**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 as bibliotecas Rasa NLU e Core para processamento de linguagem natural e aprendizado de máquina.

**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.

**1 - 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.

**1.1 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.

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.

**1.2 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.

**1.3 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.

***1.3.1 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.

**1.4 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.

**2 Histórico**

É possível dividir a história dos chatbos em três gerações em relação as técnicas utilizada para conversar com o usuário. [Neves, 2005]

**2.1 Primeira geração**

A primeira geração se baseia casamento de padrão simples. O primeiro chatbot de maior reconhecimento nessa geração foi ELIZA [[Weizenbaum, 1966]]. Ele simulava um psicanalista da rogeriano.

Os chatbots dessa geração usam técnicas simples para manter diálogo, usando casamento de padrão em busca de palavras e frases chaves. Eles utilizam pares <Pergunta-Resposta> para conversar. Ao receber uma entrada do usário, é buscado o padrão mais próximo para então mostrar a resposta associada a esse padrão.

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 doeças psiquiatricas.

Apesar do sucesso, as técnicas usados neles são limitadas no que se reflete em dialogos com poucos detalhes, principalmente em diálogos demorados.

Para manter uma conversa proxima a real é 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.

**2.2 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 e também a dividir o chatbot em módulos.

A partir dessa geração, chatbots passam a contar com modelo de processamento morfossintatico para identificar variações morfologicas e sintáticas das frases. Isso permite converter frases semelhantes para uma forma padrão com base em uma tabela de substituição [Neves, 2005]. Outro avanço foi a inclusão de bases de conhecimento de dominio especifico, baseado nas ideias de construçao de sistemas especialistas.

O chatbot que merece destaque dessa geração é JULIA [Mauldin, 1994], desenvolvido por um ambiente de entretenimento multiusuário de 1989 ((TinyMUD29)). JULIA interagia como se fosse um humano com outras pessoas por meio de salas de chat. Ela conseguia guardar informações sobre o usuário a partir de conversa com ele. Essa funcionalidade tinha por objetivo dar maior credibildiade ao dialogo.

A base de diálogo de JULIA era feita com um grafo (Rede de Ativação) em que cada nó possui um titulo, palavras-chave, um valor de ativação e um molde de replica. Os nós são ligados por arestas com pesos positivos e negativos. Sendo que o valor de ativação poderia variar de acordo com a entrada passada para o chatbot. O nó com maior valor era escolhido.

JULIA teve grande influencia no design de chatbots, e serviu como base para outros bots, como JFRED [30 <http://www.turinghub.org/JFRED.php>]

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 fechamento), além de ter quer oferecer tratamento a perguntas desconhecidas.

**2.3 Terceira geração**

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

A linguagem mais importante dessa geração foi a AIML (Artificial Intelligence Markup Language) [Wallace, 2009]. AIML possui estrutura simples, inspirada na técnica utilizada na ELIZA, mas apresenta maior capacidade de processamento de conversas do que os chatbots das duas primeiras gerações.

**2.4 Evolução dos sistemas conversacionais**

Os primeiros sistemas conversacionais eram Sistemas de Pergunta-Resposta restritos a perguntas e respostas previamente catalogadas. Esses sistemas são do inicio da década de 1960. São exemplos: BASEBALL [Green et al.], LUNAR [Woods, 1973] e SHRDLU [Winograd, 1972].

O primeiro sistema de grande divulgação a conseguir manter diálogos, ainda que simples, foi ELIZA [Weizenbaum, 1966]. Esses sistemas foram nomeados Chatterbots. Eles conviveram na década de 70 junto com os Sistemas de Pergunta-Resposta.

Também na década de 70 existiam os Sistemas para compreensão e geração de histórias, que apesar de não serem sistemas de conversação, ajudaram nas pesquisas de Processamento de Linguagem Natural, o que também ajudou nas implementações de sistemas de conversação. [Schank et al. 1975] [Allen, 1995]

Na década de 80, acompanhando os avanços do processamento de voz, ganharam espaço os sistemas conversacionais via voz. Exemplos desses sistemas são: *Hearsay*‐*II* [Erman et al., 1980] e “*Put*‐*That*‐*There*” [Bolt, 1980].

