UNIVERSIDADE FEDERAL FLUMINENSE

ALEXANDRE LUCAS GOMES

IMPLEMENTAÇÃO DE CHATBOT COMO INTERFACE PARA OBTER INFORMAÇÕES SOBRE EVENTOS

Niterói

2018

ALEXANDRE LUCAS GOMES

IMPLEMENTAÇÃO DE CHATBOT COMO INTERFACE PARA OBTER INFORMAÇÕES SOBRE EVENTOS

Trabalho de Conclusão de Curso submetido ao Curso de Graduação em Ciência da Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Bacharel em Ciência da Computação.

Orientador:

José Viterbo

NITERÓI

2018

ALEXANDRE LUCAS GOMES

IMPLEMENTAÇÃO DE CHATBOT COMO INTERFACE PARA OBTER INFORMAÇÕES SOBRE EVENTOS

Trabalho de Conclusão de Curso submetido ao Curso de Graduação em Ciência da Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Bacharel em Ciência da Computação.

Niterói, \_\_\_ de \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ de 2018.

Banca Examinadora:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

José Viterbo – Orientador

UFF - Universidade Federal Fluminense

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Nome do Avaliador 1

UFF - Universidade Federal Fluminense

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Nome do Avaliador 1

UFRJ – Universidade Federal do Rio de Janeiro

AGRADECIMENTOS

A Deus, que sempre iluminou a minha caminhada.

A meu Orientador José Viterbo pelo estímulo e atenção que me concedeu durante o curso.

Aos Colegas de curso pelo incentivo e troca de experiências.

A todos os meus familiares e amigos pelo apoio e colaboração.

RESUMO

É apresentada uma implementação de um chatbot para informar sobre eventos, palestras e palestrantes, implementada na linguagem Python utilizando a plataforma Telegram como interface com o usuário.

O proposito desse trabalho é criar um chatbot para Telegram usando a linguagem de programação Python e a biblioteca Rasa NLU para processamento de linguagem natural e aprendizado de máquina.

Palavras-chaves: Chatbot, Inteligência artificial, Processamento de Linguagem Natural

LISTA DE ILUSTRAÇÕES

[Figura 1: Máquina de estados de um sistema de viagens 22](#__RefHeading___Toc516450578)

[Figura 2: Exemplo dos dados usados para treinamento 27](#__RefHeading___Toc516450579)

[Figura 3: Exemplo de sinônimo 27](#__RefHeading___Toc516450580)

[Figura 4: Texto com sua intenção e entidade 27](#__RefHeading___Toc516450581)

[Figura 5: Template DSL Chatito 29](#__RefHeading___Toc516450582)

[Figura 6: Exemplo do resultado da identificação 32](#__RefHeading___Toc516450583)

SUMÁRIO

[RESUMO 5](#__RefHeading___Toc517342688)

[LISTA DE ILUSTRAÇÕES 6](#__RefHeading___Toc517342689)

[SUMÁRIO 7](#__RefHeading___Toc517342690)

[1 INTRODUÇÃO 9](#__RefHeading___Toc517342691)

[1.1 Definição do problema 10](#__RefHeading___Toc517342692)

[1.2 Objetivo 10](#__RefHeading___Toc517342693)

[1.3 Organização do trabalho 10](#__RefHeading___Toc517342694)

[2 Agentes conversacionais 11](#__RefHeading___Toc517342695)

[2.1 Sistema e agentes conversacionais 11](#__RefHeading___Toc517342696)

[2.2 Histórico 12](#__RefHeading___Toc517342697)

[2.2.1 Primeira Geração 12](#__RefHeading___Toc517342698)

[2.2.2 Segunda Geração 13](#__RefHeading___Toc517342699)

[2.2.3 Terceira geração 13](#__RefHeading___Toc517342700)

[2.3 Tipos de agentes conversacionais 14](#__RefHeading___Toc517342701)

[2.3.1 Chatbots baseados em regras 14](#__RefHeading___Toc517342702)

[2.3.2 Chatbots baseados em corpus 15](#__RefHeading___Toc517342703)

[2.3.3 Chatbots baseados em Recuperação de informação (RI) 15](#__RefHeading___Toc517342704)

[2.3.4 Agentes de diálogo baseados em frame 16](#__RefHeading___Toc517342705)

[3 Aplicações de agentes conversacionais 18](#__RefHeading___Toc517342706)

[4 O HOST-BOT 18](#__RefHeading___Toc517342707)

[4.1 Requisitos de Negócios 20](#__RefHeading___Toc517342708)

[5 Implementação 21](#__RefHeading___Toc517342709)

[5.1 Ferramentas utilizadas 21](#__RefHeading___Toc517342710)

[5.1.1 Telegram 21](#__RefHeading___Toc517342711)

[5.1.2 Telepot 22](#__RefHeading___Toc517342712)

[5.1.3 Bibliotecas 22](#__RefHeading___Toc517342713)

[5.2 Comunicação com o usuário 22](#__RefHeading___Toc517342714)

[5.3 Processamento de linguagem natural 23](#__RefHeading___Toc517342715)

[5.4 Treinamento do modelo 24](#__RefHeading___Toc517342716)

[5.5 Criação dos dados 24](#__RefHeading___Toc517342717)

[5.6 Domínio 25](#__RefHeading___Toc517342718)

[5.7 Pipeline 26](#__RefHeading___Toc517342719)

[5.8 Execução 27](#__RefHeading___Toc517342720)

[6 Avaliação da ferramenta 29](#__RefHeading___Toc517342721)

[7 Conclusão 30](#__RefHeading___Toc517342722)

[Referências bibliográficas 31](#__RefHeading___Toc517342723)

# INTRODUÇÃO

Comunicação por mensagens de texto é bastante usada nos dias de hoje, sendo uma das funcionalidades mais usadas por pessoas que acessam a Internet. Esse tipo de comunicação provê acesso a contatos ao redor do globo permitindo troca de informação e interação em tempo real. Indivíduos dos mais diversos grupos sociais e etários utilizam mensagens de texto para se comunicar com familiares, amigos e clientes. No Brasil, cerca de 85% dos jovens de 18 a 24 anos de idade e 25% das pessoas com mais de 60 anos usam a Internet, sendo o telefone móvel o equipamento mais usado para acessar a Internet contando com o uso por 94,6% das pessoas que acessam a Internet (IBGE, 2016). Existem mais de 3,2 bilhões de contas de mensageiros instantâneos, sem contar contas de mensageiros instantâneos de dispositivos móveis (“Instant Messaging Statistics Report, 2015-2019”, 2015). A comunicação por mensagem de texto pode ser bastante útil para pessoas mais idosas por suas dificuldades em usar interfaces mais convencionais.

