UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIAS
CURSO ENGENHARIA DA COMPUTAÇÃO

AUGUSTO ZANONI FRADE SOUZA SANTIAGO

DESENVOLVIMENTO DE UM SISTEMA DE CHATBOT PARA PERGUNTAS FREQUENTES SOBRE A LEI DE DIREITO AUTORAL

São Luís
2022

AUGUSTO ZANONI FRADE SOUZA SANTIAGO

DESENVOLVIMENTO DE UM SISTEMA DE CHATBOT PARA PERGUNTAS FREQUENTES SOBRE A LEI DE DIREITO AUTORAL

Trabalho de conclusão de curso apresentado ao curso de Engenharia da computação da Universidade Federal do Maranhão como requisito parcial para obtenção do grau de Bacharel em Engenharia da Computação.

Orientador: Profº. Dr. Bruno Feres de Souza

São Luís
2022

AUGUSTO ZANONI FRADE SOUZA SANTIAGO

DESENVOLVIMENTO DE UM SISTEMA DE CHATBOT PARA PERGUNTAS FREQUENTES SOBRE A LEI DE DIREITO AUTORAL

Trabalho de conclusão de curso apresentado ao curso de Engenharia da computação da Universidade Federal do Maranhão como requisito parcial para obtenção do grau de Bacharel em Engenharia da Computação.

Aprovado em ___/___/_____

BANCA EXAMINADORA

_____

Professor: Dr. Bruno Feres de Souza (Orientador)
Universidade Federal do Maranhão

_____

Professor: Profa. Dra. Maria da Glória Almeida Bandeira
Universidade Federal do Maranhão

_____

Professor: Profa. Dra. Vandecia Rejane Monteiro Fernandes
Universidade Federal do Maranhão

SANTIAGO, Augusto. **DESENVOLVIMENTO DE UM SISTEMA DE CHATBOT PARA PERGUNTAS FREQUENTES PARA LEI DE DIREITO AUTORAL**. Orientador: Dr. Bruno Feres de Souza. 2022. Monografia (Bacharel em Engenharia da Computação) - Universidade Federal do Maranhão, [*S. l.*], 2022.

”Se não tem sobreviventes, então quem conta as histórias”

(Ted Elliot e Terry Rossio; Piratas do Caribe, 2003)

Agradecimentos

Esses agradecimentos vão para todos que ajudaram para que fosse possível chegar tão distante. Um agradecimento a familiares e amigos.

Esse trabalho não seria possível sem o tempo e dedicação do professor doutor Bruno Feres de Souza, além dos momentos ministrando suas aulas extremamente enriquecedoras.

Um agradecimento especial a discente Gracelynne Oliveira Santos de biblioteconomia e sua orientadora Maria Da Glória Almeida Bandeira, pois sem elas esse trabalho não seria possível.

Gostaria de fazer um agradecimento não usual, mas como diria o Rapper Snoop Dogg, devemos valorizar as próprias conquistas, então esse agradecimento vai para mim por acreditar durante toda essa trajetória e fazer disso realidade.

Resumo

Chatbots existem há mais de 20 anos e são softwares baseados em inteligência artificial (IA) que buscam realizar uma comunicação entre a máquina e o ser humano com uso de um ramo da IA chamado de processamento de linguagem natural (NLP). A criação desse tipo de software se justifica devido a alta demanda por dúvidas com respostas objetivas, tornando esse processo muito mais ágil e democrático com a utilização de um *bot*. Em vista disso, esse trabalho tem como objetivo a implementação de um *chatbot* para atender a dúvidas frequentes dos usuários da biblioteca do campus da Universidade Federal do Maranhão. Para tanto, utilizou-se a linguagem de programação Python e o *framework* Rasa para implementação do *bot*, uma vez que o processo de desenvolvimento é oneroso e repleto de etapas. Ademais, devido a necessidade de um time multidisciplinar, parte do conteúdo do bot foi fornecido por uma discente da área de biblioteconomia da universidade. A sessão de testes foi realizada de forma automática com algoritmos de análise e métricas como precisão e *recall*, além da matriz de confusão para garantir a qualidade do *software* criado.

