CENTRO UNIVERSITÁRIO DE ANÁPOLIS – UniEVANGÉLICA
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

OTÁVIO VIRGILIO BATISTA RAGAZZO
REINALDO DAVID RIBEIRO

HEVA, ASSISTENTE INTELIGENTE BASEADA EM PROCESSAMENTO DE LINGUAGEM NATURAL PARA APRIMORAR A EXPERIÊNCIA DE USUÁRIO

ANÁPOLIS
2020

OTÁVIO VIRGILIO BATISTA RAGAZZO
REINALDO DAVID RIBEIRO

HEVA, ASSISTENTE INTELIGENTE BASEADA EM PROCESSAMENTO DE LINGUAGEM NATURAL PARA APRIMORAR A EXPERIÊNCIA DE USUÁRIO.

Trabalho de Conclusão de Curso II apresentado como requisito parcial para a conclusão da disciplina de Trabalho de Conclusão de Curso II do curso de Bacharelado em Engenharia de Computação do Centro Universitário de Anápolis – UniEVANGÉLICA.

Orientador(a): Prof. Natasha Sophie Pereira.

Anápolis
2020

OTÁVIO VIRGILIO BATISTA RAGAZZO
REINALDO DAVID RIBEIRO

HEVA, ASSISTENTE INTELIGENTE BASEADA EM PROCESSAMENTO DE LINGUAGEM NATURAL PARA APRIMORAR A EXPERIÊNCIA DE USUÁRIO.

Trabalho de Conclusão de Curso II apresentado como requisito parcial para a obtenção de grau do curso de Bacharelado em Engenharia de Computação do Centro Universitário de Anápolis – UniEVANGÉLICA.

Aprovado(a) pela banca examinadora em [dia] de [mês] de 2020, composta por:

_____
Prof. Natasha Sophie Pereira
Orientador

_____
Prof. [nome do professor]

_____
Prof. [nome do professor]

Agradecimentos

Gostaríamos de agradecer primeiramente a Deus que é o centro de nossas vidas, e a todos os professores que nos acompanharam e incentivaram nesta grande jornada, de modo especial aos nossos orientadores M. Natasha Sophie e Willian Pereira, que exerceram com excelência seu papel, sendo atenciosos e pacientes. É importante também citar, nosso amigo Danillo Nogueira (Bacharel em Sistemas de Informação), que nos ajudou com dicas e sugestões nesse trabalho.

Reinaldo agradece também, a sua mãe e seu pai que sempre incentivaram e acreditaram em seu potencial, estando ao seu lado dando muito carinho e apoio. E a todas as suas três irmãs que ajudaram em seu desenvolvimento como pessoa, profissional e aluno. Também a sua namorada que em momentos difíceis, sempre esteve dando seu apoio e tranquilidade durante a elaboração deste trabalho.

Otávio agradece seu pai e sua mãe por estarem sempre ao seu lado, dando total apoio. Agradece também a toda família e amigos por acreditarem em seu potencial, e a todos que estiveram ao seu lado, incentivando e levantando o moral. Obrigado a todos por tudo.

"A vida é uma guerra, os amigos são os soldados,
e quando o capitão cai, eles estão aqui pra levantá-lo"
A fúria da tormenta - Pato Papão

Resumo

Um *chatbot* é um software que pode se comunicar com um humano usando linguagem natural. A necessidade de agentes de conversação tornou-se ampla, baseada no constante uso de máquinas pessoais e na necessidade de um usuário se comunicar, além do desejo de fabricantes fornecerem interfaces de comunicação mais humanas. Embora os *chatbots* executem muitas tarefas, a principal função que eles devem desempenhar é entender declarações dos seres humanos e respondê-las adequadamente. No passado, métodos simples com modelos e regras fixas eram usados para a construção de arquiteturas de *chatbot*, porem, com o aumento das capacidades de aprendizado, arquiteturas baseadas em redes neurais substituíram esses modelos. Neste trabalho, foi utilizado conceitos e técnicas de inteligência artificial para construir um agente conversacional inteligente, moldando uma arquitetura funcional para se construir Heva, uma assistente inteligente para um sistema médico.

Palavras-chaves: Inteligência Artificial, *Chatbots*, Experiência de Usuário, Assistente Inteligente.

Abstract

A chatbot is software that can communicate with a human using natural language. The need for conversation agents has become broad, based on the constant use of personal machines and the need for a user to communicate, in addition to the desire of manufacturers to provide more humane communication interfaces. Although chatbots perform many tasks, the main function they must perform is to understand human beings' statements and respond accordingly. In the past, simple methods with models and fixed rules were used to build chatbot architectures, however, with the increase in learning capacities, architectures based on neural networks have replaced these models. In this work, artificial intelligence concepts and techniques were used to build an intelligent conversational agent, shaping a functional architecture to build Heva, an intelligent assistant for a medical system.

Palavras-chaves: Artificial intelligence, Chatbots, User Experience, Smart Assinstant.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

ANN	Redes Neurais Artificiais, do inglês Artificial Neural Network
API	Interface de programação de aplicação, do inglês Application Programming Interface
IA	Inteligência Artificial
IHC	Interação Humano-Computador
IPA	Assistente Pessoal Inteligente, do inglês *Intelligent Personal Assistant*
LSTM	Long Short-Term Memory
ML	Aprendizado de Máquina, do inglês *Machine Learning*
NLP	Processamento de Linguagem Natural, do inglês Natural *Language Processing*
NLU	Entendimento de Linguagem Natural, do inglês Natural Language Understanding
RNA	Redes Neurais Artificiais
UX	Experiência do Usuário, do inglês *User Experience*

Sumário

1. Introdução

1.1. Justificativa e Delimitação do Estudo

Para Babich (2017), *UX* (*User Experience*) é "a experiência que um usuário tem ao interagir com um produto, então *UX Design* é o processo pelo qual um designer tentou determinar qual será essa experiência", com isso, tem-se que o grande desafio da *UX* é encurtar o caminho a ser percorrido pelo usuário para realizar ou executar uma ação no sistema. O fluxo de entrar em um sistema para realizar alguma tarefa, complexa ou não, e ser obrigado a seguir um passo a passo monótono pode levar o usuário à fadiga.

Com base no que foi apresentado por Bozkurt (2018), uma Automação Inteligente de Processos (AIP), também conhecida como *chatbot* pode ser definida como um serviço alimentado por regras, com o qual um usuário interage por meio de uma interface de bate-papo na tentativa de humanizar a interação humano-computador, tornando a experiência de usuário um pouco mais natural do ponto de vista humano. Este serviço pode estar voltado para qualquer conteúdo, variando de funcional a divertido.

Segundo Cahn (2017), quem primeiro conceituou *chatbot* foi Alan Turing, quando levantou o questionamento sobre as máquinas serem capazes de pensar. Para o autor, "desde Turing, a tecnologia de *chatbots* melhoraram com os avanços no processamento de linguagem natural e no aprendizado de máquina" (CAHN, 2017). Assim tendo indícios de que *chatbots* convencionais tendem a não agradar tanto aos usuários por serem monótonos e não conseguirem decifrar e entender sua verdadeira necessidade.

Cahn (2017) descreve os *chatbots* como "sistemas online de diálogo humano-computador com linguagem natural" e, por este motivo, pesquisadores, engenheiros e cientistas investigam meios de atribuir inteligência e capacidade de tomar decisões a partir de Linguagem de Processamento Natural (NLP, do inglês *Natural Language Processing*). Este recurso é capaz de prover respostas com performance positivamente impactante e, assim, evitar insatisfações por parte dos clientes e ser um diferencial no meio da concorrência no mercado.

Porém, a maioria das soluções de *chatbots* atualmente são genéricas, ou seja, funcionam de forma monótona e não conseguem entender o real significado por trás de cada palavra em uma frase. Isso se deve ao fato de as soluções serem implementadas a partir de um *script* linear. É possível evidenciar esta linearidade a partir de uma tentativa de sair do

roteiro do *bot* e, durante esta, contabilizar o número de interações de correção do usuário até o diálogo do *bot* ser reiniciado.

Segundo a OURCYCLE (2020), CyclePlantões é uma solução de *software* ágil e inteligente voltada para a gerência de plantões em hospitais, através da qual médicos plantonistas, gestores de unidades, diretores e equipe de recursos humanos estão conectados a um ecossistema fácil e organizado.

Contendo a usabilidade e facilidade como principais artifícios para uma interação saudável entre o usuário e software CyclePlantões, foi levantada a seguinte questão: Como utilizar os artifícios de NLU para desenvolver um assistente virtual inteligente que entenda a intenção da frase de um usuário e interaja com o sofware a partir deste contexto?

1.1.1. Objetivos

1.1.2. Geral

A partir da compreensão de que é necessário aliar a usabilidade e a facilidade para promover uma interação mais amigável entre usuário e software, delineia-se o objetivo deste trabalho, a saber: desenvolver uma assistente virtual inteligente para o sistema CyclePlantões mediante aplicação de recursos de Entendimento de Linguagem Natural (NLU, do inglês Natural Language Understanding). O intuito da aplicação destes recursos é aprimorar a interação do software, inserido em um contexto, em função da ideia expressa pelo usuário.

1.1.3. Específicos

- Criar um dataset com um padrão de design de conversação simples e inteligente.
- Treinar um algoritmo de processamento de linguagem natural que entenda uma frase como entrada e retorne uma resposta coerente como saída.
- Testar as possibilidades e os caminhos criados a partir de uma conversa com contexto linear ou aleatório.
- Possibilitar a comunicação do assistente com a API do sistema CyclePlantões.

2. Fundamentação Teórica

2.1 Inteligência Artificial

A Inteligência Artificial (IA) é um dos campos mais recentes em ciência e engenharia, e também um dos mais estudados e visados por estudiosos, porém este tema não é tão recente assim. O trabalho começou logo após a Segunda Guerra Mundial, juntamente com a biologia molecular e teve este nome cunhado em 1956 (INSTITUTO DE ENGENHARIA, 2018).

O campo da IA não tenta apenas compreender, mas também construir entidades inteligentes. Para o estudo da IA, e consequentemente a construção dessas entidades inteligentes, são definidas quatro diferentes abordagens, cada uma apresentando métodos específicos, sendo eles: i) pensando como um humano; ii) pensando racionalmente; iii) agindo racionalmente; e iv) agindo como seres humanos (RUSSELL; NORVIG, 2013). Russell e Norvig (2013, p.4) apresentam definições interessantes para cada uma delas:

- Pensando como um humano: para determinar como programas pensam como seres humanos, é necessário definir como um ser humano pensa. É preciso penetrar nos componentes reais da mente humana. Existem três formas de fazer isso, através de introspecção, através de experimentos psicológicos e através de imagens cerebrais. Depois de obter uma teoria para a mente suficientemente precisa, será necessário expressar a teoria para um programa de computador.
- Pensando racionalmente: é o uso do campo da lógica para resolução de problemas. A tradição lógica dentro da IA espera desenvolver programas que consigam resolver qualquer problema solucionável descrito por notação lógica para criar sistemas inteligentes. Existem dois problemas nessa abordagem, o primeiro é que não é fácil enunciar o conhecimento informal nos termos formais exigidos pela notação lógica, o segundo, são os recursos computacionais, já que problemas com algumas centenas de fatos podem esgotar esses recursos, a menos que ele tenha alguma orientação sobre as etapas de raciocínio que deve tentar primeiro.
- Agindo racionalmente: um agente simplesmente é aquele que age. Certamente todos os programas de computador realizam alguma coisa, mas espera-se que um agente computacional faça mais. Espera-se que ele opere sob controle autônomo, perceba seu ambiente, persista por um período de tempo prolongado, adapte-se a mudanças

e seja capaz de criar e perseguir 18 regras. Um agente racional é aquele que age para alcançar o melhor resultado ou, quando há incerteza, o melhor resultado esperado.

- Agindo como seres humanos: a base para essa definição é o Teste de Turing. Proposto por Alan Turing em 1950, o teste foi projetado para fornecer uma definição operacional satisfatória de inteligência. No Teste de Turing, o computador passará no teste se um interrogador humano, depois de propor algumas perguntas por escrito, não conseguir distinguir se as repostas que recebe estão vindo de um computador ou de uma pessoa. Para que um computador seja aprovado no Teste de Turing, é necessário que ele tenha capacidade de: i) representar conhecimento, armazenando o que sabe ou ouve; ii) possuir raciocínio automatizado e utilizar as informações armazenadas sempre com a finalidade de responder perguntas e tirar novas conclusões; e iii) conseguir aprender e se adaptar a novas circunstâncias, detectar e extrapolar padrões, ou seja, exercer o Aprendizado da Máquina.