No final da decada de 1980 é criado o primeiro sistema conversacional de voz com imagem humana: "*The Knowledge Navigator*”. (HASEGAWA et al., 1995)

**3 Tipos de agentes conversacionais**

**3.1 Chatbots baseados em regras: ELIZA e PARRY**

ELIZA foi projetado para imitar um psicólogo rogeriano.

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

Men are all alike.

IN WHAT WAY

They're always bugging us about something or other.

CAN YOU THINK OF A SPECIFIC EXAMPLE

Well my boyfriend made me come here.

YOUR BOYFRIEND MADE YOU COME HERE

He says I'm depressed much of the time.

I AM SORRY TO HEAR YOU ARE DEPRESSED

..

..

WHO ELSE IN YOUR FAMILY TAKES CARE OF YOU?

My father

YOUR FATHER

You are like my father in some ways.

WHAT RESEMBLANCE DO YOU SEE

You are not very aggressive but I think you don't want me to notice that.

WHAT MAKES YOU THINK I AM NOT AGGRESSIVE

You don't argue with me.

WHY DO YOU THINK I DON'T ARGUE WITH YOU

You are afraid of me.

DOES IT PLEASE YOU TO BELIEVE I'M AFRAID OF YOU

My father is afraid of everybody.

WHAT ELSE COMES TO MIND WHEN YOU THINK OF YOUR FATHER

Bullies.

DOES THAT HAVE ANYTHING TO DO WITH THE FACT THAT YOUR BOYFRIEND MADE YOU COME HERE

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

(0 YOU 0 ME) [padrão]

->

(WHAT MAKES YOU THINK I 3 YOU) [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:

You hate me

A reposta que viria de ELIZA seria:

WHAT MAKES YOU THINK I HATE YOU

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

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

Ex.:

I know everybody laughed at me

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

(I \*) -> (You say you 2)

E geraria por resposta:

YOU SAY YOU KNOW EVERYBODY LAUGHED AT YOU

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

WHO IN PARTICULAR ARE YOU THINKING OF?

Se nenhuma palavra chave é encontrada, ELIZA incentiva a pessoa com que se está conversando a falar mais usando termos como “PLEASE GO ON” ou “THAT’S VERY INTERESTING”.

O framework de ELIZA com usas regras de padrão/resposta ainda é usado em sistemas mais novos como ALICE.

Alguns anos depois, foi criado o chatbot PARRY (Colby et al., 1971),, também com foco na psicologia, mas usado para estudar esquizofrenia. Além do casamento de padrão semelhante ao usado em ELIZA, 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.

**3.2 Chatbots baseados em corpus**

Chatbots baseados em corpus, em vez de usar regras construídas manualmente, eles extraem as informações de conversas entre pessoas ou entre pessoas e maquinas. As respostas de chatbots também podem ser extraídas de textos que não são conversas.

Existem dois tipos de chatbots baseados em corpus: sistemas baseados em recuperação de informação.

Assim como chatbots baseados em regras, 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.

**3.2.1 Chatbots baseados em Recuperação de informação (RI)**

O principio por trás desses chatbots é gerar uma resposta ao usuário repetindo alguma informação extraída de um corpus de texto por pessoas. A diferença entre as implementações desses chatbots está em qual corpus é utilizado e como decidir qual informação é considerada boa para ser usada como resposta.

Uma escolha é usar dados de conversas entre humanos. Esses dados podem vir de plataformas como Twitter. Outra escolha é usar diálogos de filmes. Após o chatbots entrar em ação, as respostas obtidas do usuário humano podem também serem usadas para treinar o bot.

Dado o corpus e a sentença do usuário, sistemas baseados em RI podem usar qualquer um algum algoritmo de recuperação para escolher uma resposta apropriada de um corpus. Os métodos mais 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 et al. 2009, Leuski and Traum 2011).

1. Retornar o turno mais similar: dado uma consulta q do usuário e um corpus C, retornar t em C que é mais similiar 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 (usando tf-idf).

Retornar a reposta ao turno mais similar pode parecer o metodo mais intuitivo, mas retornar o turno mais similar costuma demonstrar melhores resultados na prática (Ritter et al. 2011, Wang et al. 2013).

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.