Palavras-chave: IA, Rasa, Rule-based

Abstract

Chatbots have been around for over 20 years and are software based on artificial intelligence (AI) that seek to carry out a communication between the machine and the human being using a branch of AI called natural language processing (NLP). The creation of this type of software is justified due to the high demand for questions with objective answers, making this process much more agile and democratic with the use of a bot. In view of this, this work aims to implement a chatbot to answer frequent questions from users of the campus library of the Federal University of Maranhão. For that, the Python programming language and the Rasa framework were used to implement the bot, since the development process is onerous and full of steps. In addition, due to the need for a multidisciplinary team, part of the bot's content was provided by a student from the university's librarianship area. The testing session was performed automatically with analysis algorithms and metrics such as precision and recall, in addition to the confusion matrix to ensure the quality of the software created.

Keywords: IA, Rasa, Rule-based

1. Introdução

1.1. Contextualização

Desde quando se criou o termo inteligência artificial (IA) na década de 50 o mundo tem imaginado essa tecnologia como algo utópico, e essa concepção trouxe inspirações para os filmes e livros de ficção científica da atualidade. Ao longo dos anos a IA tem sido usada em diversas áreas, no campo de processamento de imagens, processamento de linguagem natural, estatística, entre outras. Pensando nos filmes criados com o conceito de que uma IA seria capaz de reproduzir uma mente humana e pensar como um humano, é possível voltar à realidade com uma tecnologia que tem sido amplamente utilizada na última década, os *chatbots*.

Um chatbot, por definição, pode ser descrito como um programa de computador projetado para simular conversação com um humano [Adamopoulou e Moussiades, 2020]. Os chatbots tem se popularizado cada vez mais e a cada ano o número de artigos científicos acerca do tema tem aumentado gradativamente. Aplicações mais utilizadas para tal são de atendimento ao cliente, que com a ajuda deles tem sido capaz de diminuir a carga por assuntos simples que ocupam especialistas nos serviços prestados por grandes empresas e instituições [Valentina et al, 2022].

1.2. Justificativa

O uso de chatbots para propósitos educacionais teve um grande avanço nas últimas décadas, sendo um dos marcos desse percurso dado pelo *bot* SmarterChild no ano de 2001. Tal projeto escreveu sua marca como sendo a primeira vez que um chatbot ajudou as pessoas em seu dia a dia. O *bot* era capaz de buscar e retornar informações da internet sobre filmes, livros e outros, sendo uma ideia única para a época, pois até então não existia nada parecido com isso. Ele ficou disponível em plataformas de mensageria populares como o American Online (AOL) e Microsoft (MSN Messenger). [Molnár e Szüts, 2018]

O SmarterChild serviu de inspiração para criação de outros *bots* ao longo dos anos, e trouxe influência para a criação de novas ferramentas que facilitam a criação de um *chatbot* [Molnár e Szüts, 2018]. O presente trabalho surgiu em parceria com a área de Biblioteconomia da Universidade Federal do Maranhão, que apontou que muitas pessoas tinham dúvidas sobre a Lei de Direito Autoral, que é a lei que regulamenta registro de patentes. Utilizar um *chatbot* capaz de responder a algumas das perguntas mais frequentes torna o acesso à informação mais fácil aos usuários.

A lei 9610 de fevereiro de 1998 é um material que, em sua forma original, pode ser de difícil entendimento ou esclarecimento sobre alguma dúvida mais objetiva. Muitas pessoas optam por consultar um especialista na área, o que pode ser custoso devido a demanda. Para isso defende-se a ideia da criação de um *chatbot* que seja capaz de auxiliar e democratizar o acesso a esse tipo de informação.

1.3. Objetivos

Esse projeto tem como objetivo abranger a biblioteca universitária no campus da Universidade Federal do Maranhão, servindo de auxílio para os usuários que buscam por uma resposta objetiva e comumente feita sobre a Lei de Direito Autoral, reduzindo assim a carga de perguntas feitas aos especialistas no assunto.

Como objetivos específicos têm-se abranger canais de comunicação comuns à universidade possibilitando fácil acesso e integração e permitindo a rápida adição de conteúdos, tendo em vista o surgimento de novas questões e mudanças significativas na legislação sobre o assunto.

1.4. Organização

O presente documento apresentará nos tópicos seguintes os referenciais teóricos para a criação de um *chatbot*, onde serão detalhados os conceitos básicos sobre os principais tópicos exigidos para o seu entendimento e desenvolvimento, o que inclui uma descrição geral, detalhes sobre o modelo específico a ser utilizado, e alguns termos relevantes para complementar a ideia ao redor do tema.