A Inteligência Artificial está ligada diretamente a várias atividades do cotidiano do ser humano, e pode ser aplicada em diversos nichos. Russel (1962) em seu livro “Inteligência Artificial”, afirma que a IA abrange uma enorme variedade de subcampos, do geral (aprendizagem e percepção) até tarefas específicas, como jogos de xadrez, demonstração de teoremas matemáticos, criação de poesia, direção de um carro em estrada movimentada e diagnóstico de doenças, reforçando a ideia de que a IA é relevante para qualquer tarefa intelectual, sendo verdadeiramente um campo universal.

A inteligência artificial é um campo da computação que procura reproduzir, por meios computacionais, capacidades que os seres humanos possuem de forma nata. Dessa forma, define-se um sistema como inteligente quando ele apresenta a capacidade de:

- Raciocinar.
- Planejar.
- Resolver problemas.
- Realizar indução, dedução lógica e abdução.
- Armazenar conhecimento.
- Comunicar-se através de uma linguagem natural.
- Perceber-se e adaptar-se ao meio.
- Aprender

2.2 Aprendizado de máquina

Um dos mais notáveis assuntos quando se fala de Inteligência Artificial é o Aprendizado de Máquina (ML, do inglês *Machine Learning*), que segundo Simon, Annina e Deo (2015), "é um campo da ciência da computação que evoluiu do estudo do reconhecimento de padrões e da teoria do aprendizado computacional". Ainda segundo os autores, "*Machine learning* é o aprendizado e a construção de algoritmos que podem aprender e fazer previsões sobre conjuntos de dados".

Um algoritmo de aprendizado de máquina tem como sua base a construção de um modelo a partir de entradas de exemplo para fazer previsões ou escolhas orientadas a dados, em vez de seguir instruções estáticas.

O aprendizado de máquina pode ser realizado de forma supervisionada ou não supervisionada. Segundo Simon, Annina e Deo (2015), no aprendizado de máquina supervisionado é necessário um conjunto predefinido de 'exemplos de treinamento', que facilitam a capacidade da máquina de chegar a uma conclusão precisa quando dados novos são fornecidos, e no aprendizado de máquina não supervisionado, o algoritmo recebe uma grande quantidade de dados onde deverá encontrar padrões e relacionamentos por conta própria.

2.3 Redes Neurais

O termo "Rede Neural" (NN do inglês *Neural Network*), é bem sugestivo, isso se dá pela grande quantidade de tecnologias criadas com base em Inteligência Artificial. Gurney (2004) utiliza uma linha de raciocínio peculiar e admissível para introduzir o que seria uma rede neural, notavelmente, máquinas podem possuir algo parecido com um cérebro, que pode ser potencialmente carregado com atributos baseados em conotações de ficção científica, como nos mitos de Frankenstein, referenciando diretamente à vida artificial e raciocínio de um organismo não vivo.

2.3.1 Redes Neurais Biológicas

Existe uma intersecção quando se fala em neurônios biológicos e artificiais, isso porque um neurônio artificial é baseado na estrutura de um biológico, de modo que as conexões de uma rede neural artificial se equiparam à estrutura montada por um grupo de uma rede neural biológica. A figura 1 apresenta a estrutura de um neurônio biológico.

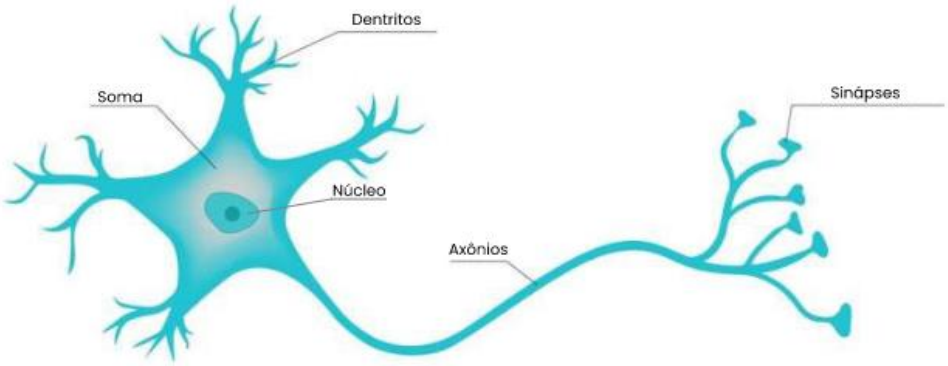


Figura 1 - Neurônio biológico. Fonte: Adaptado de Roos (2019)

Roos (2019) descreveu neurônios biológicos como células individuais, cada uma composta do corpo principal da célula, juntamente com muitos tendões que se estendem a partir desse corpo. O corpo, ou soma, abriga o maquinário que mantêm as funções celulares básicas e o processamento de energia, como o núcleo contendo DNA e organelas para construção de proteínas e processamento de açúcar e oxigênio. Existem dois tipos de tendões: dendritos, que recebem informações de outros neurônios e trazem para o corpo celular, e axônios, que enviam informações do corpo celular para outros neurônios.

Kiranyaz et al. (2019), observam que sistemas neurais, em geral, são construídos a partir de uma grande diversidade de tipos de neurônios, variando total ou parcialmente de propriedades estruturais, neurotransmissoras e nas propriedades elétricas em células e tecidos. Por exemplo, na retina de mamíferos, existem aproximadamente 55 tipos diferentes de neurônios para realizar uma detecção visual de baixo nível. De acordo com os sistemas neurológicos, várias operações distintas com pesos (parâmetros) adequados são criadas para atribuir uma diversidade de treinamento a tempo de execução e aprendizado a muitas funções neurais.

2.3.2 Redes Neurais Artificiais:

Os modelos simplificados de neurônios biológicos podem ser montados para formar o neurônio estereotipado em modelos de ML. A figura 2 apresenta a estrutura de um neurônio artificial.

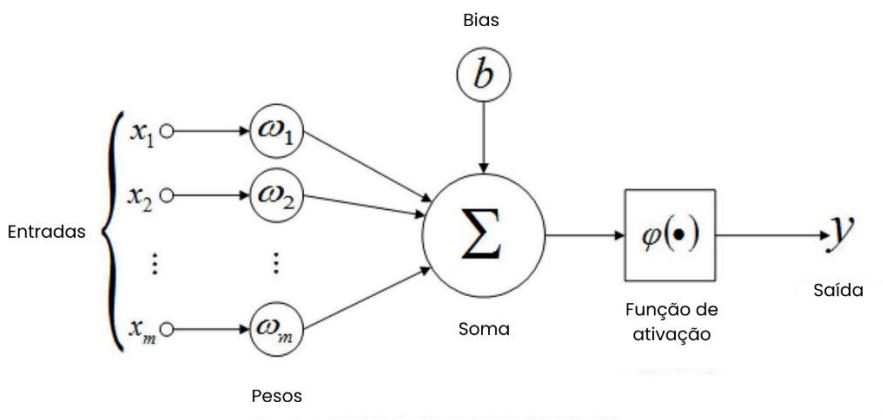


Figura 2 - Neurônio artificial. Fonte: Adaptado de Roos (2019)

Segundo Ross (2019), o neurônio artificial recebe entradas ou ativações vindas de outros neurônios. As ativações são multiplicadas por pesos sinápticos, e esses pesos são modelos de forças sinápticas em neurônios biológicos, na forma de que os pesos podem assumir valores negativos. As ativações ponderadas são somadas, modelando o processo de acumulação que acontece no corpo celular de um neurônio biológico. Um termo de bias é adicionado à soma, modelando a sensibilidade geral do neurônio. O valor somado é moldado por uma função de ativação, geralmente uma função *sigmoid*[1] é utilizada. Isso modela a taxa mínima intrínseca de neurônios biológicos (zero) ou a taxa máxima (devido a detalhes nos mecanismos fisiológicos pelos quais os picos são gerados).

A Redes Neurais Artificiais (ANNs) foram projetadas com o intuito de simular as ações de tomada de decisão de um neurônio biológico, sendo assim, os melhores modelos de RNA têm uma estrutura extremamente similar à uma estrutura biológica. Kiranyaz et al. (2019) , citam que a estrutura de McCulloch-Pitts é o modelo mais típico de ser encontrado, usado principalmente em redes *feedforward* (olhar para a frente), na qual um neurônio se encarrega de conectar à próxima camada de neurônios, seguindo uma única direção, sem um caminho de volta, sempre partindo da camada de entrada rumo à camada de saída.

2.3.3 Camadas Ocultas

Chakravorty e Chakrabarti (2009) explicam que as camadas ocultas residem entre as camadas de entrada e saída, por isso o motivo pelo qual elas são chamadas de ocultas. A

[1] Segundo Sharma (2017), uma das funções de ativação mais comuns usadas em redes neurais. Ela sempre retorna um valor entre 0 e 1, de modo que grandes valores positivos convergem para 1 e grandes valores negativos convergem para 0.

palavra “oculto” implica que elas não são visíveis para os sistemas externos e são “privadas” para a rede neural. Em uma rede neural pode haver zero ou mais camadas ocultas.

Chakravorty e Chakrabarti (2009) ressaltam que a performance de uma rede pode ser aumentada se o volume de computação for reduzido. Sabe-se que na rede neural a saída está relacionada à entrada através do peso da entrada nas camadas ocultas. Ou seja, se a quantidade de camadas for diminuída, o tempo computacional será reduzido e pode-se obter maior eficiência.

2.3.4 Perceptrons

Sagar Sharma (2017) descreve o perceptron como um algoritmo simples que utiliza a lógica de *feedforward*, porém o que a torna tão especial é o fato de ela ser a primeira implementação de um algoritmo tomador de decisões baseados em aprendizado de máquina e redes neurais. A figura 1 apresenta a estrutura de uma perceptron.

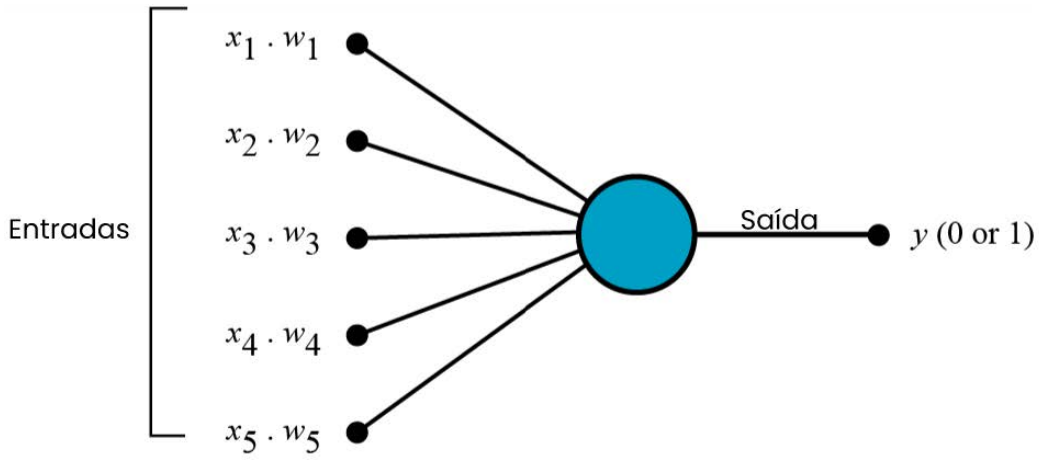


Figura 3 - Representação de um perceptron. Fonte: Adaptado de Sharma (2017)

O modelo não possui camadas ocultas, tornando-o assim um classificador linear, cada neurônio faz um cálculo a partir de várias entradas com valor real e retorna uma única saída binária (0 ou 1).

2.4 Redes Neurais Recorrentes

Segundo OLAH (2015), seres humanos não reiniciam seus pensamentos a cada segundo. Por exemplo, ao ler um artigo, você entende cada palavra com base em sua compreensão das palavras anteriores. Você não descarta o raciocínio e começa a pensar novamente "do zero", de modo que seus pensamentos têm persistência. Uma Rede Neural Recorrente (ANN) visa justamente dar essa capacidade de persistência ao fluxo de processamento de uma rede neural.

Redes neurais tradicionais são carentes de conservação e continuidade de dados. Por exemplo, imagine que você deseja classificar que tipo de evento está acontecendo em cada ponto de um filme. Não fica claro como uma rede neural tradicional poderia usar seu raciocínio sobre eventos anteriores no filme para informar os posteriores. As redes neurais recorrentes resolvem esse problema. São redes com *loops*, permitindo que as informações persistam. A figura 4 esquematiza o funcionamento de uma RNR.