Em vez de usar corpus de conversas humanas, a abordagem usando RI pode extrair as respostas de texto que não seja conversa. Por exemplo, o COBOT chatbot (Isbell et al., 2000) gerava respostas a partir de frases de um corpus que combinava o Manifesto do Unabomber por Theodore Kaczynski, artigos sobre abdução alien e os roteiros de “O Grande Lebowski” e “Planeta dos Macacos”. Chatbots que desejam gerar turnos para perguntas informativas podem usar texto da Wikipédia para extrair as sentenças que contenham as respostas (Yan et al., 2016).

**3.3 Agentes de dialogo basedos em frame**

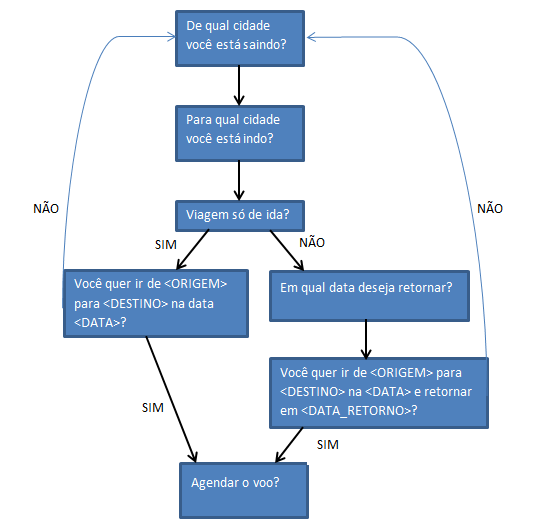
Sistemas de dialogo modernos orientados a tarefas são baseados em ontologia de domínio, um estrutura de conhecimento representando os tipos de intenções que o sistema pode extrair das sentenças dos usuários. A ontologia define um ou mais frames, sendo cada um uma coleção de slots, e define os valores 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).

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 linha área.

Tipos no GUS, assim como em Agentes de dialogo basedos em frame, tem uma estrutura hierárquica. Por exemplo, o tipo data é ele mesmo um frame com slots, como números inteiros ou dias de semana.

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 deseja e então realizar a ação desejada pelo usuário (responder uma pergunta, agendar uma viagem). 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.



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 estado 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 controla a conversa com o usuário. Ele 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.

Quem está no controle de uma conversa tem a iniciativa na conversa. Sistema que controlam a conversa desse jeito são chamados de sistema-iniciativa. Diferentemente, numa conversa normal entre pessoas, a iniciativa é alternada entre os participantes da conversa (Bobrow et al. 1977,Walker and Whittaker 1990).

A vantagem de um sistema com apenas uma iniciativa e estados finitos é que sempre se sabe qual pergunta está sendo respondida. Isso significa que o sistema pode se preparar com um modelo de linguagem afinado para as respostas para cada pergunta e isso também faz o entendimento da linguagem natural mais fácil. No entanto, essa arquitetura de estados finitos costuma ser aplicada apenas a tarefas simples como informar numero de cartão de credito, nome ou senha.

No entanto, a maioria das aplicações o usuário precisa de mais flexibilidade para expressar seus desejos. No planejamento de viagens uma frase pode preencher mais de um slot ao mesmo tempo:

Eu quero um voo de São Paulo para o Rio de Janeiro só de ido após as 17:00 na terça-feira.

Em alguns casos onde existem várias frames, um usuário pode dizer algo para alterar os frames, indo, por exemplo, de reserva de voo para reserva de aluguel de carro:

Eu gostaria de alugar um carro assim que eu chegar ao aeroporto

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.

A arquitetura do GUS também tem regra associadas aos slots. Por exemplo, pode ter uma regra associada ao slot Destino sendo que o conteúdo desse slot também será usado para preencher a informação de local de estadia, caso se peça um hotel.

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.

Por causa dessa necessidade de alterar o controle, a arquitetura do GUS é um sistema de regra de produção. Nesse sistema diferentes tipos de entradas ativam produções diferentes. As regras de produção podem trocar o controle de acordo com fatores como entrada do usuário e histórico do dialogo.

Sistemas de dialogo profissionais disponibilizam interfaces e bibliotecas para facilitar a criação de sistemas com estados finito ou regras de produção.