Após descrever os conceitos gerais é necessário descrever a parte específica, abrangendo o *framework* que foi escolhido para o desenvolvimento desse *chatbot*, e como funciona o desenvolvimento dos componentes que vão ser utilizados para configurá-lo para o propósito específico deste trabalho. O tópico de trabalhos relacionados, apresentará um escopo geral de como andam os projetos na área e como têm sido os resultados obtidos. A partir desses conhecimentos é possível seguir para a etapa de implementação.

A etapa de implementação é a descrição sobre como foi o processo de desenvolvimento do *chatbot* utilizando a tecnologia escolhida e pontuada no referencial teórico, onde serão apresentados alguns requisitos funcionais e não funcionais e a configuração para o *chatbot*. As perguntas selecionadas que farão parte do fluxo também serão apresentadas neste tópico.

Após implementar, algumas mudanças podem surgir após avaliar alguns pontos. O próximo tópico, experimentos, foi criado para desenvolver acerca dos ajustes e escolhas utilizadas para avaliação do chatbot e dificuldades enfrentadas e adaptações que precisaram ser feitas para melhor performance do chatbot. Nesse

tópico serão descritos os resultados obtidos, que serão complementados no último tópico.

2. Referencial teórico

2.1 Conceitos básicos

2.1.1 Chatbot

Chatbot é o termo que se deriva de uma expressão para o que em sua tradução significa robôs de conversação, que nada mais é do que um programa que pode ser executado em algum meio, para interação com humanos. Em sua essência, os chatbots devem ser próximos da experiência de conversação interpessoal, apesar de ser feita através de um algoritmo. [Adamopoulou e Moussiades, 2020]

Naturalmente o chatbot é desenvolvido seguindo um fluxo de conversação que procura atender ou entreter (caso seja esse o propósito) o usuário, tendo início a partir de uma saudação que pode ser desde um simples “oi” até um “opa, tudo bem?”, e cabe ao algoritmo identificar o que seria uma saudação e iniciar as respostas de acordo com o previsto em cada fluxo [Höhn e Bongard-Blanchy, 2021].

Diversas empresas têm adotado os chatbots como solução para atender seus clientes e usuários, uma vez que, conforme o crescimento da companhia ocorre se torna mais difícil atender a todas as solicitações de forma personalizada (através de humanos). Exemplos disso atualmente são: Statsbot (Beleza e Moda), Poncho (Meteorologia), Homehawkai (Imóveis), HealthTap (Saúde), entre diversos outros aplicados nas mais diversas áreas [Thorat e Jadhav, 2020].

Durante a pandemia de Covid-19, foram desenvolvidos chatbots por algumas entidades governamentais a fim de atender as dúvidas do público em geral, pois havia novas informações chegando a todo momento e com a procura por dúvidas se tornou inviável atender a todas as perguntas. A partir de um fluxo era possível de forma objetiva sanar algumas das perguntas mais frequentes e alguns extras que chegaram a ser desenvolvidos não só pelo Brasil, mas por outras entidades globais. [Höhn e Bongard-Blanchy, 2021]

2.1.2 Rule-Based Chatbot

O design de um chatbot é moldado a partir de como ele vai lidar com as informações que serão recebidas do usuário. Um chatbot que utiliza a metodologia de regras é feita através da identificação de padrões predeterminados. O contexto também pode compor a regra prescrita para o chatbot, uma das regras aplicadas

poderia se referir ao uso da interrogação em alguma posição da entrada do usuário como indicativo de uma pergunta.

Normalmente trabalhar com esse tipo de abordagem traz algumas desvantagens, uma delas é o fato de, devido a o desenvolvedor ter que escrever uma extensa tabela de regras e fluxos, fica mais difícil de corrigir possíveis erros na entrada do usuário e variações linguísticas da mesma forma de expressão, a exemplo de um identificador do tipo greetings poderia ser tanto “oi”, “ola”, “oiê”, “alô”, entre outras mais completas. [Thorat e Jadhav, 2020]

Para entender melhor o modelo serão apresentados alguns exemplos de lógica utilizados para essa abordagem. O primeiro deles é através de caracteres em expressões regulares: a seguinte expressão con[sc]en[sc]us será capaz de reconhecer palavras que possam ter sido escritas de forma incorreta por questões ortográficas, porém representam a mesma sonoridade, então as palavras “consensus”, “concensus”, “consencus” e “concencus” seriam reconhecidas através de uma expressão regular simples [RASA, 2022].