Na área empresarial, comunicação por mensagens de texto podem ser usadas para fazer comprar, pagar contas e fazer transações bancárias. Nos Estados Unidos, 66% dos consumidores usam agentes de texto ou de voz em seu dia-a-dia e 21% usam para realizar as operações supracitadas (TAYLOR, 2018). Algumas vantagens citadas para o uso de texto nesses casos são a precisão do texto em relação à voz e a privacidade de realizar as tarefas enquanto há pessoas por perto. Empresas também fazem uso dessa ferramenta para comunicação interna, usando ferramentas como Slack (SLACK, [s.d.]) e WhatsApp (“WhatsApp”, [s.d.]) que permite o uso de comunicação em grupo entre funcionários da empresas e auxilia na comunicação com clientes.

Comunicação por mensagens de texto também possuem outras aplicações como no setor de turismo (SHIH; FAN, 2015), o que permite que agências de viagem interajam melhor com seus clientes tirando qualquer tipo de dúvida sobre preços, disponibilidades e serviços, e no setor educacional, permite maior interação entre alunos e professores, o que acarretou maior interesse nos estudos e melhor aproveitamento da matéria estudada (BOUHNIK; DESHEN, 2014).

Uma das ferramentas que podem ser usadas por meio de mensagens de texto é o agente conversacional ou chatbot. Chatbots são sistemas de diálogo que se comunicam por linguagem natural (JIA, 2003). O primeiro pensamento de um chatbot foi dado por Alan Turing em seu artigo “Can machine think?” (TURING, 1950). Desde a publicação desse artigo de Turing, a tecnologia por trás dos chatbots teve grandes avanços tanto no processamento de linguagem natural quanto em aprendizado de máquina o que permitiu melhor aproveitamento desse tipo de sistema. Muitos outros programas similares foram criados desde então. Sistemas de diálogo têm sido usados em diversas tarefas: auxiliar na compra de produto, atendimento de cliente de empresa, responder perguntas sobre determinado domínio, guiar uma pessoa pela navegação de site.

Agentes conversacionais são cada vez mais comuns em nosso dia-a-dia. Exemplos dessas aplicações são Siri, da Apple, Cortana, da Microsoft, Alexa, da Amazon. Empresas como Pizza Hut e HBO criaram chatbots como uma forma dos clientes interagirem com seus serviços, fazerem pedidos e tirar dúvidas (VentureBeat, 2016). Ocorreu também um aumento do número de ferramentas para interagir com esses agentes, como Facebook (“How To Build Bots for Messenger”, [s.d.]), Skype (“What are Skype bots and how do I add them as contacts?”, [s.d.]), and Telegram (“Telegram Bot Platform”, [s.d.]).

## Definição do problema

Existem algumas formas para obter informações sobre os eventos desejados, por exemplo, falando com pessoas ou buscando em sites. No entanto, nem sempre essas pessoas estão disponíveis para responder as perguntas desejadas ou as informações dos eventos desejados estão dispersas por diversos sites. Outra questão é que se pode preferir usar mensagens de texto para se comunicar. De que forma um chatbot pode ajudar a tornar o acesso a informações sobre eventos e palestras sempre disponível e mais prático de se acessar por meio de mensagens de texto?

## Objetivo

O objetivo desse trabalho é criar um chatbot usando a plataforma Telegram para informar as pessoas sobre eventos, palestras e palestrantes auxiliando tanto aqueles que querem conhecer novos eventos para participar quanto aqueles que querem obter informações sobre as palestras e palestrantes do evento que vão participar.

Esse objetivo desdobra-se em:

Estudar a API de envio de mensagens por bots do Telegram.

Estudar as ferramentas de processamento de linguagem natural e aprendizado de máquina.

Projetar um chatbot que seja capaz de apresentar informações relativas a eventos, palestras e palestrantes organizadas de forma adequada.

Implementar e testar um protótipo do chatbot projetado.

## Organização do trabalho

O Capítulo 2 descreve o que são agentes conversacionais e seus tipos, traçando um histórico deles. O Capítulo 3 exemplifica aplicações de agentes conversacionais. O Capítulo 4 apresenta a modelagem do chatbot. O Capítulo 5 apresenta detalhes da implementação. O Capítulo 6 descreve a avaliação feita. Capitulo 7 encerra este trabalho com as conclusões obtidas.

# Agentes conversacionais

## Sistema e agentes conversacionais

Agentes conversacionais (ou chatbots) são programas de computador que se comunicam com usuários por meio de linguagem natural (SHAWAR; ATWELL, 2007) e são projetados para interpretar a interação do usuário e reagir a essa interação com uma resposta buscando manter a coerência do diálogo dando a ilusão que se está conversando com outro ser humano decidindo pela melhor ação disponível na situação em que se encontra. Devido a isso são considerados agentes inteligentes. (RUSSELL; NORVIG; DAVIS, 2010). Eles podem ser projetados para realizar uma tarefa particular e por isso realizam um diálogo com o usuário para obter as informações necessárias para ajudar a completar uma tarefa. Ex.: Siri, Cortana. Ou também podem ser projetados para ter conversas mais extensas, imitando uma conversa humana em vez de realizar uma tarefa específica. Eles podem ser usados para responder a perguntas feitas em websites de vendas, tirar duvidas sobre investimentos, ajudar alunos a estudarem e também na área de entretimento de jogos eletrônicos.

A arquitetura de chatbots pode ser dividida em dois grupos (MARTIN; JURAFSKY, 2017): baseada em regras e baseada em corpus. Chatbots baseados em regras são aqueles que usam regras construídas manualmente para lidar com a conversa. Chatbots baseados em corpus usam grande quantidade de dados de conversas para aprender a associar uma fala do usuário com uma resposta para ele.

A comunicação de um chatbot se dá por meio de um diálogo que é composto de múltiplos turnos onde em cada turno os participantes fazem sua contribuição a conversa e passam a oportunidade de interação para o outro. Um turno pode ser uma sentença, uma palavra, ou múltiplas sentenças.

Dois motivos levaram ao crescimento de agentes conversacionais: primeiro, a expansão da web como forma de comunicação. Agentes que usam a web podem alcançar um grande um número de pessoas com seus serviços. Segundo, a linguística computacional, que é um campo da inteligência artificial que foca em software de linguagem natural, teve grandes avanços. Por exemplo, melhorias na tecnologia de parsing aumentaram consideravelmente o a capacidade de programas entenderem linguagem natural (LESTER; BRANTING; MOTT, 2004).