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 |

Agentes de dialogo podem extrair essas informações das frases dos usuários por meio de abordagens de parsing semântico.

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 | manhã | tarde | 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”.

Após o preenchimento dos slots pode-se usar um classificador que usa features similares da sentença para predizer qual slot que o usuário deseja preencher. Em adição as features unigram, bigram e trigam de palavras pode-se usar também features de entidades nomeadas ou features indicando que uma palavra está em um léxico particular e o classificador retornaria o nome do slot. Um segundo classificador pode depois ser usado para determinar o preenchimento do slot nomeado.

Em contexto industriais, sistema baseados em aprendizado de maquina para preenchimento de slots costumam ser inicializados a partir de um sistema baseado em regras usando uma abordagem de aprendizado semi-supervisionada. Um sistema baseado em regras é criado para o domínio, e um conjunto de teste é rotulado. A medida que novas entradas vindas do usuário são recebidas, elas colocadas em par com o rotulo feito pelo sistema baseado em regras para assim formar tuplas. Um classificador pode então ser treinado usando essas tuplas e usando o conjunto de teste para comparar a performance com o sistema baseado em regras. A medida que mas amostras se tornam disponíveis, o classificador pode obter resultados melhores que o sistema que utiliza regras (Suendermann et al.,2009).

**Avaliando preenchimento de slots**

Para se avaliar o uso de entendimento de linguagem natural para preenchimento de slots, pode-se usar a métrica a seguir:

Taxa de erro de slot para uma sentença = (nº de slots inseridos/deletados/substituídos)/(nº de slots para a sentença)

Considere por exemplo, a frase a seguir:

Marque uma reunião com Fernando as 10:30 na sala 104.

E considere que foi extraída as seguintes informações e sua estrutura:

|  |  |
| --- | --- |
| Slot | Conteúdo |
| PESSOA | Fernando |
| HORA | 11:30 |
| SALA | 104 |

Usando a métrica citada anteriormente, a taxa de erro é de 1/3, já que a hora está errada.

* 1. **Avaliando sistemas de dialogo**

Para saber quão bom é o funcionamento do sistema é precisa avalia-lo. Se a tarefa é simples e clara pode-se simplesmente medir o sucesso ao desempenhar a tarefa (o voo desejado pelo usuário foi reservado?).

Agora, para obter uma avaliação mais detalhada pode-se analisar a taxa de satisfação do usuário com o sistema. Para isso, é preciso colocar usuários para interagir com o sistema e após a interação eles preenchem um questionário sobre a experiência que tiveram Walker et al. (2001). As respostas são associadas a valores que variam de 1 a 5 e então é calculada uma média sobre todas as avaliações.

Como costuma ser economicamente inviável fazer avaliações com humanos após cada mudança no sistema, pode-se usar fatores para avaliação de performance que tenham correlação positiva com as avaliações humanas. Os critérios que podem ser usados para avaliar esses fatores é buscar aumentar o sucesso de realização da tarefa e diminuir os custos.

Sucesso na realização da tarefa: O sucesso pode ser medido avaliando quão correta é a solução final. Para arquiteturas baseadas em quadros, isso pode significar o número de slots que foram preenchidos. As vezes a percepção do usuário que a tarefa foi completa é um indicador melhor do que a realização da tarefa . (Walker et al., 2001).

Custo da eficiência: Custos da eficiência são medidas da eficiência do sistema em ajudar usuários. Isso pode ser medido pelo tempo gasto no dialogo em segundos, o número de turnos utilizados ou o número total de consultas (Polifroni et al., 1992).Outra métrica é contar o número de turnos do usuário necessários para corrigir erros dividido pelo numero de turnos (Danieli and Gerbino 1995,Hirschman and Pao 1993).

Custo de qualidade: custo de qualidade é a medida de outros aspectos que afetam a percepção do sistema pelo usuário. Exemplos dessas medidas: numero de vezes que o sistema não conseguiu retornar uma sentença, número de time-outs do sistema pois o usuário demorou muito para responder,quão bem o sistema entendeu e respondeu o usuário, quão apropriada são as perguntas, respostas e as mensagens de erro (Zue et al. 1989, Polifroni et al. 1992).