Outra abordagem comum utilizando ainda expressões regulares é o uso de quantificadores para marcar uma letra que possa surgir por erro de digitação, letras próximas no teclado ou ortografia. No exemplo, “humou?r” é uma expressão regular que representaria a possibilidade de a letra “u” vir adicionalmente na palavra e que seria reconhecida como humor da mesma maneira [RASA, 2022].

Chatbots do tipo rule based seguem muito de regras feitas através de expressões regulares, porém não ficam exclusivamente amarrados a esse tipo de abordagem, podendo serem complementados fazendo uso de NLU, ou Entendimento de Linguagem Natural, que permitirá uma amplitude maior no campo de reconhecimento de palavras e significados e será abordado de forma mais aprofundada no próximo tópico [Thorat e Jadhav, 2020].

2.1.3 NLU - Natural Language Understanding

O processo de desenvolvimento de um chatbot vem acompanhado de um requisito, que se trata do Natural Language Understanding (NLU). O NLU é um ramo da Inteligência Artificial que lida com o entendimento de linguagem humana, esse entendimento é feito identificando a intenção do usuário com uma determinada entrada e aplicar uma resposta adequada ao que o usuário deseja, um exemplo disso é: o usuário deseja saber se o deve ir a praia amanhã, então pergunta ao bot, “o clima estará bom amanhã?”, o entendimento para essa pergunta deve ser “qual a previsão do tempo para amanhã?” tendo uma ação de buscar a resposta adequada. [Adamopoulou e Moussiades, 2020]

A linguagem é algo fundamental para os humanos, porém existem variações que não são tão lógicas para que a máquina possa compreender de forma clara e objetiva, o que nos leva ao desenvolvimento do Processamento de Linguagem Natural (NLP).

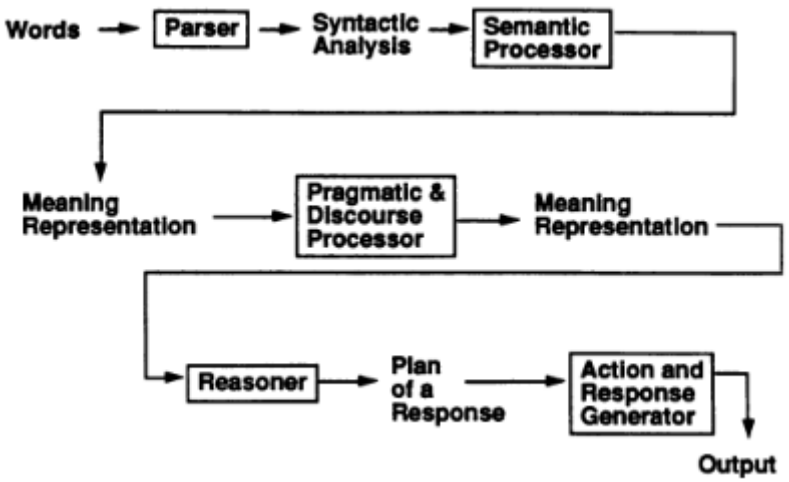

Figura 1 - Fluxo genérico de um sistema NL [Bates, 1993]

O fluxo apresentado na figura 1 descreve de forma genérica o processo de processamento e entendimento de linguagem natural, onde as etapas marcadas com borda representam as etapas pertinentes ao chatbot, e as etapas sem marcação são saídas de e entradas de próximas etapas após algum tipo de processamento [Bates, 1993]. Partindo da primeira entrada que é dada pelo usuário através de uma ou mais palavras, segue então para um parser que vai separar as palavras conforme significado e relevância (ex: João é uma palavra que representa um nome), o resultado disso é uma análise sintática como estamos habituados na língua portuguesa e também nas demais linguagens.