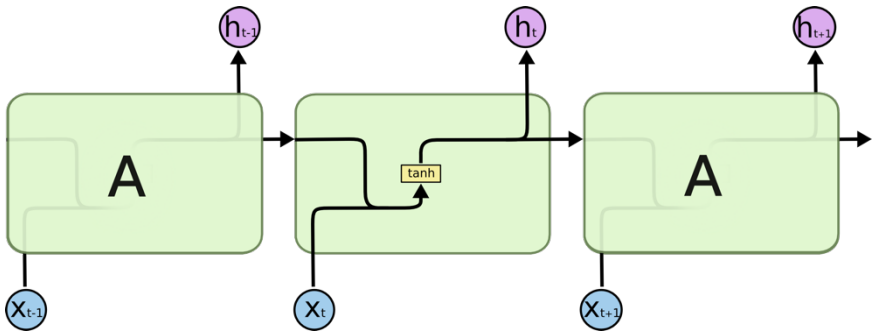


Figura 4 - Representação de uma RNR. Fonte: OLAH (2015)

Esses *loops* fazem as redes neurais recorrentes parecerem meio misteriosas. No entanto, descobre-se que elas não são tão diferentes de uma rede neural normal. Uma rede neural recorrente pode ser compreendida como várias cópias da mesma rede, cada uma passando uma mensagem a um sucessor. É evidenciado na figura 5 o desenrolar de um loop de uma RNR:

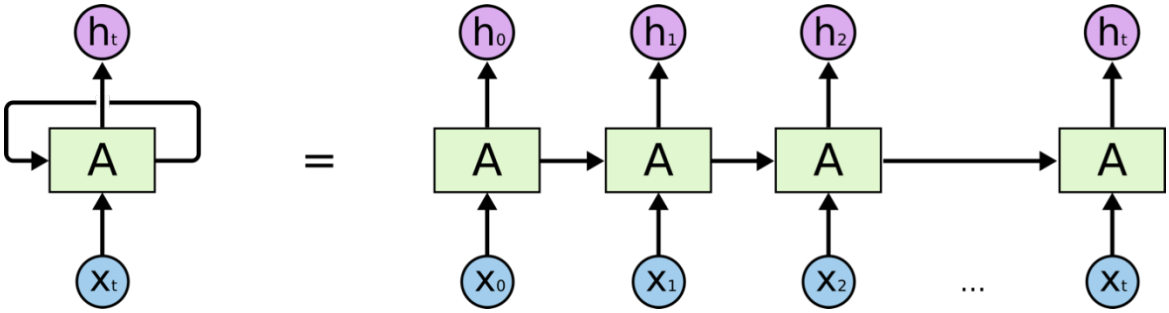


Figura 5 - Representação de uma RNR desenrolada. Fonte: OLAH (2015)

Essa natureza em cadeia revela que as redes neurais recorrentes estão intimamente relacionadas a sequências e listas, que compõem a arquitetura comum de uma rede neural.

Segundo OLAH (2015), RNRs têm tido um sucesso incrível na aplicação de RNNs a uma variedade de problemas: reconhecimento de fala, modelagem de linguagem, tradução, legendagem de imagens, entre outras. Isso graças à introdução ao uso de *Long Short Term Memory Networks* (LSTMs, ou Redes de Memória de Longo Prazo), um tipo especial de rede neural recorrente que é capaz de aprender dependências de longo prazo.

2.4.1 *Long Short-Term Memory Networks*

O modelo de LSTM foi introduzido por Hochreiter e Schmidhuber (1997) e foi projetado explicitamente para evitar o problema de dependência de longo prazo, tarefas como lembrar-se de informações por longos períodos é praticamente o comportamento padrão, e não algo que eles lutam para aprender.

LSTMs trabalham com a mesma estrutura em cadeia de RNRs, porém o módulo de repetição tem uma estrutura diferente. Ao invés de trabalhar com uma única função de processamento, ela trabalha com quatro. A figura 6 representa como é uma LSTM.

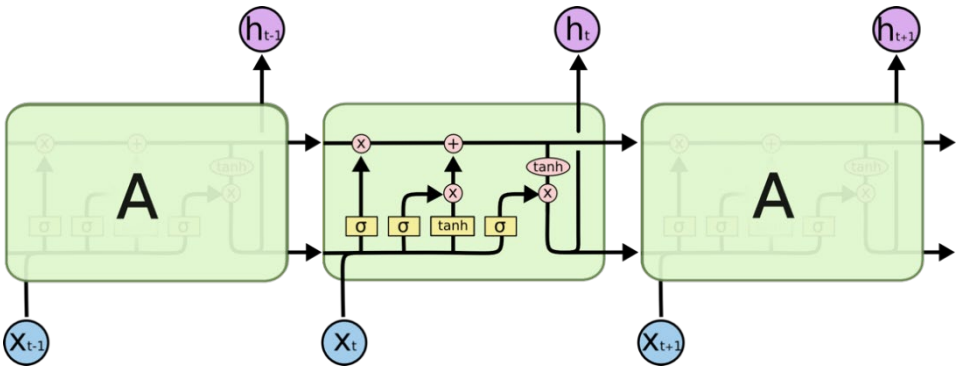


Figura 6 - Representação de uma LSTM. Fonte: OLAH (2015)

Para um entendimento mais técnico sobre LSTMs é essencial compreender o que é o estado da célula. Sherstinsky (2018) diz que o estado da célula funciona como um transportador, percorrendo toda a cadeia, com apenas algumas pequenas interações lineares. O LSTM tem a capacidade de remover ou adicionar informações ao estado da célula, cuidadosamente regulado por estruturas chamadas de portas.

Sherstinsky (2018) descreve a estrutura de portas como uma forma opcional de permitir a passagem de informações. Elas são compostas de uma camada de rede neural *sigmóide* e uma operação de multiplicação pontual. A camada *sigmóide*, por fim, emite como resultado um número entre zero e um, dizendo quanto de cada componente deve ser deixado passar. Um valor inteiro de 0 significa "não passar", enquanto um valor inteiro de 1 significa "deixe passar".

2.5 Processamento de Linguagem Natural

Processamento de Linguagem Natural (NLP, do inglês *Natural Language Processing*) representa e analisa a linguagem humana computacionalmente. Essa subárea surgiu para facilitar o trabalho do usuário, uma vez que nem todos os usuários possuem conhecimento em linguagem específica da máquina.

Khurana et al. (2017) expõem suas aplicações em vários campos, como tradução automática, detecção de *spam* em e-mails, extração de informações, sumarização, atendimento médico e resposta a perguntas. NLP é um tratado de Inteligência Artificial e Linguística, dedicado a fazer com que os computadores entendam as afirmações ou palavras escritas em línguas humanas.

Linguística, segundo Khurana et al. (2017), é a ciência da linguagem que inclui estudos sobre fonologia, formação de palavras, estrutura de frases, sintaxe, semântica e pragmática. Uma linguagem pode ser definida como um conjunto de regras ou conjunto de símbolos. O símbolo é combinado e usado para transmitir informações, porém, os símbolos são tiranizados pelas regras. O processamento da linguagem natural pode basicamente ser classificado em duas partes, que são a compreensão e interpretação dos símbolos e a geração de linguagem natural, que evolui sua função de apenas entender para gerar um símbolo.

2.6 Entendimento de Linguagem Natural (NLU)

Entendimento de Linguagem Natural (NLU, do inglês *Natural Language Understanding*) compreende uma ampla gama de tarefas diversas, como vinculação textual, resposta a perguntas, avaliação de similaridade semântica e classificação de documentos. A capacidade de aprender efetivamente com o texto bruto é crucial para aliviar a dependência do aprendizado supervisionado na NLP.

A maioria dos métodos de aprendizado supervisionado utilizados exigem quantidades excessivas de dados rotulados manualmente, o que restringe sua aplicabilidade em muitos domínios que sofrem com a escassez de dados. Radford (2018) complementa dizendo que nessas situações, os modelos que podem aproveitar informações linguísticas de dados não rotulados fornecem uma alternativa valiosa para a coleta de mais dados, o que pode ser uma alternativa demorada e cara. Além disso, mesmo nos casos em que há uma supervisão considerável, aprender boas representações de maneira não supervisionada pode fornecer um aumento significativo no desempenho do algoritmo. A evidência mais convincente para isso até agora tem sido o uso extensivo de combinações de palavras pré-treinadas para melhorar o desempenho em várias tarefas da NLP.

Os grandes ganhos podem ser alcançados pelo pré-treinamento generativo de um modelo de linguagem em uma base de dados diversificada de texto não rotulado, seguido de um ajuste fino discriminativo em cada tarefa específica. Em contraste com as abordagens anteriores, são usadas transformações de entrada com reconhecimento de tarefa durante o ajuste fino para obter uma transferência efetiva e exigir mudanças mínimas na arquitetura do modelo.

2.7 Chatbots

Assim como as pessoas usam a linguagem para comunicação humana, elas querem usar seu idioma para se comunicar com computadores. Zadrozny et al. (2000) concordaram que a melhor maneira de facilitar a Interação Humano-Computador (IHC) é permitindo que os usuários “expressem seus interesses, desejos ou consultas direta e naturalmente, falando, digitando e apontando”.

Chatbots são programas de computador que interagem com o usuário. Segundo Shawar e Atwell (2007), os sistemas de *chatbot* não são construídos apenas para imitar a

conversa humana e entreter os usuários, existem infinitas outras aplicações em que os *chatbots* podem ser úteis, como educação, recuperação de informações, negócios e comércio eletrônico.

2.7.1 *Chatbots* Convencionais

Um *chatbot* comum consiste em objetos de dados chamados de tópicos e categorias. O tópico é um elemento opcional de nível superior, possui um atributo de nome e um conjunto de categorias relacionadas a esse tópico. As categorias são a unidade básica de conhecimento. Cada categoria é uma regra que define a forma como uma entrada será convertida em uma saída, consistindo em um padrão que corresponde à entrada do usuário e um modelo que é usado para gerar a resposta do *chatbot* convencional.

O padrão é simples, consistindo apenas de palavras, espaços e os símbolos "_" (sublinhado) e "*" (asterisco). As palavras podem consistir em letras e números, mas nenhum outro caractere. As palavras são separadas por um único espaço. A linguagem do padrão é invariável a maiúsculas e minúsculas. A ideia da técnica de correspondência de padrões baseia-se em encontrar a melhor e mais longa correspondência de padrões.

Antes do algoritmo corresponder às frases, é aplicado um processo de normalização para cada entrada, para remover toda a pontuação, dividindo a entrada em duas ou mais frases, se apropriado, e convertendo todas as letras para caixa alta (maiúsculas). Por exemplo, se a entrada for: "Eu não sei. Você tem um arquivo robots.txt?". Depois da normalização, será: "EU NAO SEI VOCE TEM UM ARQUIVO ROBOTS DOT TXT".

Após a normalização, o intérprete tenta corresponder palavra por palavra para obter a correspondência de padrão mais longa, pois espera-se que essa seja a melhor. Esse comportamento pode ser descrito em termos do conjunto de arquivos e diretórios *Graphmaster*[2], que possui um conjunto de nós chamado *nodemappers*[3] e ramificações[4], representando as primeiras palavras de todos os padrões e símbolos curinga (Wallace, 2003).

No entanto, Shawar e Atwell (2007) afirmam que todas essas categorias são codificadas manualmente, consumindo tempo e restringindo a adaptação a novos domínios

[2] Algoritmos estruturados em forma de grafo utilizado por *chatbots* convencionais para buscar uma resposta satisfatória a partir de uma avaliação das sentenças de entrada.
[3] Coleção de nós que compõe um *Graphmaster*, são responsáveis por mapear os ramos de cada nó.
[4] Palavras sozinhas ou os caracteres curingas.

do discurso e novas linguagens, tornando desvantajosa a utilização de um modelo conversacional comum.

Outras limitações citadas por Ayanouz, Anoua e Benhmed (2020) são:

- Diálogos baseados em conjuntos fixos de regras.
- Complexidade em reconhecer erros gramaticais.
- A maioria dos *chatbots* responde apenas às perguntas de um nicho restrito ou às perguntas definidas no banco de dados.
- Complexidade em entender o significado ou o contexto de uma frase não clara ou sem um objetivo explícito.
- Dificuldade em mudar repentinamente de assunto.
- Respostas sem nenhum contexto que reduzem o nível de precisão.

2.7.2 *Chatbots* Inteligentes

Um dos desafios da Inteligência Artificial é investigar como projetar e construir máquinas que são capazes não apenas de entender e raciocinar, mas também de perceber e expressar emoções.

