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 Filho

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 Filho – Orientador

UFF - Universidade Federal Fluminense

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

Nome do Avaliador 1

UFF - Universidade Federal Fluminense

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

Nome do Avaliador 1

UFRJ – Universidade Federal do Rio de Janeiro

Dedico este trabalho a(o) minha(meu) esposa(o) e aos meus estimados filhos.

AGRADECIMENTOS

A Deus, que sempre iluminou a minha caminhada.

A meu Orientador Fulano de Tal 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.

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

**LISTA DE ILUSTRAÇÕES**

Figura 1: exemplo..............................................................................................30

**LISTA DE TABELAS**

[Tabela 1: Lista dos tutores Orientadores do TCC. 19](#_Toc267917793)

**LISTA DE GRÁFICOS**

[Gráfico 1: Exemplo de um gráfico. 26](#_Toc268005981)

LISTA DE ABREVIATURAS E SIGLAS

LDB – Lei de Diretrizes e Bases

MALO – Museu ao Ar Livre de Orleans

SMA – Secretaria Municipal de Administração

SME – Secretaria Municipal de Educação

SUMÁRIO

**Introdução**

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. Além de novas interfaces que contam com ferramentas para interagir com esses agentes, como Facebook Messager e Telegram.

Avanços nas pesquisas de aprendizado de máquina e Processamento de linguagem natural permitiram melhor proveito desses agentes.

Nesse trabalho será falado sobre um agente conversacional para obter informações sobre eventos, e as palestras e palestrantes deles.

No primeiro capítulo fala o que são agentes conversacionais, aplicações e desafios em sua implementação.

O capitulo 2 traça um histórico dos agentes conversacionais

Capitulo 3 descreve os tipos de agentes conversacionais.

Capitulo 4 apresenta conceitos usado nesse trabalho.

Capitulo 5 apresenta detalhes da implementação.

Capitulo 6 encerra com a conclusão.

**1 Conceitos**

**1.1 Processamento de Linguagem Natural**

Processamento de Linguagem Natural (PLN) é uma área de conhecimento que tem por objetivo reunir conhecimento para entender como pessoas usam a linguagem para desenvolver sistemas que entendam e manipulem a linguagem de forma a realizar as tarefas desejadas (CHOWDHURY, 2005). É uma das áreas de pesquisa mais antiga dentro de Inteligência Artificial. Apesar de antiga, os avanços na área foram lentos devido a necessidade de montar os sistema de PLN de forma manual o que exigia muito tempo e esforço. No entanto, em 1990, três fatores levaram a aceleração do progresso na área de PLN. Primeiro, o desenvolvimento de um grande corpus de textos etiquetados, como o Brown Corpus, o Penn Treebank (“Linguistic Data Consortium - Linguistic Data Consortium”), e o British National Corpus (“British National Corpus”, [s.d.]). Segundo, o desenvolvimento de técnicas estatísticas e de aprendizado de máquina para extrair gramáticas, ontologias, e outras informações de corpus etiquetados. Terceiro, competições como MUC e TREC (“Text REtrieval Conference (TREC) Home Page”, [s.d.]) que funcionavam como incentivos a inovações da área.

**1.2 Tokenização**

Processo onde a entrada passada pelo usuário é dividida em uma séria de entidades léxicas distintas. A tokenização, no entanto, pode ser bem complexa. Possíveis causas de complexidades são acrônimos, expressões técnicas e declarações com palavras escritas erradamente.

**1.3 SVM**

SVM (Support Vector Machine) é um conjunto de algoritmos de aprendizado de máquina supervisionado usado para classificação e regressão. A partir de um conjunto de exemplos para treinamento, sendo cada exemplo já classificado como sendo membro de uma de duas categorias, um algoritmo de treinamento de SVM constrói um modelo que atribui novos exemplos a uma categoria ou a outra. (CORTES; VAPNIK, 1995). Um exemplo de uso de SVM é classificar a intenção de determinado texto.

**1.4 Featurizer**

Featurizer identifica features. Uma feature é uma propriedade mensurável de uma propriedade sendo observada (BISHOP, 2006). Pode ser usado, por exemplo, na classificação de textos.

**1.5 Named Entity Recognition**

A função desse processo é classificar elementos de um texto em categorias como Pessoa ou Local.