O próximo passo é a análise semântica que vai analisar o conjunto de palavras para encontrar um sentido na frase ou palavra que foi inserida inicialmente, essa etapa é mais complexa, portanto passa por dois blocos lógicos até chegar a uma representação de significado mais acurado. Após ter o significado encontrado, o bloco chamado de reasoner vai realizar um raciocínio semântico analisando o conjunto de regras prescritas para identificar qual ação o usuário deseja disparar gerando o plano de resposta, que já é um direcionamento para a ação que vai gerar uma saída definitiva no último bloco lógico [Bates, 1993].

A partir da base teórica de NLU podemos partir para uma forma mais concreta e partir para a aplicação disso em um chatbot utilizando uma linguagem orientada a chatbots, que varia de acordo com a ferramenta que está sendo utilizada, a exemplo do Rasa Open Source que se utiliza do Conversation-Driven Development, sendo uma forma de escrever uma lógica para o entendimento da linguagem usando uma sintaxe moderna, direcionada diretamente para o Rasa e sua estrutura [RASA, 2022].

2.1.4 Conversation-Driven Development (CDD)

O conceito de CDD foi inicialmente apresentado pela equipe do Rasa, e foi desenvolvido e pensado com a ajuda do engajamento da sua extensa comunidade que faz uso da ferramenta e tem participado ativamente em fóruns e contribuições ao código do projeto até chegar a versão que se tem hoje (versão 3)[Santos et al, 2022].

A utilização dessa abordagem é importante para permitir que o chatbot se aprimore com o passar do tempo. Segundo a documentação do Rasa, o CDD tem as requer as seguintes ações [RASA, 2022]:

- Compartilhar - Entregar o chatbot a diversos usuários o quanto antes.
- Revisar - Rever as conversas regularmente
- Anotar - Melhorar o NLU com base em conversas reais
- Testar - Sempre realizar testes no chatbot antes de entregá-los e após cada atualização
- Rastrear - Identificar conversas que obtiveram bons resultados e também as que não tiveram
- Arrumar - Fazer ajustes para corrigir como seu chatbot lidou com conversas que não obtiveram bons resultados

O que esses princípios trazem, é a ideia de que o CDD é deixar com que as conversas reais guiem o desenvolvimento do chatbot para construí-lo com excelência. Para isso são necessários dois pontos: Reunir dados e compartilhar com usuários para teste o quanto antes, assim os dados ficam mais acurados com o passar dos testes seguindo os tópicos mencionados acima.

2.1.4 Artificial Intelligence Markup Language (AIML)

AIML é uma das ferramentas utilizadas em chatbots do tipo rule-based, sendo baseado em XML, com essa linguagem é possível criar certos padrões de reconhecimento dentro das tags (algo semelhante a uma expressão regular) [Santos et al, 2022]. Apesar de utilizar uma estrutura que já é bastante antiga (XML), o AIML é amplamente utilizado no contexto de chatbots.

Essa linguagem herda muitas características em comum com o XML, fazendo as adequações necessárias para o propósito que a linguagem tem. Sendo assim, é possível denotar as tags pais como sendo a “categoria” que é um conjunto de “padrão” e “modelos”, assim como é mostrado na figura abaixo. [Adamopoulou e Moussiades, 2020]

```
<category>
  <pattern>MYNAMEIS * </pattern>
  <template>Nice to meet you <set name="nameUser"><star/></set></template>
</category>
```

Figura X - Exemplo de AIML [Adamopoulou e Moussiades, 2020]

No exemplo é possível notar que os padrões se utilizam de símbolos para representar palavras de interesse e contextualização, acima é possível ver o “*” asterisco como indicativo de variável para o nome do usuário que vem após a frase “meu nome é *”.

2.2. Rasa Open Source

Rasa é uma biblioteca muito popular entre os desenvolvedores para criação de assistentes virtuais, automatizando a interação entre humanos e máquinas. Uma das grandes vantagens de se trabalhar com um projeto Open Source se dá devido ao código aberto, o que na prática possibilita com que qualquer pessoa possa utilizá-lo de forma gratuita, além de ter colaboração de desenvolvedores que contribuem para que o código do projeto fique cada vez melhor [Santos et al, 2022].

A escolha de uma biblioteca para desenvolver qualquer tipo de trabalho acadêmico ou profissional deve vir acompanhada de vários fundamentos que justifiquem a sua escolha. Nesses quesitos vale apontar algumas das características do projeto e comparar a outras opções que poderiam ter sido utilizadas. Inicialmente filtrado por linguagem de programação, o Python, que é uma linguagem comumente utilizada quando se trata de inteligência artificial [Maroengsit et al, 2019].