Ayanouz, Abdelhakim e Benhmed (2020) apresentam um fluxo de pesquisa em NLP e aprendizado de máquina focado em criar sistemas generativos que modelam características humanas como um componente-chave para conversas e interações homem-máquina. Em vez de serem assistentes virtuais orientadas a tarefas, esses sistemas têm personalidade ou identidade e exibem opiniões e emoções em configurações de domínio aberto.

O processamento de linguagem natural é a base dos *chatbots* baseados em IA. Usando algoritmos sofisticados de NLP, os *chatbots* inteligentes podem processar o texto de entrada de forma que, entendam, concluam e determinem o que foi dito ou escrito e, em seguida, declararem uma lista de todas as ações adequadas. Ayanouz, Abdelhakim e Benhmed (2020) citam quatro etapas essenciais em um algoritmo NLP para que uma mensagem seja facilmente compreendida por um *chatbot*. São elas:

- Análise léxica: inclui análise e identificação da estrutura das palavras. Divide o texto nos capítulos, depois em frases e palavras.

- Análise sintática: analisa a gramática e o arranjo de palavras para que a relação entre palavras diferentes se torne mais explícita. Frases como “o hospital vai ao médico” são rejeitadas pelo analisador sintático.
- Análise semântica: verifica se o texto é completamente significativo ou não, e ele desenha seu significado correto ao mapear as construções sintáticas. A análise semântica rejeitará frases como "fogo frio".
- Análise pragmática: analisa a interpretação conclusiva da mensagem real do texto. Tal como o significado real de uma frase ou sentença se baseia no contexto geral.

Ayanouz, Abdelhakim e Benhmed (2020) reforçam que o mecanismo de processamento de linguagem natural consiste nos algoritmos mais recentes de aprendizado de máquina usados para identificar a intenção do usuário e, em seguida, combiná-los com uma lista de intenções que são suportadas pelos *bots*, sendo elas:

- Classificador de intenções: recebe informações, interpreta seu significado e faz o relacionamento com a intenção suportada pelo *chatbot*.
- Extrator de entidade: extrai as informações críticas da consulta de um usuário.

Apesar dos avanços computacionais e dos resultados promissores alcançados com modelos generativos de texto, sistemas ponta a ponta são frequentemente treinados em conjuntos de dados recuperados automaticamente em larga escala, mas de baixa qualidade. Esses conjuntos de dados são valiosos para a experimentação e otimização do algoritmo, mas menos relevantes para a construção de agentes de conversação que refletem características específicas do tipo humano, que por sua vez, também são difíceis de avaliar quantitativamente.

2.8 Assistentes

Pegando como base o Dicio, dicionário online, define-se “Assistente” como: “Que ou aquele que assiste”, e tendo como sinônimo: “auxiliar”, “ajudante”, “adjutor” (DICIO, 2020). Ou seja, um assistente, é aquele que presta auxílio em uma determinada área, como por exemplo um assistente de vendas, suporte técnico, atendimento ao cliente, entre outros.

2.8.1 Assistentes Humanos

Tendo em vista o ramo empresarial, os assistentes humanos são bastante utilizados pelas empresas, tanto no auxílio para uma venda, quanto em um suporte ou atendimento ao

cliente. Assistentes de Telemarketing e Suporte Técnico em *Softwares* são dois exemplos que podem ser utilizados para compreender melhor o que um assistente humano realiza.

Existem dois tipos de assistentes de telemarketing, os ativos, que executam todos os processos operacionais visando fazer chamadas diretas para vender seus produtos ou serviços; e os passivos, que não efetuam ligações, mas, são responsáveis por atender os clientes e fornecer informações e esclarecimentos sobre produtos e serviços, repassando as reclamações e efetuando uma compra se for necessário (INFOJOBS, 2020).

Assim como os assistentes de telemarketing passivos, o suporte técnico em *softwares* tem o mesmo papel em atender clientes, porém o que os diferencia é o conhecimento técnico, pois além de poder esclarecer dúvidas, realizam o suporte pelo atendimento e via acesso remoto (*HelpDesk*), a fim de sanar o problema e dúvidas que os respectivos clientes podem ter.

2.8.2 Assistentes Inteligentes

Com o surgimento de novas tecnologias, como *Big Data*, Inteligência Artificial, computação em nuvem, *Machine Learning*, entre outros, avanços significativos impactaram o mundo corporativo e a rotina das pessoas que o englobam. Atualmente uma das tecnologias que tem sido envolvida por essas transformações, são os chamados assistentes virtuais inteligentes. Um assistente inteligente consiste em uma categoria de entidades inteligentes sem corpo físico, que tem por objetivo auxiliar uma pessoa ou grupo de pessoas a resolverem questões que as estão afligindo (CRUZ; ALENCAR; SCHMITZ, 2013).

A possibilidade de poder controlar dispositivos, fazer buscas e cumprir funções que geralmente são realizadas diariamente, tudo utilizando apenas comandos de voz e texto parece utópica, porém, já faz parte da realidade atual. Grandes empresas de tecnologia, como Apple (Siri), Microsoft (Cortana) e Google (Google *Assistant*) utilizam deste recurso.

Assim como o assistente humano, os assistentes virtuais inteligentes interagem com os clientes, tendo uma experiência de conversação diferenciada a fim de sanar uma dúvida, realizar uma ação e até resolver um problema de forma ágil, de maneira que não necessite de um corpo humano por trás realizando esse atendimento.

Na Figura 7 é possível perceber um exemplo simples de um atendimento ao cliente por meio de um assistente virtual resolvendo um problema específico sem a necessidade de

um atendente humano. Por exemplo: Suponha que um cliente compre um produto pela internet, e queira realizar a troca desse mesmo produto. Nesse caso o diálogo pode ser feito por um atendente virtual por meio de um chat pela internet:

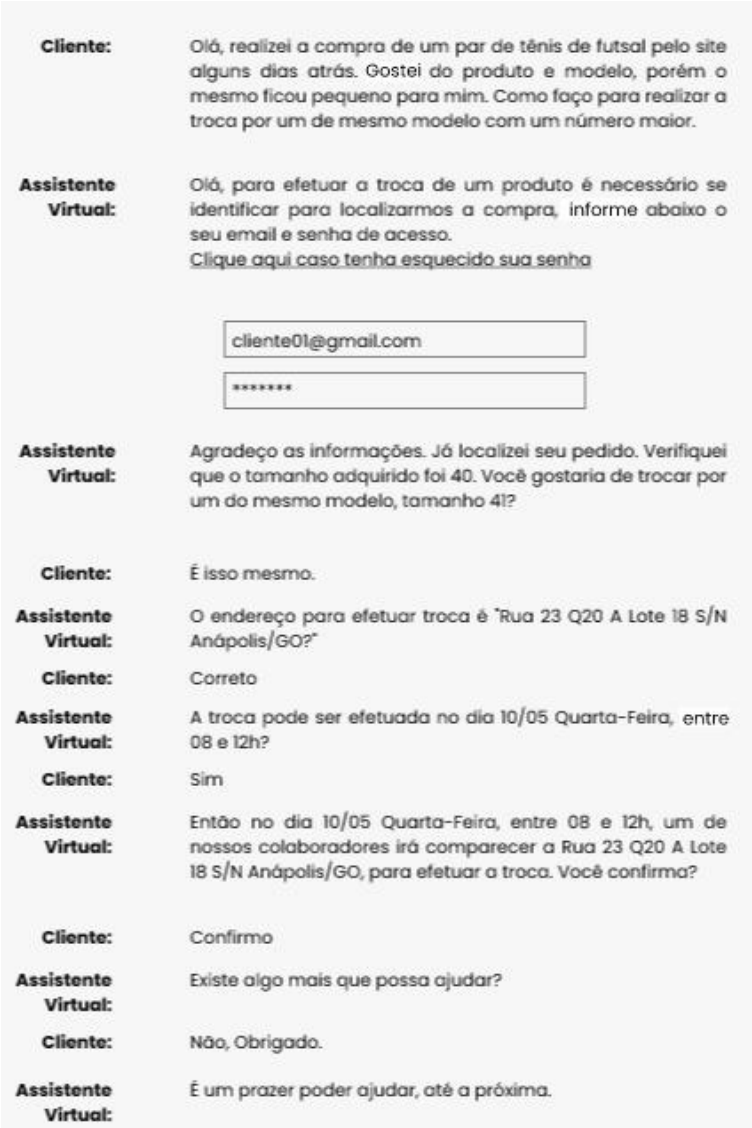


Figura 7 - Simulação de uma conversa com uma assistente virtual. Fonte: Dos autores (2020)

Observe que, durante a interação com o cliente, o assistente virtual:

- Forneceu informações, indicou ao cliente que ele precisava se identificar para fazer a alteração do pedido, e conseguiu encontrar o pedido e suas informações.
- Elaborou perguntas, confirmou o dia, horário e local onde iria ser realizada a troca.
- Solicitou informações ao cliente, pediu que ele fornecesse o seu e-mail e senha.

  Apresentou sugestões, sugeriu o tamanho do tênis para a troca, e o dia e horário factível para que a troca do produto fosse efetuada.

2.9 Vantagens dos Assistentes Inteligentes

Os assistentes virtuais tornam muito mais práticas as atividades do cotidiano, tanto nas empresas quanto na rotina doméstica das pessoas, auxiliando em suas tarefas e trazendo muitas vantagens significativas para a sociedade. No contexto empresarial podemos citar alguns dos inúmeros benefícios advindos do uso dessa tecnologia.

Atendimento personalizado, utilizando uma assistente virtual inteligente, as empresas conseguem proporcionar ao cliente um melhor atendimento. Isso porque essa tecnologia é capaz de se aperfeiçoar conforme vai interagindo com o público. Assim, consegue oferecer um atendimento adequado às preferências e características de seus clientes. Podem ser acessados por diversos dispositivos, e conseguem um atendimento personalizado no horário em que quiserem, diferente dos atendentes humanos, que geralmente têm horários fixos para o atendimento.

Redução de custos, é uma das maiores vantagens dos sistemas inteligentes para o ramo empresarial, pois essas ferramentas são capazes de resolver diversas questões e demandas dos consumidores, de modo que as empresas conseguem diminuir custos operacionais, dentre os quais os associados à manutenção da mão de obra humana.

Agilidade e praticidade, na sociedade em que vivemos, as pessoas procuram cada vez mais por métodos rápidos e práticos para realizarem qualquer tipo de tarefa. E por meio dessa tecnologia é possível trazer essa agilidade e praticidade que todos procuram, obtendo uma melhor experiência de usuário e, consequentemente, maior satisfação e fidelidade ao serviço/produto.

3. Planejamento da assistente inteligente HEVA

3.1 Ecossistema da assistente

Existem várias maneiras de construir uma IA de conversação inteligente. Pode-se aproveitar as plataformas de construção de *chatbots*, como *DialogFlow* criado pela Google, *Watson* criado pela IBM, wit.ai, recast.ai, Rasa.ai, dentre outras.

Como uma assistente inteligente requer uma memória de longo prazo para se lembrar das mensagens precedentes, uma boa estratégia seria a implementação de uma LSTM (rede de memória de longo prazo) para ajudar a manter o contexto das conversas.

Assim, foi definido que HEVA seria criada com base na plataforma Rasa, pois seus fluxos de implementação de LSTM dá a assistente inteligente a capacidade de permitir a construção de um fluxo conversacional independente. Este fluxo possui capacidade de processar e prever uma ação com base na intenção do usuário, que é extraída através da sequência de mensagens enviadas, além também da capacidade de prever a próxima ação que será tomada pelo usuário, permitindo a execução de uma ação antecipada pela assistente.

3.1.1 Design conversacional

Branigan et al (2010) diz que há fortes evidências de que quando duas pessoas conversam entre si, elas tendem a convergir, ou alinhar, em formas comuns de falar. Para criar uma assistente humanizada, esse padrão deve ser seguido. O estado da arte da assistente é alcançado quando os princípios de design conversacional levantados por Urban e Mailey (2019) são cumpridos, sendo eles: entender os usuários, definir persona, analisar caminhos de conversação, estratégias de escrita de diálogo e o processo detalhado de criação de diálogo natural.

Neste trabalho, o Design conversacional ajudará os desenvolvedores a criar diálogos simples e fluidos para que a interação com a assistente seja única e cativante a cada nova conversa.

3.1.2 Fluxo da IA Conversacional

Após uma IA conversacional receber uma solicitação do usuário por texto ela a passa para os módulos seguintes, conforme fluxo definido na figura 8.