Para isso é utilizado um modelo que é treinado com base em exemplos para identificar novas entidades e a qual categoria elas pertencem. Mas em alguns casos também podem ser usadas expressões regulares para identificar certas entidades, por exemplo, nacionalidades.

**2 - Agentes conversacionais**

Agentes conversacionais são programas que se comunicam por meio de linguagem natural e costumam se enquadrar em duas categorias:

Agentes conversacionais orientados a tarefas: agentes projetados para uma tarefa particular e que realizam dialogo com o usuário para obter as informações necessárias para ajudar a completar uma tarefa. Ex.: Siri, Cortana.

Chatbots: agentes projetados para conversas mais extensas, imitando uma conversa humana em vez de uma tarefa especifica. Costuma ser usado como sinônimo para agente conversacional.

Chatbots são sistemas que mantem uma conversa com o objetivo de imitar uma conversa entre humanos. Costuma serem usados para entretenimento.

A arquitetura de chatbots se divide em dois grupos: sistemas baseados em regras e sistemas baseados em corpus. Sistemas baseados em regras incluem chatbots como ELIZA e PARRY. Sistemas baseados em corpus usam grande quantidade de dados de conversas para aprender a associar uma fala do usuário com uma resposta para ele.

Um diálogo é composto de múltiplos turnos onde em cada turno os participantes fazem sua contribuição e passam a oportunidade de interação para o outro. Um turno pode ser uma sentença, uma palavra, ou múltiplas sentenças.

**2.1 Sistema e agentes conversacionais**

Sistemas conversacionais são sistemas projetados para interpretar a interação do usuário e reagir a essa interação como uma resposta buscando manter a coerência do dialogo dando a ilusão que se está conversando com outro ser humano. Devido a isso são considerados agentes inteligentes. Por terem a característica de conversar, eles são chamados de agentes conversacionais.

Agentes conversacionais (chatbots) usam técnicas que usam a linguagem natural para auxiliar usuários na busca de informações ou para realizar tarefas para as mais diversas aplicações. 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.

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).

Agentes conversacionais podem servir como uma solução com bom custo beneficio na área empresarial. Soluções automatizadas podem reduzir custos com funcionários, podem assim prover serviços de atendimentos a clientes de forma mais barata. Nessa área, agentes podem auxiliar tanto ao lidar com os clientes da empresa auxiliar a tirar duvidas quanto ajudando os funcionários em treinamentos internos para aprender as funções do trabalho.

**2.2 Aplicações**

Na área de negócios a comunicação é de extrema importância, não só com o cliente, mas também com fornecedores e entre funcionários. Agentes conversacionais podem ser usados nessa área nas seguintes formas:

Serviço de atendimento ao cliente: responder dúvidas de clientes sobre produtos e serviços.

Help Desk: Responder perguntas de funcionários sobre a empresa.

Navegação do site: guiar clientes até as partes importantes e desejadas do website da empresa.

Venda guiada: auxiliar cliente no processo de venda de produtos complexos.

Suporte técnico: responder sobre problemas técnicos sobre dispositivos ou aparelhos eletrônicos.

**2.3 Desafios técnicos**

Chatbots devem satisfazer dois requerimentos. Primeiro, ter capacidade de processamento da linguagem suficiente para ter uma conversa produtiva com os usuários. Devem entender as perguntas e responde-las com precisão. Segundo, devem operar de forma efetiva e confiável.

***2.3.1 Desafios relacionados a linguagem natural***

Processamento de linguagem natural eficiente é necessário para um agente conversacional. Para funcionar do jeito correto, ele deve interpretar a mensagem enviada pelo usuário, o que deve ser feito em relação a essa mensagem e executar as ações devidas, como responder a mensagem com outra mensagem, mostrar uma página na web, realizar alguma operação de compra. (LESTER; BRANTING; MOTT, 2004)

Por exemplo, se o usuário dissesse:

1. Eu gostaria de compra-lo agora.

O agente precisaria determinar o sentido da frase. Após isso, é preciso determinar qual ação deve ser tomada a partir dessa frase para alcançar o objetivo desejado pelo usuário. A ação vai depender o objetivo do agente no momento em que recebeu a mensagem, a conversa já realizada com esse usuário e as informações disponíveis para o agente. Então, o agente deve realizar a ação apropriada, como responder a mensagem, apresentar alguma informação ou realizar alguma outra ação.