## Histórico

É possível dividir a história dos chatbots em três gerações em relação às técnicas utilizada para conversar com o usuário (Neves, 2005).

O primeiro chatbot de maior reconhecimento da primeira geração foi ELIZA (Weizenbaum, 1966) que simulava um psicanalista da rogeriano e usava casamento de padrão para escolher a resposta a partir da mensagem recebida do usuário.

A segunda geração (inicio de 1990) passou a usar técnicas vindas das áreas de AI e PLN (Processamento de Linguagem Natural) para melhorar os diálogos.

A partir dessa geração, chatbots passam a contar com modelo de processamento morfossintático para identificar variações morfológicas e sintáticas das frases. Outro avanço foi a inclusão de bases de conhecimento de domínio especifico (Neves, 2005).

O chatbot que merece destaque dessa geração é CHATTERBOT (MAULDIN, 1994), desenvolvido para ser usado em um ambiente de entretenimento multiusuário de 1989, TinyMUD que permitia conversa entre vários usuários e criação de ambientes onde ocorriam as interações.

CHATTERBOT interagia com outras pessoas por meio de salas de chat e possuía módulos para lidar com a comunicação com a plataforma, guardava o estado das várias salas e objetos da plataforma, armazenava mensagens de usuários, e o módulo que o permitia fazer a conversa propriamente dita. Ele conseguia guardar informações sobre o usuário, com o objetivo dar maior credibilidade ao diálogo.

A terceira geração (final da década de 1990) apresenta arquitetura mais modular, uso de PLN e uso de linguagens de marcação como XML (eXtensible Markup Languages).

Um exemplo de chatbot desse período é A.L.I.C.E. (WALLACE, 2009), que usava a linguagem de marcação AIML (Artificial Intelligence Markup Language). Apesar da semelhança de A.L.I.C.E. com ELIZA em relação a arquitetura estimulo-resposta, A.L.I.C.E. apresentava mais de 40000 categorias de conhecimento e ainda contava com um conjunto de dados de linguagem natural coletados da web. Ele usava o modelo de aprendizado supervisionado, onde uma pessoa monitora as conversas que ocorreram e cria mais conteúdo AIML para tornar as conversas melhores atendendo a critérios desejados.

## Tipos de agentes conversacionais

### Chatbots baseados em regras

Os chatbots desse tipo usam técnicas simples para manter diálogo, usando batimento de padrão e de palavras chaves. Eles utilizam pares <Estimulo-Resposta> para conversar. Ao receber uma mensagem do usuário, é buscado o padrão mais próximo para então mostrar a resposta associada a esse padrão.

Dois exemplos de chatbots desse tipo foram ELIZA (WEIZENBAUM, 1966) e PARRY (Colby et al., 1972) que simulava um paciente com paranoia.

### Chatbots baseados em corpus

Chatbots baseados em corpus extraem as informações de conversas entre pessoas ou entre pessoas e máquinas. As respostas desses chatbots também podem ser extraídas de textos que não são conversas, por exemplo, textos sobre determinado assunto.

A maioria dos chatbots baseados em corpus costuma fazer pouco uso do contexto da conversa dando mais ênfase em gerar uma resposta apropriada para cada fala recentemente dita pelo usuário. Por esse motivo eles são chamados de sistemas de geração de resposta. Esses chatbots tem similaridade com os sistemas de pergunta e resposta, que respondem exatamente a pergunta feita ignorando outros objetivos da conversa. (MARTIN; JURAFSKY, 2017)

### Chatbots baseados em Recuperação de Informação (RI)

Esse tipo de chatbots gera uma resposta ao usuário repetindo alguma informação extraída de um texto fonte. A diferença entre as implementações desses chatbots está em qual fonte é utilizado e como decidir qual informação é será usada como resposta.

Dado o corpus e a sentença do usuário, sistemas baseados em RI usam um algoritmo de recuperação para escolher uma resposta apropriada de um corpus. Os métodos mais usados estão descritos a seguir:

1. Retornar a resposta para o turno mais similar: A ideia é que devemos olhar para o turno que mais se assemelha ao turno do usuário e retornar a resposta a esse turno (JAFARPOUR; BURGES, 2009).
2. Retornar o turno mais similar: A ideia aqui é casar a mensagem vinda do usuário com turnos do corpus, já que uma boa resposta geralmente vai compartilhar palavras ou semântica que o turno anterior.

Em cada um desses métodos, qualquer função de similaridade pode ser usada, como cosseno sobre palavras.

Retornar a reposta ao turno mais similar pode parecer o método mais intuitivo, mas retornar o turno mais similar costuma demonstrar melhores resultados na prática (RITTER; CHERRY; DOLAN, 2011).

A abordagem de recuperação de informação pode ser expandida usando outras informações além das palavras na consulta feita, como palavras usadas em turnos anteriores ou informações sobre o usuário.

### Agentes de diálogo baseados em frame

Sistemas de diálogo modernos orientados a tarefas são baseados em ontologia de domínio. A ontologia define um ou mais frames, sendo cada um uma coleção de slots, e define os valores que cada slot pode receber. Essa arquitetura baseada em frames foi introduzida em 1977 com o sistema GUS para planejamento de viagem (BOBROW et al., 1977).

Frame é uma estrutura de dados que pode conter um nome, uma referência a um protótipo de frame e um conjunto de slots. É uma representação coleções de informações dentro do sistema (BOBROW et al., 1977).

O conjunto de slots do frame GUS especifica o que o sistema precisa saber, e o conteúdo de cada slot é restringido aos valores de um tipo particular. No domínio de viagem, um slot pode ser do tipo cidade, data ou hora da partida.

A estrutura dos tipos desse agente permite que cada tipo contenha outros tipos, por exemplo, o tipo data pode ser composto por slots de número inteiros.

A arquitetura de controle de sistema de dialogo baseados em frame é projetada ao redor do frame. O objetivo é preencher os slots no frame com o conteúdo que o usuário informou e então realizar a ação desejada pelo usuário. A maioria dos sistemas de dialogo baseados em frame são baseados em um autômato de estado finito que é montado para a tarefa por um projetista de diálogo.

O sistema GUS implementa um sistema de viagem de avião que pede ao usuário informação para 4 slots: cidade de partida, cidade de destino, hora e se a viagem é só de ida ou de ida e volta. Associada a cada slot está uma pergunta:

|  |  |
| --- | --- |
| Slot | Pergunta |
| Cidade de origem | “De qual cidade você está partindo?” |
| Cidade de destino | “Para qual cidade você está indo?” |
| Hora de partida | “Quando você quer partir?” |
| Hora de chegada | “Quando você quer chegar?” |

O sistema faz perguntas para o usuário, preenchendo assim os slots com as informações obtidas na conversa.