Chatbots costumam ser avaliados por humanos. A avaliação com slot-filling usada com sistemas de dialogo baseados em tarefas não é apropriado para essa avaliação. E métricas de word-overlap como BLEU para comparar respostas do chatbot com respostas que seriam geradas por humanos, geram resultados ruins pois existem muitas respostas possíveis para cada turno. Essas métricas funcionam melhor quando o conjunto de respostas possíveis é menor.

Apesar da necessidade de seres humanos para avaliar chatbot, está começando a se criar modelos para avaliação automática. O classificador ADEM (Lowe et al., 2017a) é treinado para com um conjunto de respostas rotuladas por humanos com quão apropriadas elas são e então ele aprende a prever esse rotulo a partir do contexto do dialogo e das palavras da resposta do chatbot.

Outros paradigma é adversarial evaluation (Bowman et al. 2016, Kannan and Vinyals 2016, Li et al. 2017), inspirada no teste de Turing. Nesse paradigma treina-se um classificador para distinguir entre respostas geradas por humanos e respostas geradas por máquina. Quão melhor for o chatbot em enganar esse classificador, melhor o sistema.

**4 Mecanismos**

**Tokenização, pos tagging, lematização,**

* 1. **POS tagging**

Part of speech tagging é o processo de rotular cada palavra de uma frase com a sua classe gramatical a partir da definição da palavra e de seu contexto.

POS tagging é um dos primeiros etapas no processamento de linguagem natural. A partir das sentenças é atribuída uma POS tag a cada palavra da sentença identificado a qual categoria léxica ela pertence.

POS tagging pode ser encarada como uma analise morfológica simplificada, pois enquanto a analise morfológica tem por objetivo encontrar a estrutura da palavra (raiz, sufixo, prefixo, etc), no POS tagging só é atribuida uma tag a palavra da forma como ela está.

**Classes gramaticais**

Para se identificar a qual classe uma palavra pertence é preciso saber quais são as classes possíveis. Existem três grandes classes gramaticais: substantivo, verbo e adjetivo (Pustet, 2003). O tamanho e o conteúdo do conjunto de tags disponíveis é uma questão linguística, no entanto existem diferentes conjuntos de tags que podem ser escolhidas. Os corpus mais usados na língua inglesa e o tamanho do seu conjunto de tags: Brown corpus (87 tags), Lancaster-Oslo/Bergen (LOB) corpus (135 tags), Penn Treebank and Wall Street Journal (WSJ) corpus (48), and Susanne corpus (353 tags).

**Problemas**

Dados o conjunto de tags disponíveis para classificação das palavras e a sentença a ser analise, surgem os problemas abaixo:

1. Palavras ambíguas: em uma frase, algumas palavras podem pertencer a mais de uma classe gramatical o que torna essa classificação um problema.

Ex.: We can can the can.

Na frase acima, a palavra “can” pode ser tanto um auxiliar, um verbo e um substantivo. Para desfazer essa ambiguidade é preciso levar em consideração não apenas as palavras individuais, mas também o contexto onde elas estão inseridas, o assunto do texto onde elas estão sendo usadas, a função que elas desempenham na frase. Esse é um problema simples para seres humanos resolverem, mas que ainda é complicado para máquinas.

1. Palavras desconhecidas: no caso da abordagem baseada em regras, vai sempre existir alguma palavra que é coberta pelas regras escritas. Assim como nos métodos estatísticos terão algumas palavras que não estão presentes no corpus usado no treinamento. É preciso levar em consideração esse problema na hora de implementar o tagger gerando alguma informação sobre a palavra desconhecida de forma que essa informação possa ser usada no processos futuros que usem a classificação.
2. Consistência do conjunto de tags disponíveis: ao usar mais tags, pode-se ter mais informações associadas as palavras, mas passar a tomar mais tempo para diferenciar entre as tags. Uma forma de lidar com isso é atribuir mais de uma tag a palavra e deixar para etapas posteriores decidirem qual tag que deve ser utilizada pois essas etapas teriam acesso a informações que não estão disponíveis para o pos tagger.