**2.4 Requisitos de Negócios**

Agentes conversacionais só poderão ser usados em um ambiente de negócio se atender a suas necessidades. Para isso eles devem ter um padrão que permita atender a diversos usuários em larga escala. Eles devem ser escaláveis, ter bom desempenho, confiáveis, seguros e se integrarem aos sistemas e processos da empresa que os está utilizando, oferecer ferramentas para criar e manter informações da base de conhecimento e permitir uso de ferramentas de analise e geração de relatório.

Escalabilidade. O agente conversacional terá que lidar com um grande número de conversas e contatos, por isso é importante levar em consideração essa característica ao planejar a implementação do agente. Eles também devem ser capazes de lidar com um crescimento em pouco tempo no volume de conversas por meio de alocação dinâmica de recursos.

Desempenho. Chatbots devem atender requisitos de desempenho. Estes podem ser medidos de duas formas. Primeiro, o agente deve ser capaz de lidar uma taxa especifica de mensagens por minuto. Segundo, agentes devem prover garantias de numero de conversas simultâneas com o qual conseguem lidar. Eles também devem responder dentro de poucos milissegundos para que o tempo total (que inclui a latência da rede) fique dentro de entre um ou dois segundos.

Segurança. Agentes conversacionais devem oferecer o mesmo nível de segurança já provido pela organização. Mas como conversas podem conter informações confidenciais, a segurança deve operar um nível acima do lugar onde o agente está funcionando. Por isso, agentes devem usar canais seguros e mecanismos de autenticação e autorização.

Integração. Agentes conversacionais devem se integrar com a infraestrutura já existente. Na camada de apresentação, devem estar integrados com os outros elementos personalizados do sistema. Na camada de aplicação, devem estar integrados com as regras de negócio existentes, como por exemplo, regra para transferir a conversa para um funcionário ou regra que afetem o gerenciamento do dialogo. Na camada de armazenamento de dados, os agentes devem ter facilidade para usar os dados da empresa, por exemplo, catalogo de produtos.

Também é importante que os dados gerados pelo agente possam ser armazenados e utilizados para aprender como melhorar o funcionamento do agente e saber os interesses dos usuários que usaram o sistema.

**3 Histórico**

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

**3.1 Primeira Geração**

O primeiro chatbot de maior reconhecimento nessa geração foi ELIZA (Weizenbaum, 1966). Ele simulava um psicanalista da rogeriano.

Os chatbots dessa geração usavam técnicas simples para manter diálogo, usando batimento de padrão em busca de palavras e frases chaves. Eles utilizam pares <Pergunta-Resposta> para conversar. Ao receber uma mensagem do usuário, era buscado o padrão mais próximo para então mostrar a resposta associada a esse padrão. ELIZA sofria com a falta de um modelo interno que influenciasse o rumo da conversa e mantivesse o status da conversa.

ELIZA levou a criação de novos chatbots seguindo modelo semelhante.

Outro chatbot dessa época é PARRY [Colby et al., 1972] que simulava um paciente com paranoia. PARRY usava uma estratégia para conversação e era mais avançado que ELIZA. Ele posuia um modelo de estado mental onde variavam os níveis de medo e raiva.fazia julgamentos acerca da pessoa com que ele conversa, podendo suspeitar da pessoa ou acusar a pessoa de ter más intenções, assim como uma pessoa paranoica, além de guardar um estado interno de como ele se sentia em relação a pessoa com que ele conversava.

PARRY continha um modelo de estado mental onde variavam os níveis de medo e raiva. De acordo com o tópico abordado, PARRY ficava mais desconfiado ou nervoso. Se o nível de raiva era alto, ele escolheria uma resposta de um conjunto de respostas hostis. Se o usuário mencionasse o tópico pelo qual PARRY tivesse desilusão, aumentaria o nível de medo.

Apesar do sucesso, as técnicas usadas neles eram limitadas no que refletia em diálogos com poucos detalhes.

Para manter uma conversa próxima ao real era necessário um grande numero de pares <Pergunta-Resposta>. Além desse problema, o chatbot não armazenava informações sobre o usuário, o que permitiria personalização da conversa.

**3.2 Segunda Geração**

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 morfossintatico para identificar variações morfologicas e sintáticas das frases [Neves, 2005]. Outro avanço foi a inclusão de bases de conhecimento de domínio especifico.

O chatbot que merece destaque dessa geração é CHATTERBOT (MAULDIN, 1994), desenvolvido para ser usado em um ambiente de entretenimento multiusuário de 1989, TinyMUD.