Assim que sistema obtiver informação suficiente ele consulta o banco de dados para obter informações sobre os voos disponíveis.

O objetivo do entendimento de linguagem natural é extrair três coisas da fala do usuário: classificação do domínio, intenção e preenchimento de slot.

Classificação de domínio: identificar sobre qual domínio o usuário está falando.

Intenção: identificar o objetivo do usuário com o dialogo.

Preenchimento de slot: extrair da fala do usuário o conteúdo dos slots de acordo com a intenção demonstrada na conversa.

# Aplicações de agentes conversacionais

Agentes conversacionais podem ser usados na área de negócio das seguintes formas:

Service Desk: Prestar auxílio aos funcionários respondendo perguntas com respostas simples, mas que ocorrem com frequência. O chatbot descrito em (KOTTORP; FILIP, 2017) consegue responder dúvidas de seus funcionários em relação a cinco tópicos: atendimento de pedido, Skype, Outlook, e-mail no dispositivo móvel pessoal e VPN utilizando informações já coletadas anteriormente de casos do Service Desk da empresa. A partir de cada fala do usuário, as entidades e a intenção detectadas são usadas para responder as dúvidas do usuário.

Serviço de atendimento ao cliente: O BlueBot (FAGGELA, 2018) da empresa aérea KLM Royal Dutch Airlines ajuda clientes a comprar passagem, mostra reservas, lembra do check-in, dá informações atualizados sobre o voo e ainda responde a perguntas dos passageiros. Ele é usado por meio do Facebook Messenger.

Informar clientes: O Chatfuel Chatbot (KŘÍŽ, 2017) foi criado pela lavanderia espanhola Mr. Jeff com o objetivo de obter novos clientes provendo informações sobre os serviços e produtos disponíveis, oferta de assinatura de serviço de lavagens de roupas por um valor fixo mensal, respostas sobre perguntas que são feitas frequentemente e indicar disponibilidade de serviço na região do cliente. Esse chatbot se comunica por meio do Facebook Messenger.

# O HOST-BOT

O Host-Bot foi projetado em módulos onde cada um realizará sua função e passará a informação produzida para o módulo seguinte a fim de realizar a tarefa desejada.

A primeira ação a ser executada é o treinamento. Ele serve para gerar o modelo usado pelo interpretador a fim de identificar a intenção e as entidades presentes na mensagem enviada pelo usuário. Para treinar é usado um conjunto de exemplos composto por frases, onde cada frase é associada a uma intenção e a uma lista de entidades. O treinamento só precisa ser realizado novamente se o conjunto de dados usados para treinar for alterado e o modelo precisar considerar essas alterações.

A interação entre o usuário e o chatbot começa pelo módulo de comunicação que funciona como interface de diálogo entre o usuário e o chatbot utilizando mensagens de texto. Esse módulo possui duas funcionalidades: passar as informações vindas do usuário para o módulo de análise de mensagens do bot e passar a resposta gerada pelo bot para o usuário.

Após receber a mensagem, é executado o módulo de análise. Ele utiliza o modelo gerado pelo módulo de treinamento e a partir dele é feito o processamento da mensagem enviada pelo usuário. Cada frase passada para esse módulo é interpretada de forma a serem identificadas a intenção que o usuário tem com essa mensagem e as entidades encontradas nela.

A partir das entidades e da intenção da mensagem são feitas consultas ao banco de dados para obter as informações desejadas pelo usuário e assim gerar as respostas.

Os dados precisam ser extraídos de alguma fonte que contenha as informações que se deseja que chatbot use para responder as perguntas do usuário e então devem ser armazenados no banco de dados na forma de registros em tabelas.

Após o processamento da mensagem enviada pelo usuário e as consultas devidas para geração da resposta, é montada a mensagem que é então enviada para o usuário por meio do módulo de comunicação. Esse processo pergunta-processamento-resposta é repetido enquanto houver mensagens vindas do usuário.

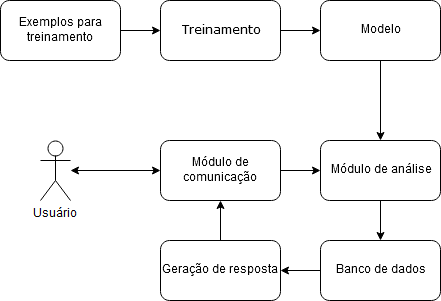


Figura - Módulos do chatbot

# Implementação

A implementação feita é um chatbot utilizando a linguagem de programação Python, a biblioteca Rasa NLU (rasa\_nlu, 2018), e é dividia em 4 módulos principais:

1) Interface com o usuário, feita por meio da plataforma do Telegram

2) Banco de dados SQLite, para armazenar os dados referentes aos eventos.

3) Processamento das mensagens do usuário.

4) Geração das respostas.

O usuário envia a mensagem por meio do Telegram ao chatbot. Ele recebe a mensagem, ela é interpreta extraindo as entidades e intenção identificadas na mensagem. A partir dessas informações, é tomada a decisão da ação a ser executada e a qual consulta ao banco de dados deve ser realizada.

## Ferramentas utilizadas

**5.1.1 Telegram**

Telegram é um serviço de mensagem instantânea desenvolvido pela empresa Telegram Messenger LLP. Existem aplicações para usar esse serviço para Android, iOS, Windows Phone, Windows, macOS e Linux. Por meio do Telegram, usuários podem enviar mensagens de texto, fotos, vídeos, mensagens em áudio e arquivos. Telegram provê APIs para desenvolvedores independentes criarem suas aplicações usando esse serviço (“Telegram F.A.Q.”).

Telegram também oferece uma API para criação de bots (“Bots”). Os bots no Telegram são aplicaçãoes de terceiros que rodam na plataforma do Telegram. Um usuário pode interagir com o bot por meio de mensagens (da mesma forma que ele interage com outros usuários) e por meio de comandos. Por meio de bots é possível obter noticias, jogar e criar ferramentas customizadas para o usuário.

**5.1.2 Python**

Python é uma linguagem de programação orientada a objeto e de código aberto que possui uma sintaxe clara e consistente além de possuir interface para diversas bibliotecas (“Python FAQ”, [s.d.]). Possui bibliotecas para processamento de linguagem natural, aprendizado de máquina e comunicação por meio do Telegram.

## 5.1.3 Telepot

Telepot é um framework em Python para Telegram Bot API (LEE, 2018). Ele serve para adicionar uma camada de abstração sobre a API do Telegram, escondendo alguns detalhes da API ajuda a construir bots para o Telegram. Esse framework possui uso fácil e conta com documentação e vários exemplos para se aprender a usá-lo.

