...]**. Cambridge: MIT Press, 2014. p. 3104-3112.

YANG, Liu *et al.* **A hybrid retrieval-generation neural conversation model**. *In*: International conference on information and knowledge management, 28., 2019, Beijing China. **Proceedings [...]**. New York: ACM, 2019. p. 1341-1350. https://doi.org/10.1145/3357384.3357881

ZHANG, J.; THALMANN, N. M.; ZHENG, J. **Combining Memory and Emotion With Dialog on Social Companion: A Review**. *In:* International conference on computer animation and social agents, 29., 2016, Geneva Switzerland. **Proceedings [...]**. New York: ACM, 2016. p. 1–9. https://doi.org/10.1145/2915926.2915952

ZHOU, C. *et al.* **A C-LSTM Neural Network for Text Classification.** arXiv preprint arXiv:1511.08630, 2015.

ZOPH, B.; KNIGHT, K. **Multi-Source Neural Translation**. *In:* Conference of the North American chapter of the association for computational linguistics: human language technologies, 2016, San Diego California. **Proceedings [...]**. Association for Computational Linguistics, 2016, p. 30–34. https://doi.org/10.18653/v1/N16-1004