- **Módulos conectores:** Esta é a plataforma de conversação, o *front-end*, e pode ser um site ou aplicativo, onde o usuário interage com a IA.
- **Módulos de entrada:** A solicitação do usuário é alimentada para este módulo, que faz Entendimento de Linguagem Natural (NLU).
- **Gerência de Diálogo:** Este módulo pega o texto NLU e mantém o contexto da conversa. Em seguida, ele usa o modelo de aprendizado de máquina, como um RNN ou LSTM, para sugerir uma ação apropriada a ser tomada. Este módulo também pode ser conectado a um *back-end* ou banco de dados para realizar processamento adicional com base na ação sugerida.
- **Módulos de Saída:** Este módulo executa a geração de linguagem natural para retornar uma resposta apropriada ao usuário.

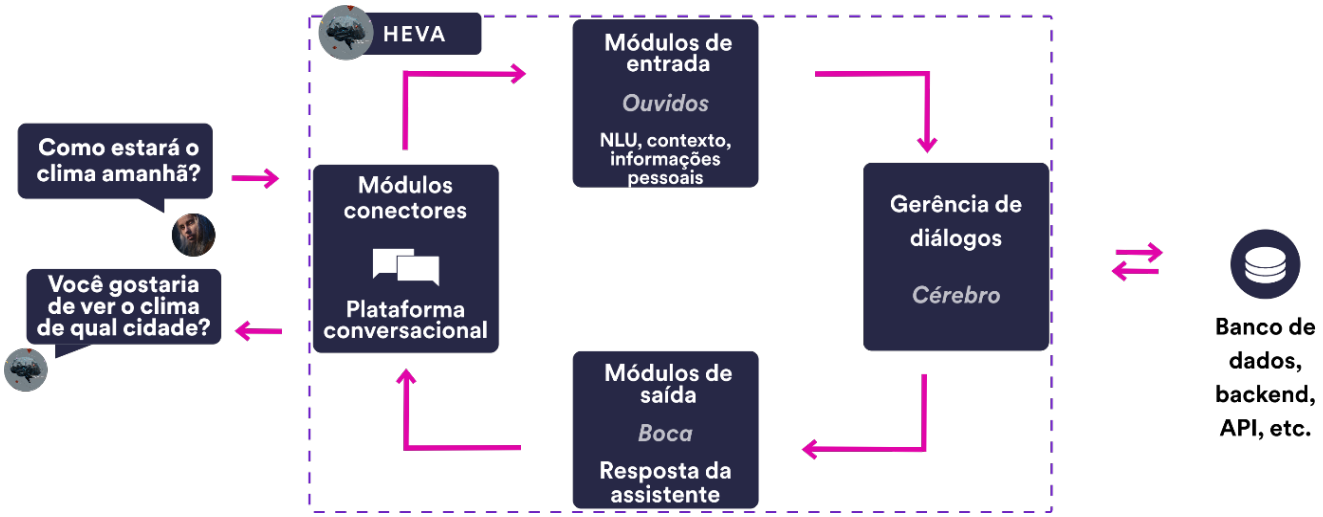


Figura 8 - Fluxo de processamentos por módulo. Fonte: Dos autores (2020).

3.1.3 NLU, NLP e o processamento dos dados

Antes de prosseguir sobre como a assistente reconhece e processa todos os textos do usuário, é fundamental entender os conceitos de intenção e entidade. De modo que intenção representa a finalidade da entrada de um usuário, o que o usuário deseja fazer. O texto de entrada do usuário é primeiro vetorizado e, em seguida, a intenção do texto é extraída. Já uma entidade representa um termo ou objeto que é relevante para suas intenções e que fornece um contexto específico para uma intenção. A Figura 9 ilustra o fluxo de processamento de dados e a manifestação da Intenção e da Entidade.

O NLU é responsável por processar o texto de entrada do usuário e entender o que o usuário está tentando dizer. Basicamente, ele pega o texto do usuário como entrada e extrai a intenção e as entidades dele. Ao extrair os dados da entrada do usuário, a intenção e as entidades são passadas para o NLP, que por sua vez decide o que acontece a seguir nesta conversa. Ele usa gerenciamento de diálogo baseado em aprendizado de máquina para prever a próxima melhor ação com base no histórico de conversas e seus dados de treinamento.

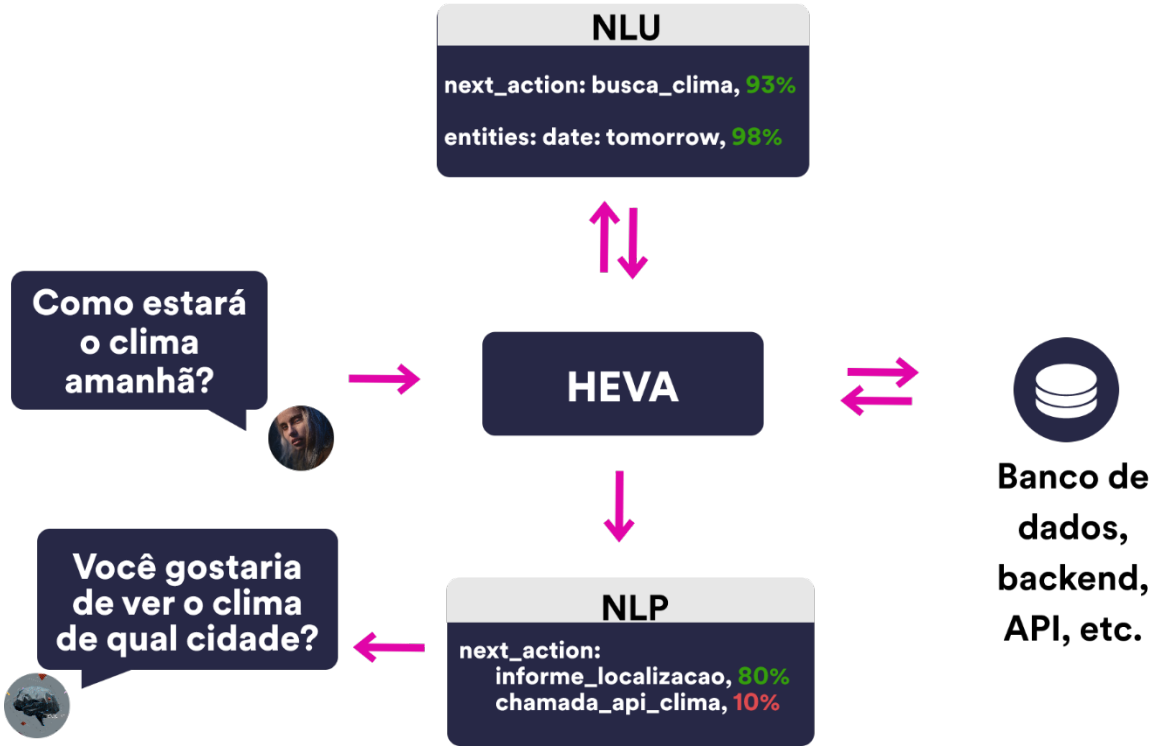


Figura 9 - Fluxo de processamento dos dados. Fonte: Dos autores (2020)

Como por exemplo, uma intenção de solicitação de clima, uma ação de "*location_question*" ou "*weather_api_call*" é sugerida. Finalmente, com base no tipo de ação, a resposta relevante é gerada e enviada para o usuário.

3.1.4 Desenvolvendo diálogos baseados em orientação a conversas

Ao projetar histórias, há dois grupos de interações conversacionais que precisam ser contabilizadas: caminhos felizes e infelizes. Caminhos felizes descrevem quando o usuário está acompanhando o fluxo de conversação como o esperado, sempre fornecendo as informações necessárias quando solicitado. No entanto, os usuários muitas vezes se desviam de caminhos felizes, implicando a ocorrência do que se chama caminho infeliz.

É importante para uma assistente lidar com caminhos infelizes graciosamente, mas também é impossível prever qual caminho um determinado usuário pode tomar. O planejamento para todos os estados possíveis em uma máquina de estados[5] requer muito trabalho extra e aumenta significativamente o tempo de treinamento.

Em alguns casos, a assistente pode lidar com um caminho infeliz desviando bruscamente a conversa para seu foco principal ao detectar a fuga de contexto, o que não é muito agradável para o usuário. A melhor solução para esse problema é adotar uma abordagem de desenvolvimento orientada a conversas com possibilidade de projetar caminhos infelizes.

O Desenvolvimento Orientado à Conversas promove continuamente a coleta de dados reais de conversação que informa exatamente como os usuários divergem dos caminhos felizes. A partir de uma análise de design conversacional sobre esses dados é possível criar histórias para realizar o que o usuário solicitou e, aos poucos, guiá-los de volta para um caminho feliz, lidando assim, com a fuga de contexto.

Babich (2017) em seu artigo *"What You Should Know About User Experience"* (em português, o que você deve saber sobre a experiência do usuário) mostra que experiências de usuário são projetadas em qualquer lugar, e que tudo, desde a forma de interagir com um produto de *software* até a localização de um botão liga-desliga, além da forma que ele é moldado, é um exemplo de elementos que constroem *UX*. A soma de suas interações com um produto se torna a experiência que você tem ao usar aquele produto.

[5] Modelagem de um comportamento composto por estados, transições e ações.

Babich (2017) diz que os sites e aplicativos mais bem-sucedidos de hoje fornecem muito mais do que informações relevantes em um formato fácil de consumir, eles criam uma experiência intuitiva que economiza o tempo do usuário. O tempo é essencial para o sucesso em ambientes móveis e de desktop, por isso, usuários desejam produtos que ajudem a atingir objetivos da forma mais rápida possível, ou seja, um padrão de *design* que permite que os usuários executem o menor número de etapas desde o momento em que tenham o primeiro contato com a ferramenta até o momento em que realizam uma ação.

Os *chatbots* ou assistentes inteligentes, por natureza, são completamente diferentes de qualquer outro tipo de interação humano-computador. Ao contrário dos sites e aplicativos, que são projetados para oferecer a mesma experiência para todos, os *chatbots* criam uma experiência única para cada usuário.

A individualidade da experiência para cada usuário se dá principalmente pelas diferentes formas que um usuário pode interagir com a assistente para realizar uma tarefa, a capacidade da assistente extrair real intenção da frase com base no contexto da conversa e das palavras usadas ajuda a certificar de que o usuário chegue ao seu objetivo da forma mais rápida possível.

O usuário também pode optar por executar ações de forma direta ao invés de criar um diálogo que coleta intenções de forma separada. As Figuras 11 e 12 apresentam dois exemplos de diálogo. Na Figura 11 o usuário opta pela abordagem com fornecimento gradual de informações; já na Figura 12 é utilizada a abordagem direta, com o repasse total de informações:

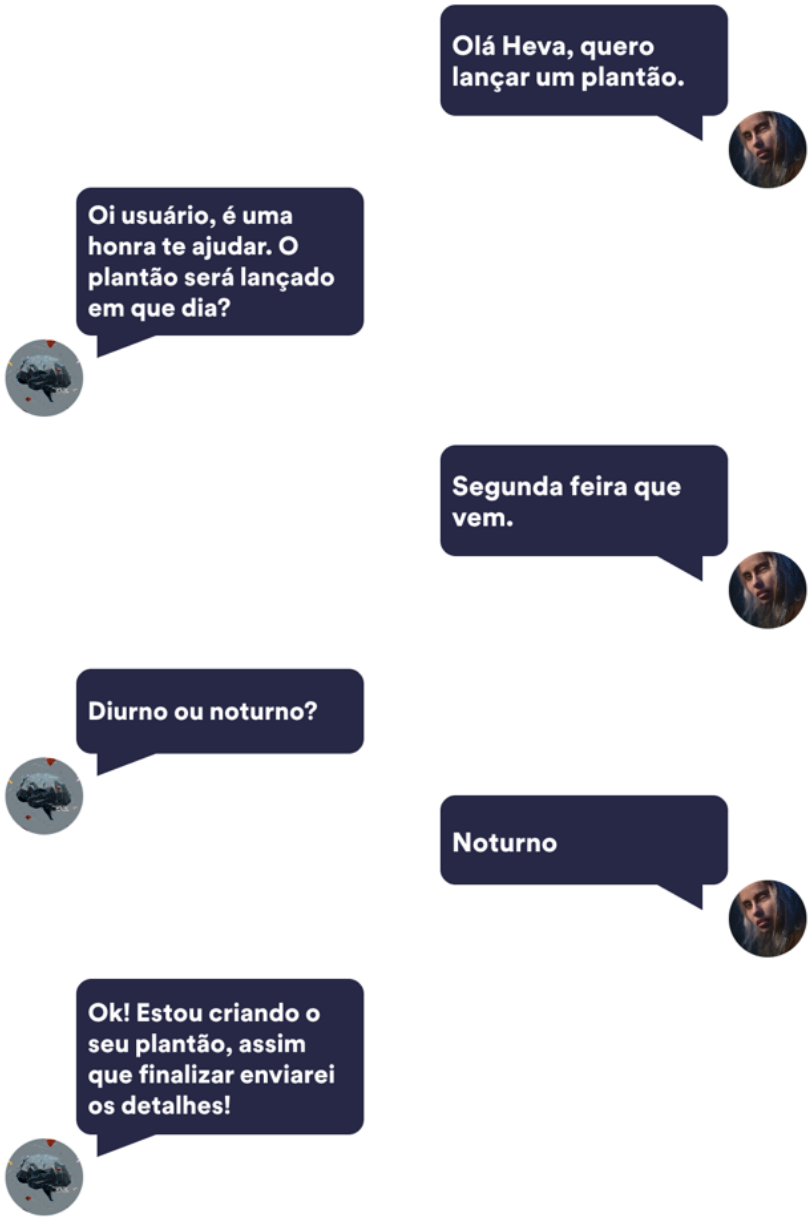


Figura 10 - Intenções podem ser extraídas em uma única interação.