## 5.1.4 Rasa NLU

Biblioteca para classificação de intenção e extração de entidades (BOCKLISCH et al., 2017). Usa uma lista de dados de testes composto por frases com suas intenções e entidades associadas e a partir dessas informações descobrem-se padrões nessas frases e esses padrões são generalizados para serem usados em outras frases que não foram usadas no treinamento. Essa biblioteca foi escrita em Python, porém pode ser usada por programas em qualquer linguagem. Oferece suporte a várias línguas, entre elas o português.

## 5.1.5 Chatito

Para criação de exemplos para treinamento e teste foi utilizada a ferramenta Chatito (PIMENTEL, 2018). Chatito é uma ferramenta que gera datasets específicos para serem usados em chatbots. Ela utiliza uma Linguagem de Domínio Especifico (DSL, na sigla em inglês) que especifica um padrão a ser seguido para gerar os dados no formato desejado. Ela oferece suporte para gerar exemplos para as no formato usado pela Rasa NLU.

Chatito pode tanto ser usado off-line, por meio de uma ferramenta de linha de comando, quanto online por meio do site (“Chatito - Generate training datasets for slot filling chatbots in a breeze using a simple DSL”, [s.d.]).

## 5.1.6 SQLite

Biblioteca que implementa um motor de banco de dados autocontido que não usa servidor (“About SQLite”). O código está em dominio público. Diferentemente de outros motores de banco de dados, SQLite escreve direto em arquivos do disco rígido. Todas as tabelas, views, triggers, índices ficam armazenados em um único arquivo.

## Comunicação com o usuário

A comunicação com o usuário é feita por meio do Telegram, que é um serviço de comunicação instantânea.

Para isso foi criado um bot nesse serviço. O usuário que deseja conversar com ele precisa adicioná-lo na lista de contato e fazer as perguntas desejadas. A cada pergunta, o bot passa pelos módulos descritos anteriormente e gera a resposta de acordo com as informações que estão no banco de dados.

## Processamento de linguagem natural

Uma fase importante do processo é identificar as entidades da mensagem do usuário e a intenção que se tem com essa mensagem, para que o bot executa as ações necessárias.

A implementação usa a biblioteca RASA NLU para compreensão de linguagem natural (BOCKLISCH et al., 2017).

Para isso é feito se utilizado um interpretador. Esse interpretador utilizado um modelo para identificar a intenção e as entidades.

São usados exemplos de mensagens para treinar o modelo. Os arquivos com os dados para treinamento são do formato json. Na Figura 2, um exemplo dos dados usados para treinamento.



Figura : Exemplo dos dados usados para treinamento

Os dados de exemplo são separados em arquivos, sendo um arquivo por intenção. Tal divisão não é necessária, mas ajuda na hora de fazer ajustes e correções.

entity\_synonyms servem para designar os sinônimos que existem nos dados usados, exemplo



Figura : Exemplo de sinônimo

Indica que “rio de janeiro” e “RJ” são o sinônimos.

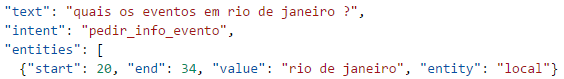


Figura : Texto com sua intenção e entidade

O atributo “text” identifica o texto a ser usado no treino.

O atributo “intent” é usado para identificar a intenção do texto.

O atributo “entities” especifica uma lista de entidades presentes no texto. No caso mostrado na imagem, o tipo da entidade é “local”, o valor dela é “rio de janeiro”, e os atributos “start” e “end” especificam onde começa e onde termina o valor “rio de janeiro” no texto.

Os dados são passados pela pipeline de treinamento, gerando uma pasta com o resultado do treinamento.

São usados alguns arquivos de configuração

config.json : utilizado para armazenar o caminho onde serão armazenados os modelos treinados.

config-spacy.json: utilizado para armazenar as informações de pipeline para processamento dos dados.

## Treinamento do modelo

O processo de treinamento do modelo ocorre da seguinte forma:

Os dados de exemplo são carregados na aplicação. De acordo as informações especificadas no arquivo config-spacy.json, é feito o treinamento. Então são gerados os arquivos de modelo para serem usados no interpretador de mensagens do chatbot.

## Criação dos dados

Para a criação dos dados de teste e treinamento foi usada a ferramenta Chatito. Um exemplo de template da DSL usada por essa ferramenta é mostrado na Figura 5:

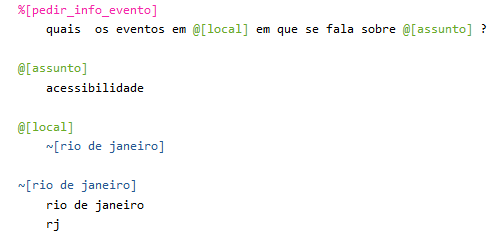


Figura : Template DSL Chatito

O sinal de porcentagem indicada a intenção da frase logo abaixo dela. Nesse caso, pedir\_info\_evento é a intenção da frase “quais os eventos em @[local] em que se fala sobre @[assunto] ?”.

O símbolo de arroba representa as entidades. Nesse caso, @[local] e @[assunto] são as entidades. Sendo que os valores abaixo da definição da entidade são os possíveis valores que ela pode assumir na frase. Nesse caso, a entidade “local” pode assumir o valor de “rio de janeiro” e a entidade “assunto” pode assumir o valor de “acessibilidade”.

~[rio de janeiro] indica que essa entidade possui sinônimos. Nesse caso, “rj” e “rio de janeiro”.

Com o arquivo preenchido de forma a gerar os exemplos, o Chatito gera as combinações possíveis substituindo as entidades da frase do topo pelos valores possíveis que as entidades podem assumir, assim gerando diversos exemplos para serem usados em treinamento e teste.

## Domínio

O chatbot é composto de entidades, ações, slots e intenções.

Slots, que são os espaços disponíveis para serem preenchidos durante a conversa de acordo com a fala do usuário.

Entidades, que são os elementos a serem identificados na fala do usuário.

Intenção, que é o objetivo do usuário com a mensagem que ele enviou ao bot.

Ações, o que o bot pode fazer.

Na presente implementação foram feitas as seguintes escolhas:

Quatro intenções

pedir\_info\_evento: pede informações sobre eventos.

pedir\_info\_palestra: pede informações sobre palestras.

pedir\_info\_palestrante: pede informações sobre palestrantes.

saudar: usuário enviou mensagem cumprimentando o chatbot. É respondido com oferta de ajuda.