TinyMUD permitia conversa entre vários usuários e criação de ambientes onde ocorriam as interações.

CHATTERBOT interagia como se fosse um humano com outras pessoas por meio de salas de chat. Foi implementado na linguagem C e possuía módulos para lidar com a comunicação com a plataforma, guardava o estado das varias salas e objetos da plataforma, armazenava mensagens de usuários, e o modulo que o permitia fazer a conversa propriamente dita. O modulo de conversa era implementado com uma série de níveis em cada nivel tinha uma responsabilidade e prioridade.

Ele conseguia guardar informações sobre o usuário a partir de conversa com ele. Essa funcionalidade tinha por objetivo dar maior credibilidade ao dialogo.

A técnica utilizada para lidar com a maioria das perguntas sobre um assunto especifico era rede de ativação: utilizava um grafo com peso no nós para codificar o que o usuário e o programa já disseram, sendo que cada nó começa um valor de ativação entre 1 e 0. O valor de ativação é aumentado ou diminuído de acordo com seu uso na resposta do bot.

Apesar dos avanços os chatbots ainda apresentavam problemas. Sendo que alguns bots de primeira geração conseguiam vencer os bots de segunda geração em competições, como o prêmio Loebner. O maior problema era a necessidade de refazer a Rede de Ativação a cada novo dominio de aplicação. Outros problemas eram ter que controlar o andamento da conversação (inicio, desenvolvimento e fim), além de ter quer oferecer tratamento a perguntas desconhecidas.

**3.3 Terceira geração**

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 coleção de dados de linguagem natural coletados da web. A.L.I.C.E. ganhou o prêmio Loebner em 2000 e 2001. A.L.I.C.E. 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.

**3.4 Hoje em dia**

Com avanços do processamento de linguagem natural e algoritmos de aprendizado de máquina, melhorias estão sendo feitas no no chatbots, como se pode ver na Cortana, da Microsoft, Alexa, da Amazon, Siri, da Apple e Google Now, do Google.

**4 Tipos de agentes conversacionais**

**4.1 Chatbots baseados em regras**

Nesse tipo de chatbot as sentenças das mensagens vindas do usuário são analisadas de acordo com regras que são ativadas de acordo com a estrutura da frase e palavras chaves. E as respostas são escolhidas de acordo com outras regras que foram ativadas. (WEIZENBAUM, 1966)

Dois exemplos de chatbots desse tipo foram ELIZA e PARRY.

Abaixo um exemplo de conversa de ELIZA extraído do artigo original:

Men are all alike. (Homens são todos iguais)

IN WHAT WAY (De que jeito)

They're always bugging us about something or other. (Eles estão sempre nos incomodando sobre alguma coisa)

CAN YOU THINK OF A SPECIFIC EXAMPLE (Você consegue pensar em um exemplo específico)

Well my boyfriend made me come here. (Bem, meu namorado me fez vir aqui)

YOUR BOYFRIEND MADE YOU COME HERE (Seu namorado fez você vir aqui)

He says I'm depressed much of the time. (Ele diz que eu estou deprimida a maior parte do tempo)

I AM SORRY TO HEAR YOU ARE DEPRESSED (Sinto muito ouvir que você está deprimida)

ELIZA funcionava comparando padrões entre as falas da pessoa que conversava com ela e as regras que ela tinha armazenadas.

(0 você me 0) [padrão]

->

(O que te faz pensar que eu 3 você) [resposta]

No padrão de ELIZA, 0 significa um curinga que combina com qualquer entrada, e na regra de resposta os números fazem referencia a posição das palavras que constituem o padrão.

Por exemplo, se o usuário digitasse:

Você me odeia

A reposta que viria de ELIZA seria:

O QUE TE FAZ PENSAR QUE EU TE ODEIO

Cada padrão está associado a uma palavra chave que possa ocorrer na fala usuário.

Palavras chaves estão associadas a níveis, sendo as palavras mais especificas em níveis mais altos e palavras menos especificas em níveis mais baixos.

Ex.:

Eu sei que todos riram de mim

Por possuir a palavra “I”, essa sentença iria combinar com a seguinte regra que tem por palavra chave “I”:

(Eu \*) -> (Você diz que você 2)

E geraria por resposta:

VOCÊ DIZ QUE VOCÊ QUE TODOS RIRAM DE VOCÊ

No entanto, “Eu” é uma palavra especifica e por isso geraria respostas pouco especificas. A palavra chave “todos” é mais interessante pois a pessoa que usa essa palavra estava referindo a uma pessoa em especifico. Por exemplo, a se deparar com a palavra “todos” a resposta gerada é:

EM QUEM EM PARTICULAR VOCÊ ESTÁ PENSANDO?