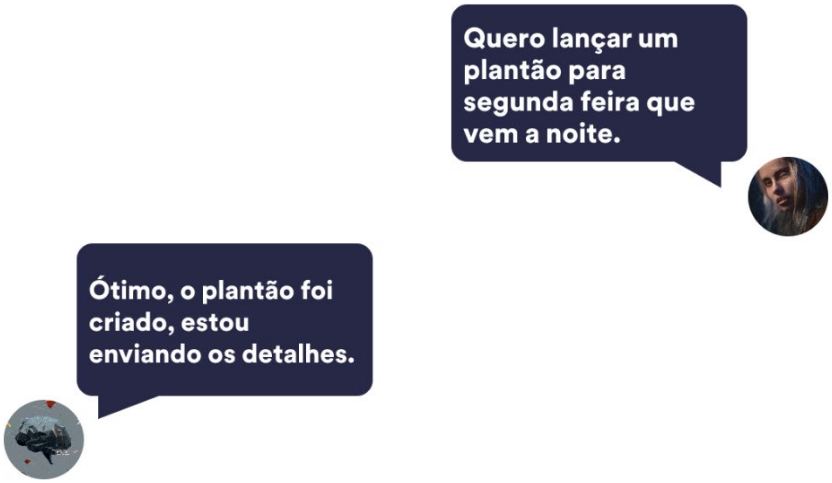


Figura 11 - Intenções podem ser extraídas em uma única interação.

3.3 Criando uma personalidade para a assistente

Os seres humanos têm uma forte tendência a dar forma ou características humanas a algo que não é humano, razão pela qual carros, barcos, e muitos outros objetos inanimados receberam nomes pelas pessoas que os usam. Os nomes têm significado e são indicadores poderosos de personalidade, caráter e identidade.

Com um chatbot ou uma assistente inteligente, não é diferente; um computador que consegue emular uma personalidade cria um sentimento de se estar em uma conversa humana, e com isso, consegue criar uma intimidade com quem está dialogando, deixando o usuário mais confortável e seguro enquanto utiliza as funcionalidades disponíveis.

Para criar a personalidade de HEVA, foi definido um cargo que a assistente atuaria e, a partir disso, foram criados fluxos de conversa a partir de diálogos de exemplo, retirados de uma fonte que demonstra a forma de agir de uma pessoa que possui esse cargo.

Após extrair os fluxos básicos, foi criada uma identidade para a assistente. Foram definidas uma idade, nome, biografia e uma identidade cultural, focadas em uma personalidade que funcione tanto com a identidade do *software* quanto com seu modelo de negócios.

Ao finalizar o estudo de personalidade e ajustar suas falas conforme sua persona, chegou-se à conclusão de que HEVA é confiante, organizada, alegre, competente em seu trabalho, adora interagir com os usuários, é extremamente pontual e confiável.

4. Desenvolvimento.

4.1 Escolha das tecnologias

A linguagem *Python* foi a escolhida para o desenvolvimento da assistente, pela sua ampla base e suporte em engenharia de dados e inteligência artificial. Como framework auxiliar utilizamos o Rasa.ai, primeiramente por ser *open source* e segundamente por superar o desempenho e acurácia das demais concorrentes em um teste de comparação de *benchmark* entre os frameworks.

4.2 Coleta e preparação dos dados

Foi utilizado como base para a criação do *dataset* simulações de conversas reais que foram coletadas através de vídeos, demonstrações e em diálogos com terceiros, totalizando 724 linhas de diálogo para o treino da assistente, e como boas práticas, foi necessário separar os arquivos para treino de NLU em diferentes arquivos nomeados de acordo com que o *cluster* de frases é destinado, como mostrado na figura:

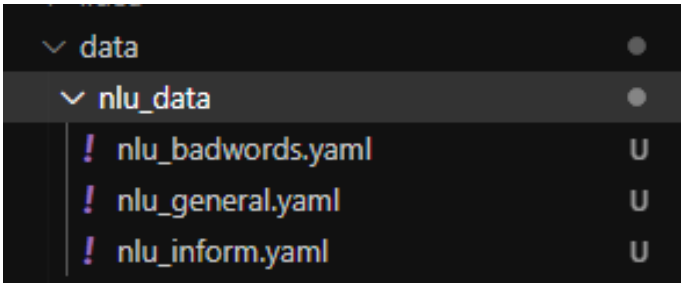
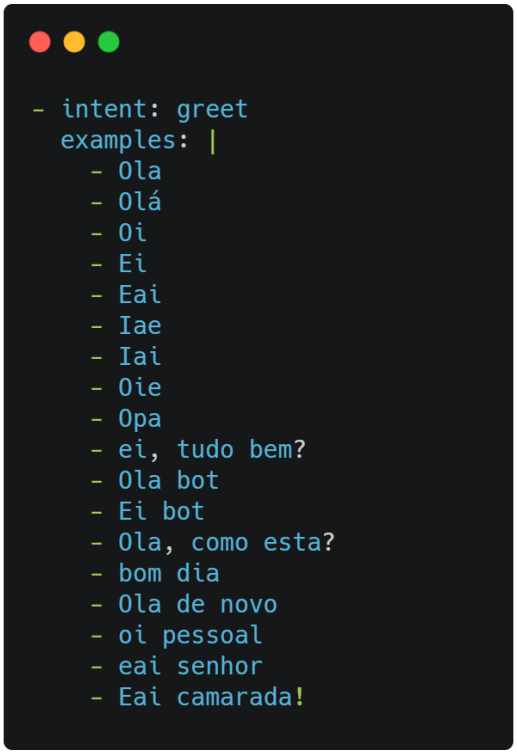
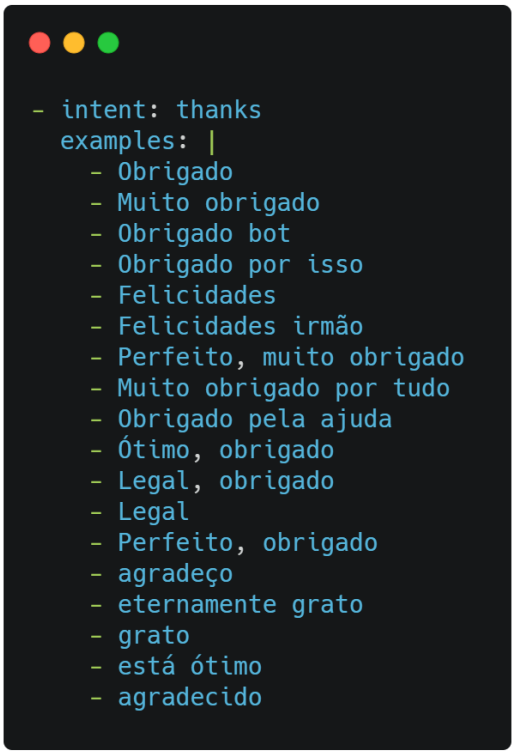


Figura 12 - Arquivos separando os datasets por coerência dos dados. Fonte: (Os autores)

Com o formato *yaml* foi possível trabalhar com listas extensas de textos, por possuir amplo suporte pela linguagem *python* e pela sua legibilidade. Seguindo o padrão usado pelo Rasa.ai, cada dado de treinamento referenciando uma intenção é criado de forma que se dê um título e logo em seguida alguns dados de exemplo, como mostra o a ilustração:



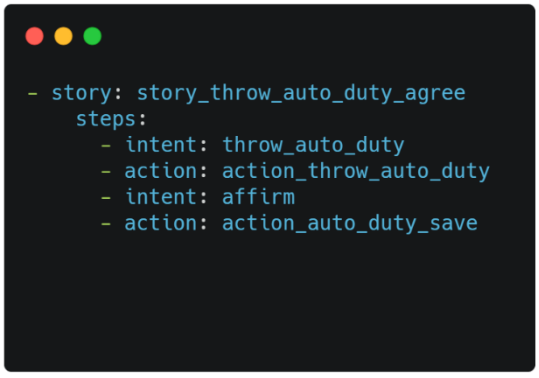
```
- intent: greet
  examples: |
    - Ola
    - Olá
    - Oi
    - Ei
    - Eai
    - Iae
    - Iai
    - Oie
    - Opa
    - ei, tudo bem?
    - Ola bot
    - Ei bot
    - Ola, como esta?
    - bom dia
    - Ola de novo
    - oi pessoal
    - eai senhor
    - Eai camarada!
```



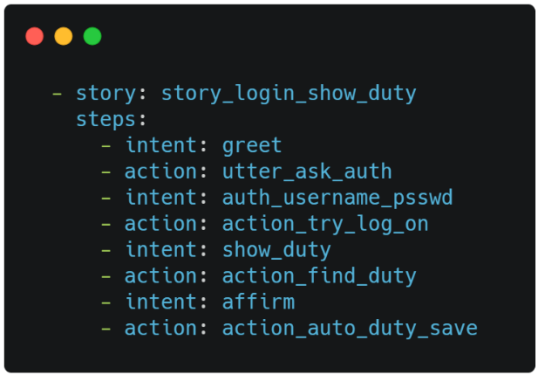
```
- intent: thanks
  examples: |
    - Obrigado
    - Muito obrigado
    - Obrigado bot
    - Obrigado por isso
    - Felicidades
    - Felicidades irmão
    - Perfeito, muito obrigado
    - Muito obrigado por tudo
    - Obrigado pela ajuda
    - Ótimo, obrigado
    - Legal, obrigado
    - Legal
    - Perfeito, obrigado
    - agradeço
    - eternamente grato
    - grato
    - está ótimo
    - agradecido
```

Figuras 13 e 15 - Formato de criação de uma intenção. Fonte: (Os autores)

Foi preciso criar também um arquivo contendo dados de treino da possível sequência de ações e interações que um usuário pode pensar em seguir, eles possuem o mesmo formato *yaml* e seguem o seguinte padrão:



```
- story: story_throw_auto_duty_agree
    steps:
      - intent: throw_auto_duty
      - action: action_throw_auto_duty
      - intent: affirm
      - action: action_auto_duty_save
```



```
- story: story_login_show_duty
  steps:
    - intent: greet
    - action: utter_ask_auth
    - intent: auth_username_psswd
    - action: action_try_log_on
    - intent: show_duty
    - action: action_find_duty
    - intent: affirm
    - action: action_auto_duty_save
```

Figura 15 e 16 - Criação de uma história de usuário para treino da assistente. Fonte: (Os autores).

4.3 Criando um pipeline para o processamento de dados e realizando o treinamento

Pipelines são os fluxos que definem como as palavras do usuário são processadas, por exemplo, uma frase pode passar por uma série de componentes, como *tokenizer*, *named entity recognizer* e um *intent classifier* antes de produzir uma distribuição sobre as possíveis intenções de diálogo.

Cada componente processa uma entrada e/ou cria uma saída. A ordem dos componentes é determinada pela ordem em que estão listados. A saída de um componente pode ser usada por qualquer outro componente que vier depois dele no pipeline. Alguns componentes produzem apenas informações usadas por outros componentes no pipeline. Outros componentes produzem atributos que são retornados após a conclusão do processamento.

Após um estudo realizado com os dados de diálogo coletados para o início do desenvolvimento da assistente, foi identificada a necessidade de entender e extrair múltiplas intenções que podem conter uma frase, com isso, o pipeline da assistente foi montado com componentes recomendados para a limpeza, extração e classificação dessas intenções.