Entre as ferramentas disponíveis para o desenvolvimento de um chatbot, apesar da escolha da linguagem, é possível destacar o Dialogflow, o Microsoft Bot Framework e o IBM Watson. Essas são algumas das mais robustas ferramentas para a construção de chatbots e assistentes, porque já trazem o pacote de inteligência artificial que as empresas por trás das ferramentas já vêm desenvolvendo durante anos (Google, Microsoft e IBM). E se mostraram muito eficazes através de seus assistentes virtuais (Google Assist, Cortana e IBM Watson Assistant). [Santos et al, 2022].

De forma geral a escolha de qualquer ferramenta seria uma escolha válida para esse projeto, porém é preciso levar em consideração não apenas o processo de desenvolvimento, mas também o contexto no qual esse projeto se insere. Para a aplicação de um chatbot no contexto universitário, ferramentas que não possuem código aberto são uma barreira para o aprendizado de modo geral, pois as coisas funcionam através de uma caixa preta. Outro fator é relacionado às funcionalidades, que apesar de serem ferramentas muito robustas, fazer uso delas é algo oneroso e que para o contexto em que se insere, seria uma barreira no processo como um todo.

O Rasa representa um meio termo ideal para o desenvolvimento desse projeto, através de uma licença open source, possui as ferramentas necessárias para o desenvolvimento de um chatbot baseado em regras que deve ser capaz de atender ao propósito de responder a questões relacionadas a uma base de conhecimento sobre Leis de Direito Autoral.

Integrações serão de extrema importância para o propósito que buscamos atingir, o Rasa proporciona integrações com: Sites, Facebook Messenger, Slack, Telegram, Hangouts, etc. O que é mais do que suficiente para que possamos implementar em praticamente qualquer ambiente acadêmico [Rasa, 2022].

A curva de aprendizado é algo que tem que ser levado em consideração, muito devido ao tempo investido e o tempo disponível para realização do projeto, logo uma ferramenta simples nem sempre pode proporcionar o que estamos buscando atingir, porém algo complexo exige mais tempo, além de tornar passível de erros no decorrer. O Rasa é uma ferramenta complexa composta de vários módulos, mas para isso conta com uma comunidade ampla, o que torna mais acessível o conhecimento, uma vez que é compartilhado de forma franca [Rasa, 2022].

Apesar de ser um projeto open source, o Rasa também conta com soluções empresariais para projetos mais robustos, com confiança de empresas como a Adobe e Toyota, aumentando a segurança na escolha dessa biblioteca. A arquitetura mostra a robustez do projeto como um todo e sua capacidade na geração de chatbots inteligentes [Rasa, 2022].