Se nenhuma palavra chave é encontrada, ELIZA incentiva a pessoa com que se está conversando a falar mais usando termos como “POR FAVOR CONTINUE” ou “ISSO É MUITO INTERESSANTE”.

4.2 Chatbots baseados em corpus

Chatbots baseados em corpus extraem as informações de conversas entre pessoas ou entre pessoas e maquinas. 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)

**4.2.1 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: dada uma consulta q e um corpus C, encontre o turno t em C que é mais similar a q e retorne o turno seguinte (a resposta humana para t em C):

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).

1. Retornar o turno mais similar: dado uma consulta q do usuário e um corpus C, retornar t em C que é mais similar a q:

A ideia aqui é casar a consulta q do usuário com turnos de C, 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.

**4.3 Agentes de dialogo baseados em frame**

Sistemas de dialogo modernos orientados a tarefas são baseados em ontologia de domínio, que é uma representação de conceitos que pertencem a uma parte do mundo, no caso, as relações entre as entidades que o agente pode extrair das mensagens do usuário. 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 como sistema GUS para planejamento de viagem (BOBROW et al., 1977).

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

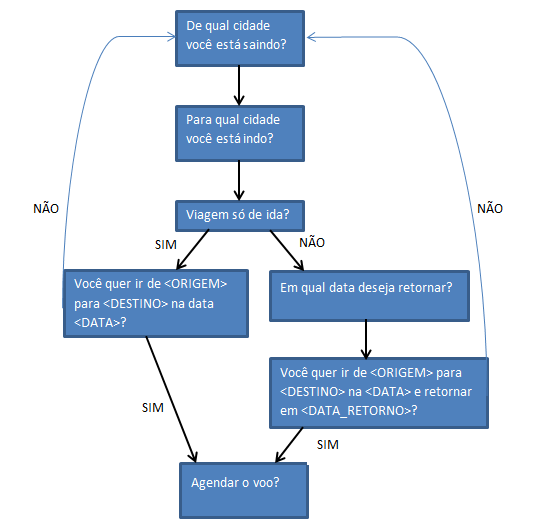
O conjunto de slots do frame GUS especifica o que o sistema precisa saber, e o conteúdo de cada slot é restringido aos valores de um tipo particular. No domínio de viagem, um slot pode ser do tipo cidade, data ou hora da partida.

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.

**4.3.1 Estrutura de controle para diálogos baseados em frames**

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 dialogo.

O exemplo abaixo tem como base o sistema GUS, detalhado em (BOBROW et al., 1977).



Considerando a maquina de estados representada na figura anterior que implementa um sistema de viagem de avião que tem por objetivo pedir 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?” |

Cada nó da figura anterior corresponde a uma pergunta dos slots e cada seta aponta ao caminho que deve ser seguido de acordo a resposta do usuário. Esse sistema pergunta uma série de perguntas para o usuário, ignorando tudo que não for uma resposta direta a pergunta feita e depois segue para a pergunta seguinte.

A arquitetura do GUS para sistema de dialogo baseado em frame tem o frame de um jeito mais flexível. O sistema faz perguntas para o usuário, preenchendo assim os slots mesmo que a resposta preencha vários slots ou não responda a pergunta feita. O sistema pula as perguntas cujo slots já foram preenchidos. Desse jeito, os slots podem ser preenchidos fora de ordem. Por isso a arquitetura do GUS é considerada de iniciativa mista, já que o usuário pode ter um pouco de iniciativa ao decidir o tópico falado.

Assim que sistema obtiver informação suficiente, ele realiza a ação necessária, por exemplo, consulta o banco de dados para obter informações sobre os voos disponíveis.

Já que os usuários podem mudar entre frames durante a conversa é necessário saber de qual frame é cada slot e qual informação vai para cada slot e poder alterar o controle do dialogo para o frame que está sendo trabalhado.