**Métodos de POS tagging**

Métodos baseados em regras

Os primeiros sistemas de POS tagging eram baseados em regras, onde um conjunto de regras é escrito e então aplicado a o texto que se deseja analisar. Possivelmente o primeiro sistema de POS tagging foi feito por Klein and Simpson (1963) e era composto pelas regras e por um dicionário. O dicionário era usado para restringir as tags possíveis das palavras para aquelas que existem no dicionário. Aí então as regras eram aplicadas para palavras desconhecidas que tinham as palavras a esquerda e direita sem ambiguidade. O maior problema dessa abordagem era a necessidade de escrever essas regras, que era trabalhoso e precisava de conhecimento de linguística.

**Métodos baseados em Cadeias de Markov**

Os métodos baseados em regras começaram a ser substituídos por métodos estocásticos no início da década de 90. O que permitiu essa mudança foi a grande disponibilidade de dados: léxicos, corpora, corpora em duas línguas. A partir dessas informações é possível aprender padrões

Existem duas dificuldades nessa etapa. Uma é a ambiguidade, pois algumas palavras de uma língua podem ter mais uma tag associada a ela. Outro problema são as palavras desconhecidas pois não se consegue atribuir tag a ela. O jeito de resolver esses problemas é levar o contexto em volta da palavra em consideração para se escolher a categoria mais provável para palavra.

Existem diversos métodos para esse processo. Elas podem ser divididas em dois grupos: métodos baseados em regras e métodos estatísticos. O primeiro grupo usa de regras linguísticas para categorizar as palavras. Essas regras tinham que reunidas manualmente o que era um grande problema. À medida que mais recursos de linguagens foram ficando disponíveis, os métodos estatísticos começaram a substituir os métodos baseados em regras.

O processo de POS tagging é usado em diversas tarefas associadas a aplicações de processamento de linguagem natural. Tarefas como parsing (análise sintática) ou parsing parcial (analise sintática restrita a alguns tipos de frases). Outra aplicação é extração de informação, que tem por objetivo extrair informação estruturada de texto sem estrutura. Reconhecimento de Entidade Nomeada é uma subtarefa de extração de informação que faz uso de tagging para identificar as entidades de um texto e as relações que existem entre elas (Cardie, 1997).

Sistemas de Information retrieval e de question answering, tradução automatizada, word-sense disambiguation também usam tagging.

**4.3 AIML**

AIML oferece tratamento de informação em quatro niveis linguisticos: morfosintatico, semantico, discursivo e pragmatico. Para oferecer esses tratamentos, a AIML usa variáveis de controle e a base de conehcimento da propria linguagem.

Nivel morfosintatico:

- Uso de tags para fazer substituições para corrigir erros de digitação ou normalizar termos para serem processados de forma unica pelo bot.

Ex.: <substitute find=”becuase” replace=”because”>

* Base de categorias faz o tratamento sintatico da entrada do usuário. Isso é feito por blocos identificados pelas tags <category></category>. Dentro desse bloco ficam um padrão (<pattern>) que realizar um casamento com a entrada do usuário e este padrão está associado a uma ou mais respostas (<template>). Podendo usar simbolos especiais (\* e \_) para casar padrão com qualquer sequencia de caracteres.

Nivel semantico

-A base de descrição do chatbot é formada por uma ´serie de atributos para identificar as caracteristicas do bot.

-Base de topicos que é separada em categorias. Por exemplo, é possivel criar categorias dentro de um topico. Essas categorias só serão utilizadas quando o tópico em questão for referenciado na conversa.

Nivel Discursivo

- Base de substituição de pronomes. Uso da tag <substitute> para trocar os pronomes, sendo que essa tag está associada a tag <person> indicando assim se tratar de uma pessoa da conversa.

Ex.: <person>

<substitute find=”he was” replace=”I was”/>

</person>

Variáveis de controle de deiticos. Essas variáveis são usadas para definir termos relacionados ao tempo e são definidas pela tag <set> e obtidas pela tag <get>.

Nivel pragmatico

Variáveis com informação sobre o usuário. São definidas pela tag <set> e obtidas pela tag <get>. Guardam informações sobre o usuário que está conversando.

ALICE [ALICE Bot, 2016] foi implementada usando AIML e venceu três edições do premio Loebner e uma edição do Chatterbox Challenge. Devido ao sucesso, ALICE ajudou na popularização do AIML como técnica usada na implementação de chatbots.