Cinco entidades/slots

Nome: do evento/palestra/palestrante

Descrição: do evento/palestra/palestrante

Assunto: do evento/palestra

Data: do evento/palestra

Hora: do evento/palestra

Local: do evento/palestra

Evento: evento sobre o qual está sendo falado na conversa atual

Palestra: palestra sobre a qual está sendo falada na conversa atual

Palestrante: palestrante sobre o qual está sendo falado na conversa atual

Ações

buscar\_info\_palestra: busca informações sobre as palestras de acordo com os slots preenchidos.

buscar\_info\_palestrante: busca informações sobre os palestrantes de acordo com os slots preenchidos.

buscar\_info\_evento: busca informações sobre os eventos de acordo com os slots preenchidos.

## Pipeline

As mensagens que o usuário envia passam pelos componentes da pipeline para serem processadas, sendo que cada componente tem sua responsabilidade. A mensagem passa pelo componente, é gerada uma saída que é então passada para o componente seguinte da pipeline até chegar ao final.

A pipeline usada para o processamento das mensagens é composta dos seguintes componentes:

nlp\_spacy: inicializa as estrutura do spacy, por isso deve ser o primeiro componente da pipeline. Não produz saída para os outros componentes.

tokenizer\_spacy: cria tokens usando o tokenizer do spacy.

intent\_featurizer\_spacy: Cria as features para classificação de intenção.

intent\_entity\_featurizer\_regex: criação de expressões regulares para auxiliar a classificação de intenção e entidades.

ner\_crf: utiliza spacy para extrair as entidades.

ner\_synonyms: mapeia os valores das entidades com seus respectivos sinônimos.

intent\_classifier\_sklearn: gera por saída a intenção da mensagem e um ranking com as possíveis intenções da mensagem indicando um valor de confiança de cada possibilidade. Utiliza para esse fim uma SVM linear.

## Execução

O usuário envia mensagem para o chatbot. O chatbot interpreta a mensagem identificando a intenção e as entidades. Exemplo do resultado da identificação na Figura 3.



Figura : Exemplo do resultado da identificação

Então a partir da intenção é selecionada a ação a ser tomada e a partir do valor das entidades identificadas.

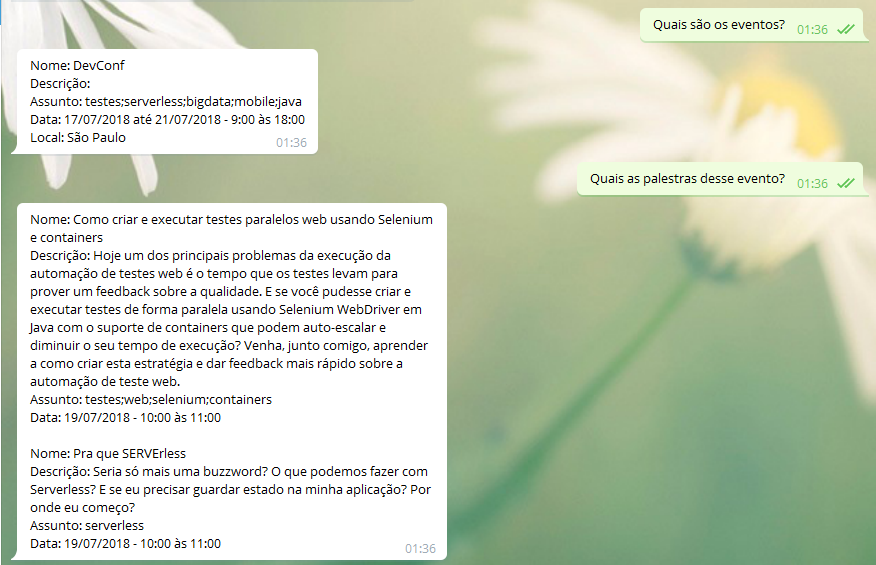
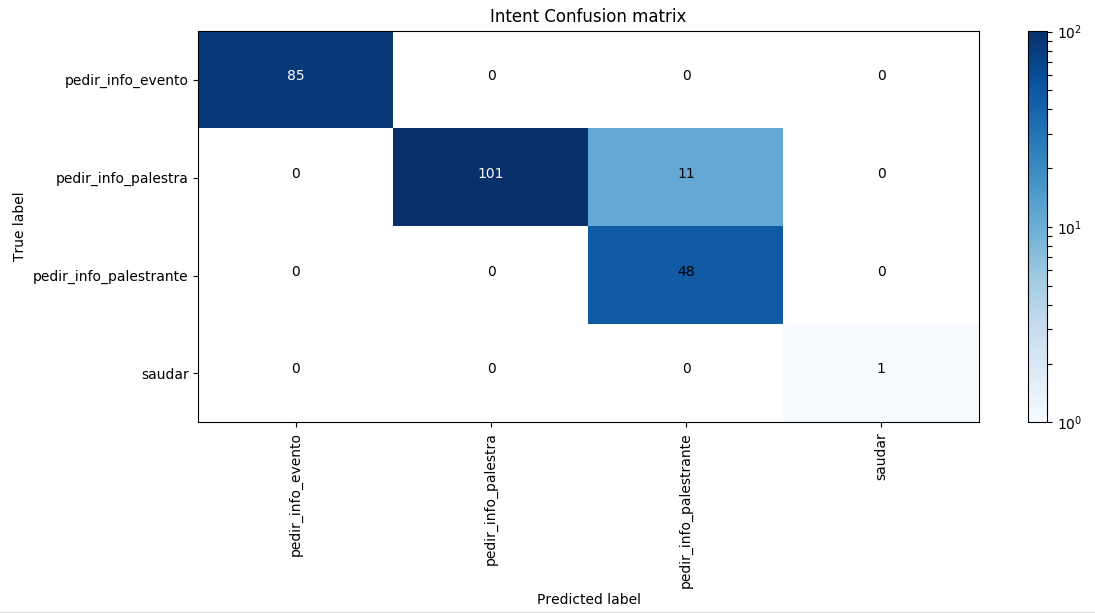


Figura : Exemplo de conversa com o chatbot

# Avaliação da ferramenta

A partir dos dados de exemplo criados pela ferramenta Chatito, são criados dois conjuntos de dados: um para treinamento e um para testes. O primeiro conjunto é usado para gerar o modelo que será usado para interpretar as mensagens vindas do usuário enquanto o segundo será usado para avaliar quão bom está o modelo. Os dados de teste são passados pelo interpretador e compara-se a classificação gerada com a classificação associada a cada exemplo do conjunto de dados de teste. Após isso é desenhada a matriz de confusão com os resultados gerados pela avaliação. Uma matriz de confusão é uma de tabela que permite visualizar o desempenho de um algoritmo de classificação mostrando quais elementos foram classificados corretamente ou erroneamente.