**4.3.2 Entendimento de linguagem natural para preencher slots**

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

Por exemplo, da fala abaixo:

Me mostre os voos da manhã de São Paulo para o Rio de Janeiro na terça-feita

O sistema poderia construir a representação a seguir:

|  |  |
| --- | --- |
| Dominio | Viagem aérea |
| Intenção | Mostrar voo |
| Cidade Origem | São Paulo |
| Data Origem | Terça-feira |
| Hora Origem | Manhã |
| Cidade Destino | Rio de Janeiro |

E a frase a seguir

Me acorde amanhã as 6

Poderia gerar a representação abaixo:

|  |  |
| --- | --- |
| Dominio | Despertador |
| Intenção | Ativar despertador |
| Hora | 01/01/2017 06:00:00 |

O método usado no sistema GUS original é usar regras escritas manualmente.

Entre os métodos para serem usadas para extrair estão expressões regulares. Outra forma é usar gramática semântica com suas regras. Uma gramática semântica é uma gramática livre de contexto onde o lado esquerdo de cada regra corresponde a entidades semânticas sendo expressas como na forma a seguir:

MOSTRAR -> me mostre | posso ver

HORA\_PARTIDA\_RANGE -> (antes | depois | em torno de) HORA | da manhã | da tarde | da noite |...

HORA -> uma | duas | ...

VOOS -> voo | voos

ORIGEM -> (de | da) CIDADE

DESTINO -> para CIDADE

CIDADE -> São Paulo | Rio de Janeiro | Brasília

Gramáticas semânticas podem ser analisadas por qualquer algoritmo de analise de gramatica livre de contexto, dando como resultado uma hierarquia e rotulando cada no analisado da entrada.

Alguns problemas podem emergir desses métodos, exemplo deles é lidar com negação. Se um usuário disser que “não pode ser na Terça-feira” o sistema pode interpretar “Terça-feira” como um objetivo e não como uma restrição aos objetivos possíveis.

A abordagem usando regras costuma ser utilizada. Ela tem por vantagens a precisão e caso o domínio seja pequeno pode ser o suficiente para os casos necessários. Porém essas regras podem caras e demoradas de se criar.

Uma alternativa é usar aprendizado de maquina supervisionado. Partindo do pressuposto que existe um conjunto de dados de teste que associa cada sentença com a semântica correta, e um modelo de sequencia que relacione sentenças aos conteúdos dos slots.

Por exemplo, dada a sentença:

Eu quero viajar para São Paulo na terça-feira à noite, por favor

Pode-se aplicar um classificador 1-de-N (regressão logística, rede neural, etc) que usa features da sentença como N-grams de palavras para determinar que o domínio é “Viagem aérea” e a intenção é “Mostrar voo”.

**Capitulo 5 - 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 telegrama ao chatbot. Ele recebe a mensagem, interpreta ela 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 deve ser realizada.

**5.1 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 no telegram. O usuário que deseja conversar com ele precisa adiciona-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 arquivo de perguntas e respostas.

**5.2 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.

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. Abaixo um exemplo dos dados usados para treinamento.

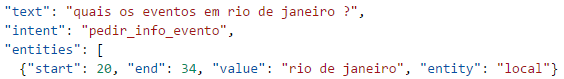


Os dados de exemplo são separados em arquivos, sendo um arquivo por intenção. Tal divisão não é necessária, mas ajuda na hora de fazer ajustes e correções.

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



Indica que “rio de janeiro” e “RJ” indicam o mesmo valor.



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.

**5.3 Treinamento do modelo**

O processo de treinamento do modelo ocorre da seguinte forma:

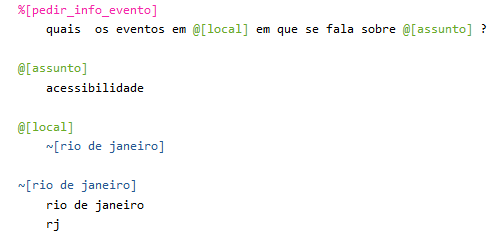
Os dados de exemplo são carregados na aplicação. De acordo as informações especificadas no arquivo config-spacy.json, é feito o treinamento. Então são gerados os arquivos de modelo para serem usados no interpretador de mensagens do chatbot.

**5.4 Criação dos dados**

Para criação de exemplos foi utilizada a ferramenta Chatito (PIMENTEL, 2018). Chatito é uma ferramenta que gera datasets especificos para serem usados em chatbots. Ela utiliza uma Linguagem de Dominio 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 nos formatos usados para os frameworks Rasa NLU e Snips 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.]).