Na figura 17 é descrito o pipeline criado que atendeu os requisitos da assistente.

```yaml
pipeline:
  - name: WhitespaceTokenizer
  - name: RegexFeaturizer
  - name: LexicalSyntacticFeaturizer
  - name: CountVectorsFeaturizer
  - name: CountVectorsFeaturizer
    intent_tokenization_flag: true
    intent_split_symbol: "+"
    analyzer: "char_wb"
    min_ngram: 1
    max_ngram: 4
  - name: DIETClassifier
    epochs: 50
  - name: EntitySynonymMapper
  - name: ResponseSelector
    epochs: 50
```

Figura 17 - Pipeline definido para as atividades da assistente. Fonte: Os autores (2020)

- *WhitespaceTokenizer*: Tokenizador usando espaços brancos como separador. Cria um *token* para cada sequência de caracteres separados por espaço em branco.
- *RegexFeaturizer*: Cria uma representação vetorial da mensagem do usuário usando expressões regulares. Para cada *regex*, um recurso será definido marcando se essa expressão foi encontrada na mensagem do usuário ou não.
- *LexicalSyntacticFeaturizer*: Cria recursos para extração de entidade. Entidades são os recursos lexicais e sintáticos para uma mensagem do usuário para oferecer suporte à extração de entidade. Move-se sobre cada token na mensagem do usuário e cria recursos de acordo com a configuração.
- *CountVectorsFeaturizer*: Cria recursos para classificação de intenções e seleção de respostas. Cria uma representação de pacote de palavras da mensagem, intenção e resposta do usuário usando o *CountVectorizer* do *sklearn*[7]. T odos os tokens que consistem apenas em dígitos (por exemplo, 123 e 99, mas não a123d) serão atribuídos ao mesmo recurso.
- *DIETClassifier*: DIET é a sigla para *Dual Intent and Entity Transformer* (do português, Transformador de Dupla Intenção e Entidade) é uma arquitetura multitarefas para classificação de intenções e reconhecimento de entidades. A arquitetura é baseada em um transformador que é compartilhado para ambas as tarefas. Além da classificação de intenções, é responsável pela extração de entidades de dupla intenção.
- *EntitySynonym*: Mapeia valores de entidades sinônimas com o mesmo valor.
- *ResponseSelector*: Os seletores preveem uma resposta do bot a partir de um conjunto de respostas candidatas. O componente Seletor de resposta pode ser usado para construir um modelo de recuperação de resposta para prever diretamente uma resposta do bot a partir de um conjunto de respostas candidatas.

Para cada atualização no *dataset* foi executado um novo treinamento, após cada treinamento, os dados são comprimidos e salvos em uma pasta models, no qual o nome do arquivo contém a data e hora em que o treinamento foi realizado como forma de backup e checkpoint para controle de mudanças, conforme mostrado na figura abaixo:

[7] *Sklearn ou scikit-learn* é uma biblioteca Python que implementa ferramentas simples e eficientes para predição analítica de dados e ML.

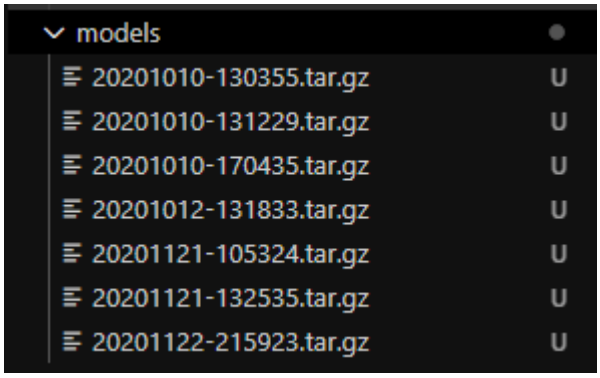


Figura 18 - Histórico de treinamento da assistente Fonte: (Os autores).

Os arquivos para treino foram otimizados de forma que não atrapalhasse o treinamento final em quesito de performance e acurácia da IA. Cada ciclo de treinamento da assistente dura em média 5 minutos. Como resultados do treinamento final, foi obtido um percentual de 97% de acurácia e um percentual de 2% de *loss,* conforme mostram as imagens abaixo.

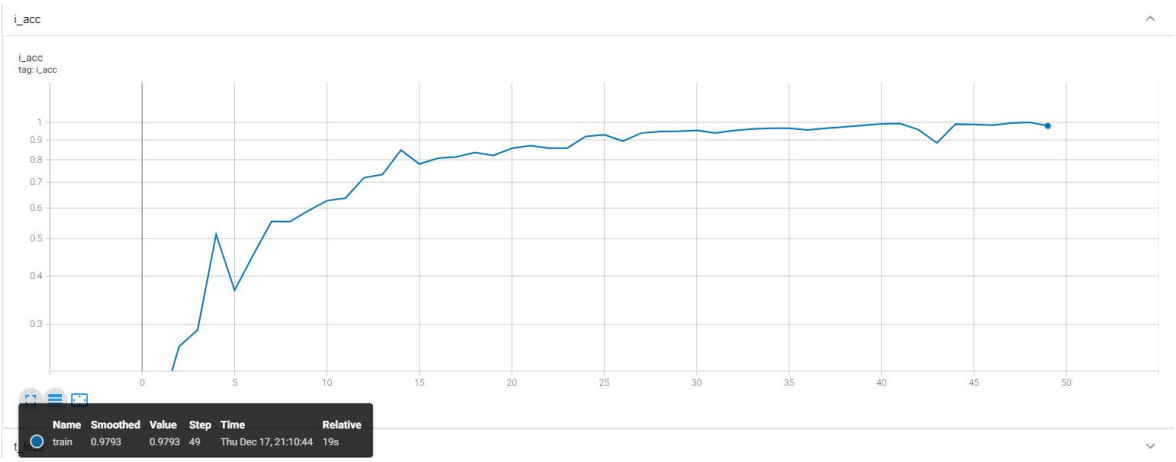


Figura 19 – Acurácia do treinamento. Fonte: (Os autores).

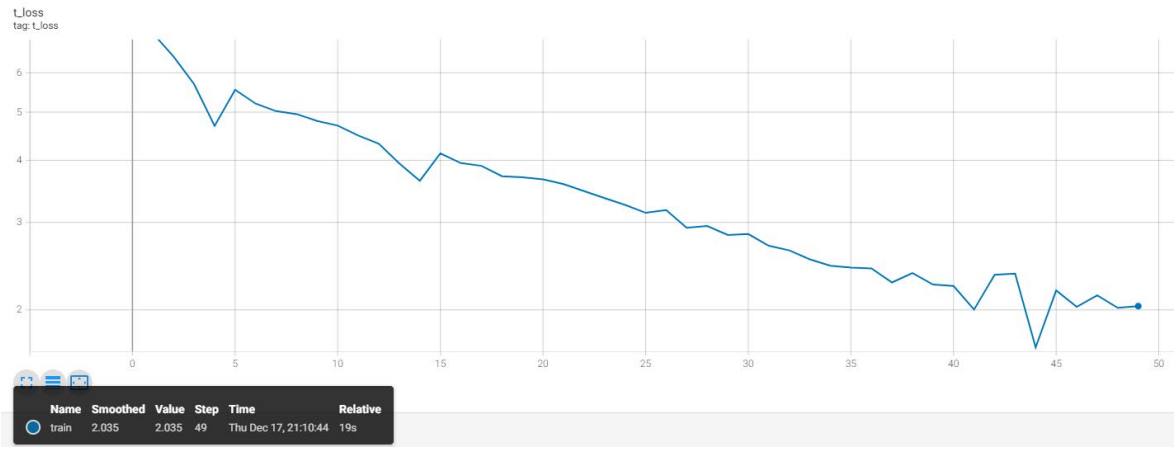


Figura 20 – Loss do treinamento. Fonte: (Os autores).

4.4 Comunicação com a API do CyclePlantões

As ações executadas pela assistente possuem sua lógica mantida em classes dentro de um arquivo *actions.py,* onde a assistente instancia e executa de acordo com que uma intenção é detectada. As ilustrações a seguir mostram como foram realizadas o processo de autenticação e buscas através da biblioteca *requests* do *python*:

```python
class ActionTryLogOn(Action):
    def name(self):
        return "action_try_log_on"

    def run(self, dispatcher, tracker, domain):

        try:

            data = {
                'usuario': tracker.slots.get('username'),
                'password': tracker.slots.get('password'),
            }

            r = requests.post('https://teste.cycleplantoes.com.br/api/login', data=data).json()
            dispatcher.utter_message(template='utter_info_auth')

        except:
            dispatcher.utter_message(template='utter_auth_failed')
            return [FollowupAction(name='utter_auth_failed_details')]

        return [
            SlotSet("name", r.get('user')['name']),
            SlotSet("user_token", r.get('token'))
        ]
```

Figura 21 – Ação de tentativa de login. Fonte: (Os autores)

Ao identificar um usuário e senha como entidades em uma intenção de autenticação, a assistente irá executar a ação *ActionTryLogOn,* que envia uma requisição ao *endpoint* de autenticação do CyclePlantões e armazena o nome do usuário e o token de autenticação do usuário caso.

```python
class ActionFindDuty(Action):
    def name(self):
        return "action_find_duty"

    def run(self, dispatcher, tracker, domain):
        if tracker.slots.get('user_token'):
            data = {
                'usuario': tracker.slots.get('username'),
                'password': tracker.slots.get('password'),
            }

            new_token = tracker.slots.get('user_token')

            headers = {"content-type": "application/json; charset=UTF-8",
                       'Authorization': 'Bearer {}'.format(new_token)}

            dispatcher.utter_message("Aguarde um instante! Estou procurando pelo seu plantão 😉")

            r = requests.get('https://teste.cycleplantoes.com.br/api/plantoes', data=data,
headers=headers)

            duty = r.json()['proximos_plantoes'][-1]

            info_message = "Seu próximo plantão será no dia {day}, das {time_ini} até as {time_end},
totalizando {total_time} horas na unidade {unity}".format(
                day=duty.get('data_plantao'),
                time_ini=duty.get('ini_plantao'),
                time_end=duty.get('fim_plantao'),
                total_time=duty.get('horas_totais'),
                unity=duty.get('nome_unidade')
            )
            dispatcher.utter_message(info_message)

            if duty.get('troca_plantao') in 'S':
                dispatcher.utter_message("Esse plantão foi trocado com: {nome}")

            return []

        return [FollowupAction(name='utter_no_auth')]
```

Figura 22 - Classe que busca o próximo plantão de um usuário: (Os autores).

Ao identificar a intenção de encontrar o próximo plantão, será checado se o usuário já está autenticado, caso não, a assistente pedirá para o usuário *logar*, caso sim, é feita uma requisição para buscar pelos próximos plantões do usuário, ao recuperar as informações, é salvo e retornado o item relacionado ao próximo plantão.

4.5 *Roadmap*

Como futuras implementações e melhorias a serem implementadas à assistente, foram apontados como prioritários um filtro de palavrões, onde a assistente irá identificar e responder a palavrões, o aprendizado em tempo de execução, no qual a assistente irá

aprender novas intenções de acordo com o diálogo com usuários e uma melhor autenticação de usuário, que visa implementar uma forma mais segura para autenticar um usuário.

Por ainda estar em desenvolvimento, a assistente não foi implementada em uma plataforma de *chat*, como o *Telegram* ou o *Whatsapp*, que são algumas das plataformas alvo da assistente. Para os testes e validações das interações da assistente foi utilizado o *rasa shell,* uma plataforma para testes feita pelo próprio Rasa.ai que roda em terminal.