Além da matriz de confusão, três medidas foram calculadas na avaliação da classificação das intenções e entidades (POWERS, 2011).

Revocação é a proporção de casos positivos reais que foram corretamente classificados como positivo. Ela é calculada da seguinte forma:

Precisão é a proporção de casos classificados como positivo que são positivos de verdade. Ela é calculada da seguinte forma:

F1 é a média harmônica da precisão e da revocação. É utilizada como uma medida de precisão da avaliação realizada. (SASAKI, 2007)

Suporte: número de exemplos de determinada classe.

Valores das métricas da avaliação da classificação de intenção

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precisão | Revocação | F1 | Suporte |
| pedir\_info\_evento | 1.00 | 1.00 | 1.00 | 85 |
| pedir\_info\_palestra | 1.00 | 0.90 | 0.95 | 112 |
| pedir\_info\_palestrante | 0.81 | 1.00 | 0.90 | 48 |
| saudar | 1.00 | 1.00 | 1.00 | 1 |
| média / total | 0.96 | 0.96 | 0.96 | 246 |

Valores das métricas de avaliação da classificação de entidades

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precisão | Revocação | F1 | Suporte |
| Assunto | 1.00 | 1.00 | 1.00 | 135 |
| Data | 0.98 | 1.00 | 0.99 | 46 |
| Evento | 1.00 | 1.00 | 1.00 | 32 |
| Local | 1.00 | 0.99 | 1.00 | 155 |
| no\_entity | 1.00 | 1.00 | 1.00 | 1530 |
| Palestra | 1.00 | 1.00 | 1.00 | 12 |
| Saudação | 1.00 | 1.00 | 1.00 | 1 |
| avg / total | 1.00 | 1.00 | 1.00 | 1911 |

# Conclusão

O propósito desse projeto era implementar um chatbot para comunicar ao usuário informações sobre eventos. Começando com poucas intenções, foi aumentando o número a medida que o chatbot era usado e com isso novas formas de interação eram descobertas e assim foi melhorando a interação.

# Referências bibliográficas

Neves, A. M. M.(2005). “**IAIML - Um Mecanismo para o Tratamento de Intenção em Chatterbots**”. Tese de Doutorado em Ciência da Computação. Universidade Federal de Pernambuco.

Colby, K.M., Hilf, F.D., Weber, S., e Kraemer, H.C. (1972). “**Turing-like Indistinguishability Tests for the Validation of a Computer Simulation of Paranoid Processes**”. Artificial Intelligence, v.3, pp. 199-222.

WALLACE, R. S. **The Anatomy of A.L.I.C.E.** In: EPSTEIN, R.; ROBERTS, G.; BEBER, G. (Eds.). . Parsing the Turing Test. Dordrecht: Springer Netherlands, 2009. p. 181–210.

Russell, S. e Norvig, P. (2010). **Artificial Intelligence: A Modern Approach**. Prentice Hall. 3rd Edition.

Christiane Fellbaum, editor. **Wordnet: An Electronic Lexical Database**. The MIT Press, 1999.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. **The Berkeley FrameNet project.** In Christian Boitet and Pete Whitelock, editors, Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics, pages 86–90, San Francisco, California, 1998. Morgan Kaufmann Publishers.

Sanda Harabagiu, Marius Pasca, and Steven Maiorano. **Experiments with open-domain textual question answering**. In Proceedings of COLING-2000, Saarbr¨uken Germany, August 2000.

John Searle. **Expression and Meaning: Studies in the Theory of Speech Acts.** Cambridge University Press, 1979.

**British National Corpus**. Disponível em: <http://www.natcorp.ox.ac.uk/>. Acesso em: 4 jun. 2018.

**Linguistic Data Consortium - Linguistic Data Consortium.** Disponível em: <https://catalog.ldc.upenn.edu/>. Acesso em: 4 jun. 2018.

British national corpus, 2003. http://www.natcorp.ox.ac.uk/.

**Text REtrieval Conference (TREC) Home Page**. Disponível em: <https://trec.nist.gov/>. Acesso em: 4 jun. 2018.

Eric Brill. **Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging.** Computational Linguistics, 21(4):543–565, 1995.

Adwait Ratnaparkhi. **A maximum entropy model for part-of-speech tagging.** In Eric Brill and Kenneth Church, editors, Proceedings of the Conference on Empirical Methods in Natural Language Processing, pages 133–142. Association for Computational Linguistics, Somerset, New Jersey, 1996.

Claire Cardie, Scott Mardis, and David Pierce. **Combining error-driven pruning and classication for partial parsing.** In Proceedings of the 16th International Conference on Machine Learning, pages 87–96. Morgan Kaufmann, San Francisco, CA, 1999.

Steven Abney. **Partial parsing via nite-state cascades.** Natural Language Engineering, 2(4):337–344, 1995.

Michael Collins. **Three generative, lexicalised models for statistical parsing.** In Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics, Madrid, 1997.

BOCKLISCH, T. et al**. Rasa: Open Source Language Understanding and Dialogue Management.** arXiv:1712.05181 [cs], 14 dez. 2017.

PIMENTEL, R. **Chatito: Generate datasets for slot filling NLU chatbots in a breeze using a simple DSL!** [s.l: s.n.].

**Chatito - Generate training datasets for slot filling chatbots in a breeze using a simple DSL.** Disponível em: <https://rodrigopivi.github.io/Chatito/>. Acesso em: 28 maio. 2018.

**rasa\_nlu: turn natural language into structured data**. Python, Rasa. Recuperado de https://github.com/RasaHQ/rasa\_nlu, 2018.

PEDREGOSA, F. et al. **Scikit-learn: Machine learning in Python**. Journal of machine learning research, v. 12, n. Oct, p. 2825–2830, 2011.

**spaCy: Industrial-strength Natural Language Processing (NLP) with Python and Cython**. Python, Explosion AI. Recuperado de https://github.com/explosion/spaCy, 2018.

CORTES, C.; VAPNIK, V. **Support-vector networks.** Machine learning, v. 20, n. 3, p. 273–297, 1995.

WEIZENBAUM, J. **ELIZA—a computer program for the study of natural language communication between man and machine.** Communications of the ACM, v. 9, n. 1, p. 36–45, 1966.

Telegram F.A.Q. .**Telegram**. Recuperado maio 29, 2018, de https://telegram.org/faq#q-who-are-the-people-behind-telegram.