**4.4 Processamento de Linguagem Natural**

Processamento de Lingaugem Natural (PLN) é uma das áreas de pesquisa mais antiga dentro de Inteligencia 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 [LDC, 2003], e o British National Corpus [Bri, 2003]. 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

and TREC [Nat, 2003] onde se competiam sistemas em tarefas comuns também foram uma força incentivadora de avanços.

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

Os maiores avanças em PLN tem sido em analise sintática. Existem dois passos na maioria da implementações da analise sintática: part-of-speech (POS) tagging; e parsing. POS tagging consiste em atribuir a cada token a parto f speech indicando sua função gramatical, como substantivo singular ou adjetivo comparativo. Existem vários algoritmos de aprendizado capazes de regras de POS tagging de corpora etiquetada, incluindo abordagens transformation-based e maximum entropy-based [Brill, 1995; Ratnaparkhi,

1996].

Duas abordagens são apropriadas para agentes conversacionais. Chunking, ou parsing robusto, consiste de métodos de estado finito para fazer separar o texto em chunks, isto é, frases sem modificadores post-head. Existem métodos rápidos e precisos para gramatica em chunk [Cardie et al., 1999; Abney, 1995]. No entanto, como chunking é um método de estado finito ele não consegue reconhecer estruturas com recursão infinita, como é o caso de algumas orações subordinadas. Gramáticas Livres de Contexto conseguem expressar recursão infinita no entanto tem o custo de de usar algoritmos de parsing mais complexos e demorados. Várias técnicas foram desenvolvidas para aprender gramáticas livres de contexto de tree banks [Sta, 2003]. O desempenho das mais precisas dessas técnicas, como lexicalized probabilistic contexto free grammars [Collins, 1997], pode ser bem alto, o tempo para fazer o parse pode ser alto também. Agente conversacionais na Web pode ter que lidar como um grande numero de declarações por segundo, então a demora para fazer o parse pode ser um fator significante na hora de escolher a abordagem. Além disso, a maioria das declarações que o chatbot tem que lidar tem estrutura simples.

A tarefa de resolver referencias na analise de discurso é pesquisada [Ref, 2003], mas um conjunto de regras é suficiente para lidar com os casos mais comuns. Por exemplo, em casos mais simples de anáfora, buscar pela palavra mais próxima pode ser suficiente para indetificar a reerencia.

Ex.: Eu quero um computador e um som se um estiver a venda.

Nesse caso, é mais provável que o artigo “um” esteja fazendo referencia ao “som” do que ao computador.

Existem mais recursos disponíveis para analise sintática do que para as analises semântica e pragmática. WordNet,uma base de dados lexica, disponibiliza semânticas léxicas para sentenças que já passaram pelo parser [Fellbaum,1999]. A maioria das abordagens usadas na analise pragmática se baseiam no contexto para resolver a ambiguidades ou regras ad-hoc.

Três abordagens têm comumente sendo aplicada ao gerenciamento do dialogo. A primeira abordagem usa maquina de estados finita (MEF) como modelo de dialogo. Transições na MEF correspondem a declarações e os estados correspondem aos objetivos do agente. Essa abordagem funciona para diálogos simples onde todas combinações possíveis de diálogos podem ser antecipadas. Porém, ela não é própria para os casos onde a ordem das declarações é imprevisível, com a entrada contendo varias informações ou diálogos onde o controle da conversação muda entre os interlocutores (Dialogo com iniciativa mista). Iniciativa mista ocorre quando uma pessoa responde uma pergunta com outra pergunta.

A segunda abordagem usa templates ou slots correspondentes a informação a ser descoberta. Essa abordagem consegue lidar com uma ordem imprevisível de declarações e também com declarações composta de forma melhor que a abordagem usando MEF, mas ainda não oferece grande ajuda para resolver o problema da iniciativa mista.

A terceira abordagem uma pilha de objetivos para gerenciar os objetivos do dialogo. Essa abordagem consegue lidar com mudança de tópicos colocando o estado do objetivo correspondente ao novo tópico na pilha, e depois retirando ele quando o tópico se encerrar. Essa abordagem é mais difícil de implementar que as anteriores mas consegue lidar com diálogos de iniciativa mista.