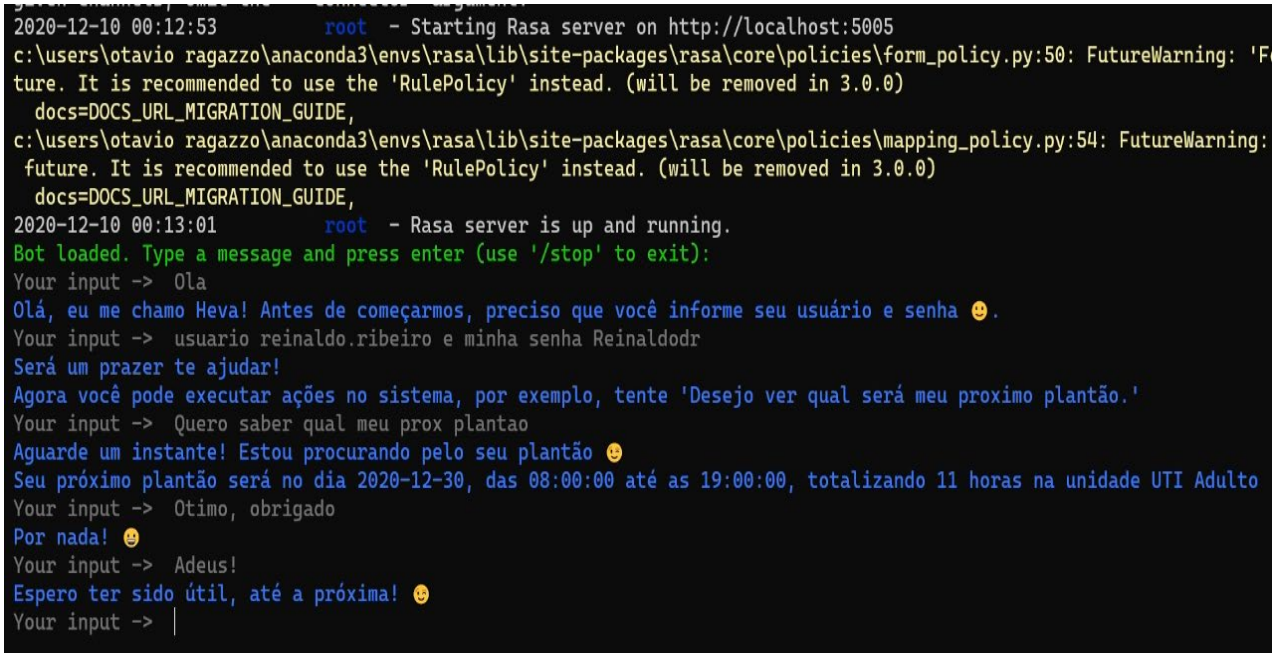


Figura 23 – Simulação de um caminho de conversação feliz. Fonte: (Os autores).

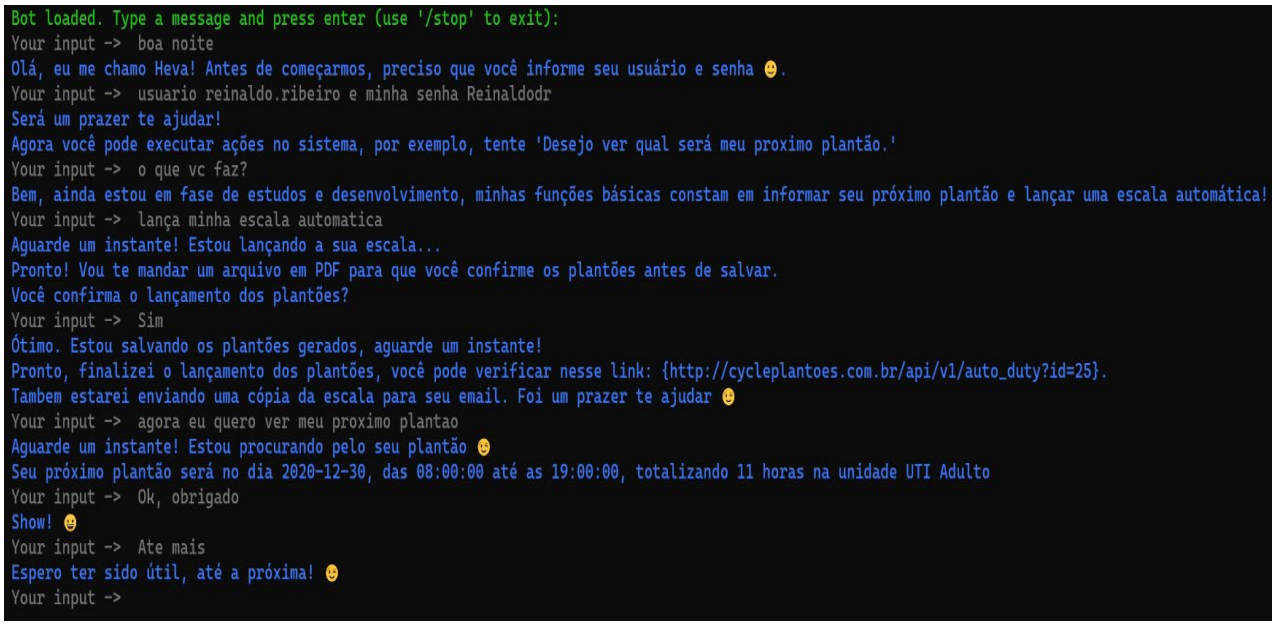


Figura 24 - Simulação de um caminho de conversação infeliz. Fonte: (Os autores).

5. Considerações Finais

O *dataset* foi criado totalmente pensando em padrões de conversação humanas, de forma que a assistente entenda e atenda as intenções de um usuário tanto em uma conversa objetiva, que vai direto ao ponto quanto em uma conversa que o usuário saia do contexto da tarefa que está sendo requerida.

Heva conseguiu facilmente realizar uma requisição para a API do CyclePlantões, enviando como parâmetros, as entidades coletadas das frases dos usuários, como por exemplo, data e turno e para qual médico será criado um plantão.

Os códigos de NLP (Processamento de Linguagem Natural) e NLU (Entendimento de Linguagem Natural) permitiram criar uma assistente humanizada que consegue emular uma conversa humana, absorvendo e entendendo a intenção da fala de um usuário e a partir disso, executar a ação desejada.

Referências Bibliográficas

AYANOUZ, Soufyane e ANOUAR ABDELHAKIM, Boudhir e BENHMED, Mohammed. (2020). **A Smart Chatbot Architecture based NLP and Machine learning for health care assistance.** Association for Computing Machinery. ACM ISBN.

BRANIGAN, H. P., PICKERING, M. J. Pearson, J. and MCLEAN, J.F. 2010. **Linguistic alignment between people and computers.** Journal of Pragmatics, 42, 2355--2368.

CAHN, Jack. CHATBOT: **Arquitetura, Design e Desenvolvimento.** University of Pennsylvania - School of Engineering and Applied Science, 2017. Reimpressão.

BABICH, Nick. **O que você deve saber sobre a experiência do usuário.** Adobe Blog. 26 de jun. de 2017. Disponível em: https://blog.adobe.com/en/publish/2017/06/26/what-is-ux-and-why-should-you-care.html#gs.l3exgz. Acesso em 15 de nov. de 2020.

BABICH, Nick. **O que um UX Designer realmente faz?** Adobe Blog. 25 de ago. de 2017. Disponível em: https://blog.adobe.com/en/2017/08/25/what-does-a-ux-designer-actually-do.html#gs.l3i9yf. Acesso em 15 de fev. de 2020.

BABICH, Nick. **10 previsões de design UX para 2018.** Adobe Blog. 03 de nov. de 2017. Disponível em: https://blog.adobe.com/en/publish/2017/11/03/10-ux-design-predictions-for-2018.html#gs.l3hsgq. Acesso em 15 de nov. de 2020.

BOZKURT, Aras. **A tecnologia se renova:** conceitos-chave sobre assistentes pessoais inteligentes (IPAs). Anadolu: Anadolu University, 2018. Reimpressão.

CHAKRAVORTY, Sandeep e CHAKRABARTI, Amrita. 2009. ***Hidden layer optimization of neural network using computational technique. In Proceedings of the International Conference on Advances in Computing, Communication and Control (ICAC3 '09).*** Association for Computing Machinery, New York, NY.

MILAJEVS, Dmitrijs and SADRZADEH, Mehrnoosh and ROELLEKE, Thomas. 2015. ***IR meets NLP:*** *On the Semantic Similarity between Subject-Verb-Object Phrases. In Proceedings of the 2015 International Conference on The Theory of Information Retrieval (ICTIR '15).* Association for Computing Machinery, New York, NY, USA, 231–240.

DICIO, Dicionário online. **Assistente**. Acesso em 11 de maio de 2020. Disponível em: dicio.com.br.

EA072 – Prof. Fernando J. Von Zuben DCA/FEEC/Unicamp – **Introdução à Inteligência Artificial.**

GURNEY, Kevin. **Uma introdução às redes neurais.** Londres: Taylor e Francis e-Library, 2004.INFOJOBS, Assistente de Telemarketing. Acesso em 11 de maio de 2020. Disponível em: www.infojobs.com.br/artigos/Assistente_de_Telemarketing_Ativo__2111.aspx

HOCHREITER, Sepp e SCHMIDHUBER, Jürgen. (1997). ***Long Short-term Memory.*** Neural computation. 9. 1735-80. 10.1162/neco.1997.9.8.1735.

INSTITUITO DE ENGENHARIA, **A História da Inteligência Artificial.** Disponível em: www.institutodeengenharia.org.br/site/2018/10/29/a-historia-da-inteligencia-artificial.

KHURANA, Diksha et al. **Processamento de linguagem natural:** estado da arte, tendências atuais e desafios. Department of Computer Science and Engineering, Manav Rachna International University, 2017.

KIRANYAZ, Serkan and AVCI, Onur and ABDELJABER, Osama and INCE, Turker and GABBOUJ, Moncef and INMAN, Daniel. (2019). ***1D Convolutional Neural Networks and Applications:*** *A Survey*.

Leôncio Teixeira Cruz e Antonio Juarez Alencar, Eber Assis Schmitz. (2013). **Assistentes Virtuais Inteligentes Conceitos e Estratégicas.**

URBAN, Margaret and MAILEY, Stephen. 2019. ***Conversation Design:*** *Principles, Strategies, and Practical Application. In Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems (CHI EA '19).* Association for Computing Machinery, New York, NY, USA, Paper C26, 1–3.

NICHOL, Alan. **Uma nova abordagem para o software de conversação.** RasaAI Blog. 04 de out. de 2017. Disponível em: https://blog.rasa.com/a-new-approach-to-conversational-software. Acesso em: 01 de set. de 2019.

OLAH, Christopher. ***Understanding LSTM Networks.*** Colah's Blog. 27 de ago. 2015. Disponível em: https://colah.github.io/posts/2015-08-Understanding-LSTMs/. Acesso em: 20 de out. de 2020.

OurCycle. **CyclePlantões**. Acesso em 1 de outubro de 2020. Disponível em: http://cycleplantoes.com.br/about.html

SHARMA, Avinash. ***Understanding Activation Functions in Neural Networks.*** Medium. 22 de jun. de 2020. Disponível em: https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0. Acesso em 30 de mat. de 2017.

SHAWAR, Bayan e ATWELL, Eric. 2007. ***Chatbots: Are they Really Useful?*** LDV Forum. 22. 29-49.

SHERSTINSKY, A. (2018). **Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network.** ArXiv, abs/1808.03314.

SIMON, Annina e DEO, Mahima e SELVAM, Venkatesan e BABU, Ramesh. 2016. ***An Overview of Machine Learning and its Applications. International Journal of Electrical Sciences e Engineering.*** Volume. 22-24.

STEVEN Bird and EWAN Klein, and EDWARD, Loper. 2009. ***Natural Language Processing with Python:*** *Analyzing Text with the Natural Language Toolkit. O'Reilly*

Media, Inc. 2009. Proceedings of the International Conference on Advances in Computing, Communication and Control. Association for Computing Machinery, New York, NY, USA.

RADFORD, Alec et al. 2018. ***Improving Language Understanding by Generative Pre-Training*.** OpenAI.

ROOS, Matthew. ***Deep Learning Neurons versus Biological Neurons*.** Medium, Mar 14, 2019. Disponível em: Acesso em: 01 de maio de 2020.

RUSSEL, Stuart J. (Stuart Jonathan), 1962 **Inteligência artificial** / Stuart Russell, Peter Norvig; tradução Regina Célia Simille. – Rio de Janeiro: Elsevier, 2013

HIGASHINAKA, Ryuichiro et al. 2014. ***Towards an open domain conversational system fully based on natural language processing.*** In COLING'14.

SHARMA, Sagar. ***"What the Hell is Perceptron?"***. Medium, Sep 9 2017. Disponível em: Acesso em: 05 de maio de 2020.

WALLACE, R. (2003). ***The Elements of AIML Style. A.L.I.C.E. Artificial Intelligence Foundation***, Inc.

WALLACE, R. and TOMABECHI, H. and AIMLESS, D. (2003). ***Chatterbots go native:*** *Considerations for an eco-system fostering the development of artificial life forms in a human world*. Publicado online: pandorabots.com/pandora/pics/chatterbotsgonative.doc.

ZADROZNY, W., BUDZIKOWSKA, M., CHAI, J., and KAMBHATLA, N. (2000). ***Natural language dialogue for personalized interaction*.** Communications of the ACM.