O template da DSL é demonstrado no exemplo abaixo:



%[pedir\_info\_evento] é a intenção.

@[local] e @[assunto] são as entidades.

O que está escrito logo abaixo de @[local] e @[assunto] são valores possíveis que serão usados dos dados a serem gerados.

~[rio de janeiro] indica que esse valor possui sinônimos. Nesse caso, “rj”.

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

**5.5 Dominio**

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.

pedir\_mais\_info\_evento: estabelece que o evento perguntada será alvo de mais perguntas.

pedir\_mais\_info\_palestra: estabelece que a palestra perguntada será alvo de mais perguntas.

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.

buscar\_mais\_info\_palestra: define qual vai ser a palestra sobre a qual será buscada mais informação.

buscar\_mais\_info\_palestrante: define qual vai ser o palestrante sobre o qual será buscado mais informação.

buscar\_mais\_info\_evento: define qual vai ser o palestrante sobre o qual será buscado mais informação.

oferecer\_ajuda: envia uma mensagem oferecendo ajuda.

**5.6 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.

**5.7 Execução**

O usuário envia mensagem para o chatbot.

Chatbot interpreta a mensagem identificando a intenção e as entidades. Exemplo do resultado da identificação abaixo.



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

**5.7 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 também 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, servir como interface de comunicação com outras APIs e criar ferramentas customizadas para o usuário.

**5.9 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 tornando seu uso mais fácil.

**5.10 Bibliotecas**

spaCy (spaCy, 2018) é uma biblioteca para processamento de linguagem natural utilizando Python e Cython que oferece tokenization, reconhecimento de entidades, POS tagging entre outras funcionalidades.

Scikit-learn (sklearn) (PEDREGOSA et al., 2011) é uma biblioteca de aprendizado de máquina utilizando Python que oferece implementações de algoritmos para regressão, classificação e clusterização.

SQLite é uma biblioteca que implementa 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.

**Capitulo 6 – Conclusão**

O proposito 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.

**Capitulo 7 – Referencias 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.

ALICE Bot - Artificial Linguistic Internet Computer Entity. *ALICE A.I. Foundation* (2016). Online em <http://www.alicebot.org/>.

Laven, S. (2016). Chatterbot Central *- The Simon Laven Page*. <http://www.simonlaven.com/>

GREEN JR, B. F. et al. **Baseball: an automatic question-answerer**. Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference. **Anais**...ACM, 1961Disponível em: <http://dl.acm.org/citation.cfm?id=1460714>. Acesso em: 15 jun. 2017

Woods, W. A. (1973) "Progress in natural language understanding: an application to lunar geology." In: Proceedings of the June 4-8, 1973, national computer conference and exposition. ACM

Winograd, Terry (1972). "Understanding natural language." Cognitive psychology, v.3, n.1, pp.1-191.

Schank, R. C., e Abelson, R. (1975). Scripts, plans, and knowledge. New Haven, CT: Yale University.

SCHANK, Roger C.; ABELSON, Robert P. Scripts, plans and knowledge. 1975, [S.l.]: Tbilisi, USSR, 1975. p.151–157.

Allen, J.F. (1995). Natural Language Understanding. The Benjamin Cummings Pub. Company, New York.

ALLEN, J. **Natural language understanding**. 2nd ed ed. Redwood City, Calif: Benjamin/Cummings Pub. Co, 1995.

HAYES-ROTH, F.; LESSER, V. R.; REDDY, D. The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty. **Readings in artificial intelligence: a collection of articles**, p. 349, 1981.

Bolt, R. A. (1980) “Put-that-there”: Voice and gesture at the graphics interface, v.14, n.3, ACM.

HASEGAWA, O. et al. **Active agent oriented multimodal interface system**. IJCAI. **Anais**...1995Disponível em: <http://ijcai.org/Proceedings/95-1/Papers/011.pdf>. Acesso em: 15 jun. 2017

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. Text. 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.

Statistical natural language processing and corpus-based computational linguistics:

An annotated list of resources, 2003. Stanford University, http://www-nlp.stanford.edu/links/statnlp.html.

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

Proceedings of the 2003 International Symposium on Reference Resolution and Its Applications to Question Answering and Summarization, Venice, Italy, June 23–24 2003.

Bird, Steven, Edward Loper and Ewan Klein (2009), Natural Language Processing with Python. O’Reilly Media Inc.

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.

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

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.