**Bots: An introduction for developers**. . Recuperado maio 29, 2018, de https://core.telegram.org/bots.

**About SQLite**. . Recuperado maio 29, 2018, de https://www.sqlite.org/about.html.

LEE, N. **telepot: Python framework for Telegram Bot API**. Python, . Recuperado de https://github.com/nickoala/telepot, 2018.

MAULDIN, M. L. **CHATTERBOTS, TINYMUDS, and the Turing Test Entering the Loebner Prize Competition.** . Recuperado junho 15, 2017, de http://www.lazytd.com/lti/pub/aaai94.html, 1994.

CHOWDHURY, G. G. **Natural language processing**. Annual Review of Information Science and Technology, v. 37, n. 1, 31 jan. 2005.

SERBAN, I. V. et al. **A Survey of Available Corpora for Building Data-Driven Dialogue Systems**. p. 56, 2017.

BOBROW, D. G. et al. **GUS, A Frame-Driven Dialog System**. p. 19, 1977.

WALKER, M.; KAMM, C.; LITMAN, D. **Towards developing general models of usability with PARADISE**. Natural Language Engineering, v. 6, n. 3&4, p. 363–377, set. 2000.

POLIFRONI, J. et al. **Experiments in evaluating interactive spoken language systems.** Association for Computational Linguistics, 1992.

DANIELI, M.; GERBINO, E. **Metrics for Evaluating Dialogue Strategies in a Spoken Language System**. p. 6, 1995.

HIRSCHMAN, Lynette; PAO, Christine. **The cost of errors in a spoken language system.** Third European Conference on Speech Communication and Technology. 1993.

ZUE, V. et al. **Preliminary evaluation of the VOYAGER spoken language system.** Association for Computational Linguistics, 1989.

ARTSTEIN, R. et al. **Semi-formal Evaluation of Conversational Characters.** In: GRUMBERG, O. et al. (Eds.). . Languages: From Formal to Natural. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. v. 5533p. 22–35.

LOWE, R. et al. **Towards an Automatic Turing Test: Learning to Evaluate Dialogue Responses.** arXiv:1708.07149 [cs], 23 ago. 2017.

BOWMAN, S. R. et al. **Generating Sentences from a Continuous Space.** arXiv:1511.06349 [cs], 19 nov. 2015.

KANNAN, A.; VINYALS, O. **Adversarial Evaluation of Dialogue Models.** arXiv:1701.08198 [cs], 2016.

LI, J. et al. **Adversarial Learning for Neural Dialogue Generation**. arXiv:1701.06547 [cs], 23 jan. 2017.

WALKER, M.; WHITTAKER, S**. Mixed Initiative in Dialogue: An Investigation into Discourse Segmentation**. arXiv:cmp-lg/9504007, 1990.

SUENDERMANN, D. et al**. From rule-based to statistical grammars: Continuous improvement of large-scale spoken dialog systems**. IEEE, abr. 2009.

LESTER, J.; BRANTING, K.; MOTT, B. **Conversational agents.** The Practical Handbook of Internet Computing, p. 220–240, 2004.

RUSSELL, S. J.; NORVIG, P.; DAVIS, E. **Artificial intelligence: a modern approach.** 3rd ed ed. Upper Saddle River: Prentice Hall, 2010. p. 30.

JIA, J. **The Study of the Application of a Keywords-based Chatbot System on the Teaching of Foreign Languages.** p. 11, 2003.

TURING, A. M. I.**—COMPUTING MACHINERY AND INTELLIGENCE.** Mind, v. LIX, n. 236, p. 433–460, 1950.

**Telegram Bot Platform.** Disponível em: <https://telegram.org/blog/bot-revolution>. Acesso em: 18 jun. 2018.

**Instant Messaging Statistics Report, 2015-2019**. , 16 mar. 2015. Disponível em: <https://www.radicati.com/?p=13001>. Acesso em: 22 jun. 2018

**What are Skype bots and how do I add them as contacts? | Skype Support.** Disponível em: <https://support.skype.com/en/faq/FA34646/what-are-skype-bots-and-how-do-i-add-them-as-contacts>. Acesso em: 18 jun. 2018.

**How To Build Bots for Messenger.** Disponível em: <https://developers.facebook.com/blog/post/2016/04/12/bots-for-messenger/>. Acesso em: 18 jun. 2018.

IBGE**. Acesso à internet e à televisão e posse de telefone móvel celular para uso pessoal: 2016, 2016.** Disponível em: <https://biblioteca.ibge.gov.br/visualizacao/livros/liv101543.pdf>

TAYLOR, G. **Retail’s Big Opportunity: 87% Of U.S. Consumers Grasp The Power Of Conversational Commerce - Retail TouchPoints.** Disponível em: <https://www.retailtouchpoints.com/topics/omnichannel-cross-channel-strategies/retail-s-big-opportunity-87-of-u-s-consumers-grasp-the-power-of-conversational-commerce>. Acesso em: 22 jun. 2018.

SLACK. **About Us.** Disponível em: <https://slack.com/about>. Acesso em: 22 jun. 2018.

WhatsApp. Disponível em: <https://www.whatsapp.com/business/>. Acesso em: 22 jun. 2018.

KOTTORP, M.; FILIP, J. **Chatbot as a potential tool for businesses**, 2017.

FAGGELA, D. **How companies are using chatbots for marketing: Use cases and inspiration.** Disponível em: <https://martechtoday.com/how-companies-are-chatbots-marketing-209475>. Acesso em: 22 jun. 2018.

SHIH, Y.-C.; FAN, S.-T. **Adoption of instant messaging by travel agency workers in Taiwan: Integrating technology readiness with the theory of planned behavior.** International Journal of Business and Information, v. 8, n. 1, 2015.

BOUHNIK, D.; DESHEN, M. **WhatsApp goes to school: Mobile instant messaging between teachers and students.** Journal of Information Technology Education: Research, v. 13, n. 1, p. 217–231, 2014.

SHAWAR, B. A.; ATWELL, E. **Chatbots: are they really useful?** LDV Forum. Anais...2007Disponível em: <http://media.dwds.de/jlcl/2007\_Heft1/Bayan\_Abu-Shawar\_and\_Eric\_Atwell.pdf>. Acesso em: 15 nov. 2016

MARTIN, J. H.; JURAFSKY, D. **Speech and Language Processing**, 2017.

POWERS, D. M. **Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation.** 2011.

SASAKI, Y. **The truth of the F-measure.** Teach Tutor mater, v. 1, n. 5, 2007.

**General Python FAQ — Python 3.6.6 documentation.** Disponível em: <https://docs.python.org/3/faq/general.html>. Acesso em: 27 jun. 2018.