Após o agente conversacional ser colocado para iniciar sua atividade, sua precisão vai melhorando. A medida que o agente vai aprendendo com seus erros a quantidade de falsos positivos e falsos negativos diminui. O aprendizado começa do chatbot ser colocado em ação e continua enquanto ele tem que lidar com a grande quantidade de mensagens. Mesmo quando o agente estiver com grande precisão, o aprendizado continua necessário para saber se o conhecimento continua atualizado de acordo com as mudanças que ocorrem na organização que usa o agente.

Três mecanismos são usados para melhorar a qualidade. Primeiro, as transcrições das conversas são guardadas para analise automática e refinada com o auxilio de um especialista no assunto tratado na conversa. Segundo, uso de ferramentas para avaliar o desempenho do agente. Terceiro, o agente conversacional realiza auto-avaliação da qualidade de seu comportamento. Um forma de realizar essa avaliação é associar um classificação a cada resposta indicando quão confiável ela foi.

**4.5 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)

**Capitulo 5 - Implementação**

A implementação feita é um chatbot utilizando a linguagem de programação Python, a biblioteca Rasa NLU (rasa\_nlu, 2018).

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.



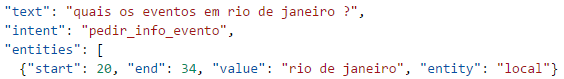
Os dados de exemplo são separados em arquivos, sendo um arquivo por intenção. Tal divisão não é necessária, mas ajuda na hora de fazer ajustes e correções.

regex\_features servem para...

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:

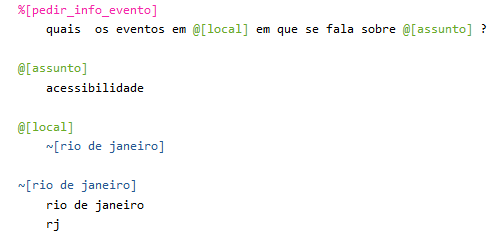
Os dados de exemplo são carregados na aplicação. De acordo as informações especificadas no arquivo config-spacy.json, é feito o treinamento. Então são gerados os arquivos de modelo para serem usados no interpretador de mensagens do chatbot.

**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”.

Slot filling chatbots

A implementação usa duas bibliotecas RASA NLU, para compreensão de linguagem natural e RASA Core, para gerenciamento de conversa. (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.

5 entidades

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

Os slots são os mesmos que as entidades.

Uso de RASA NLU

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

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.

**Capitulo 6 - Aplicações atuais, Limitações e Tendências futuras**

Aplicações atuais

Chatbots são usados como interfaces para sistemas [Laven, 2016] e como acompanhantes para sistemas imersivos. Como aplicações isoladas para entretenimento (Eliza e Alice), assistentes virtuais, bots para FAQ (Frequent Asked Questions) ou SAC (Serviço de Atendimento ao Consumidor), chatbots para jogos digitais, para uso em ensino e educação.

[Laven, 2016], [Alice Bot, 2016], [Chatbots.org, 2016]

Limitações

Chatbots apresentam limitações para manter diálogos reais sendo essas limitações provenientes principalmente ao lidar tanto com interpretação quanto geração de linguagem natural, principalmente no que se refere a analise de significado do texto. Assim ocorre a dificuldade de manter diálogos coerentes.

Outro problema é proveniente da criação e manutenção de base de conhecimento para ser consultada pelo chatbot. Por exemplo, caso o bot não obtenha a informação desejada na base de conhecimento é preciso contornar esse problema de alguma forma, seja assumindo desconhecimento do assunto ou oferecendo algum lugar para se obter a informação buscada de forma a não parar o diálogo.

Tendencias futuras

Crescimento do numero de chatbots que incorporem voz em sua comunicação.

**Capitulo 7 - Referencias**

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. (2009). “The Anatomy of A.L.I.C.E.”. Parsing the Turing Test. Springer. pp. 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.

LDC catalog, 2003. http://www.ldc.upenn.edu/Catalog/, University of Pennsylvania.

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

Text retrieval competition, 2003. National Institute of Standards and Technology,

http://trec.nist.gov/.

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.

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

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.