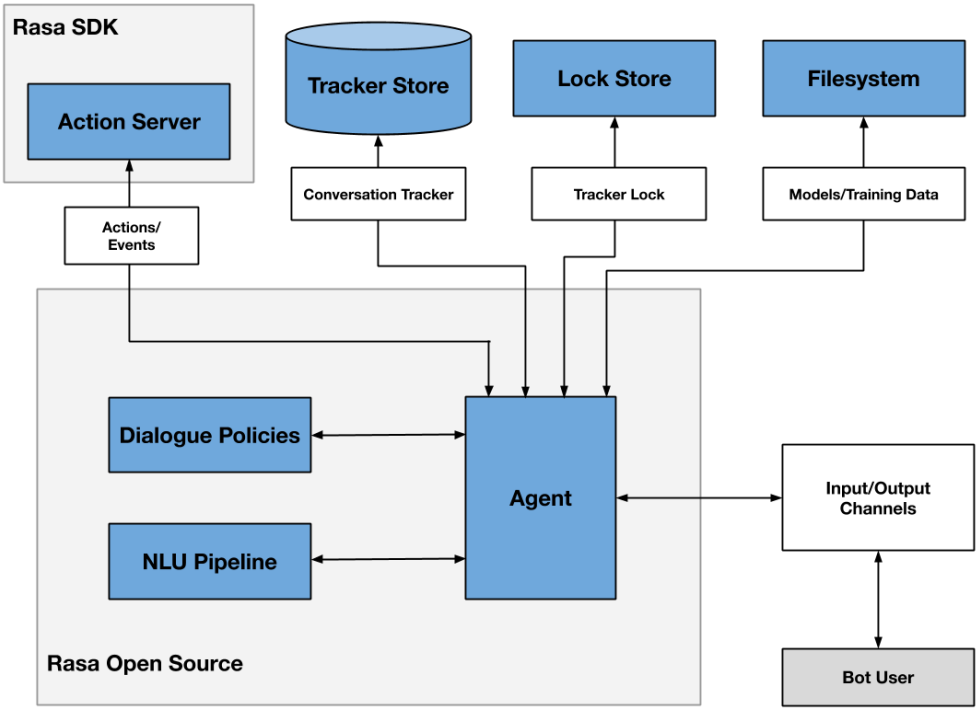

Figura 2 - Arquitetura Rasa [Rasa, 2022]

A figura 2 mostra uma visão geral sobre a arquitetura do Rasa. O principal bloco nesse diagrama é o Agente que é responsável por fazer o intermédio entre as outras partes e com o usuário de forma quase que direta. O Rasa SDK é um pacote disponibilizado em python para criar um servidor de ação que fará a interpretação de eventos e tomará uma ação para determinado evento, tudo que for implementado, o core de código é o Rasa SDK. O bloco de track store é um sistema de armazenamento de conversas que forem sendo feitas usando o chatbot e pode

ser implementado utilizando ferramentas como SQL, MongoDB ou Postgres. O Lockstore trata sobre a gerência de usuários entre conversas, pois é comum ter conversas sendo executadas simultaneamente, então é necessário este bloco para manter a lógica de cada usuário em sua respectiva conversa. O Filesystem, ou sistema de arquivos é apenas um local em disco onde ficam armazenados o modelo treinado para o chatbot e para onde serão enviadas as informações sobre o treinamento. Indo para dentro do core do Rasa temos o Dialog Polices e NLU Pipelines que são responsáveis por definir quais blocos entrarão em ação durante o diálogo, e qual fluxo será seguido durante o treinamento do modelo respectivamente. Os blocos mais relevantes serão explicados de forma detalhada ao decorrer deste tópico [Rasa, 2022].

2.2.1. NLU Pipelines

Inicialmente é preciso entender o conceito de pipeline, é uma palavra de origem inglesa que significa basicamente “cano”, uma sequência de canos em linha, entretanto o conceito se popularizou em várias áreas, na administração quando falamos de etapas de um processo. No caso do Rasa esse conceito atrelado ao processo de NLU significa também um conjunto de etapas que são previstas para o bot no tratamento das mensagens que são imputadas pelo usuário e também como será configurado o bot em caso de Fallbacks [Rasa, 2022].

O Rasa abstrai boa parte do processo, e é possível definir qual será a pipeline e rearranjar os processos através de um único arquivo, que contém as regras de NLU. Um exemplo disso pode ser visto na figura 3.

```yaml
pipeline:
  - name: WhitespaceTokenizer
  - name: RegexFeaturizer
  - name: LexicalSyntacticFeaturizer
  - name: CountVectorsFeaturizer
  - name: CountVectorsFeaturizer
    analyzer: char_wb
    min_ngram: 1
    max_ngram: 4
  - name: DIETClassifier # Dual Intent Entity Transformer
    epochs: 100
```

Figura 3 - Rasa NLU Pipeline [Autoria própria]

Cada um dos métodos é uma forma de lidar com as mensagens e fazem parte do processo de treinamento do bot. O bot funciona através de “intents”, ou seja, um conjunto de mensagens podem representar a mesma intenção, ex: oi, olá, boa tarde, etc. Seguindo o conceito da figura 3, inicialmente seria feito uma separação de palavras por espaço, sendo assim uma mensagem poderia ser representada assim: [boa, noite], falando de forma simplificada. O próximo passo

descrito na imagem é o “RegexFeaturizer”, responsável por transformar a mensagem do usuário em um vetor de expressão regular, o motivo disso é que será mais fácil fazer a classificação e comparar usando. O próximo passo do pipeline, devido a sua maior complexidade, precisa ser explicado em detalhes no próximo tópico [Rasa, 2022].

Como dito previamente, o pipeline serve no momento do treinamento do modelo criado, sendo assim, imaginando o seguinte cenário: três componentes, A, B e Last. O treinamento de um modelo vai passar pelos seguintes passos:

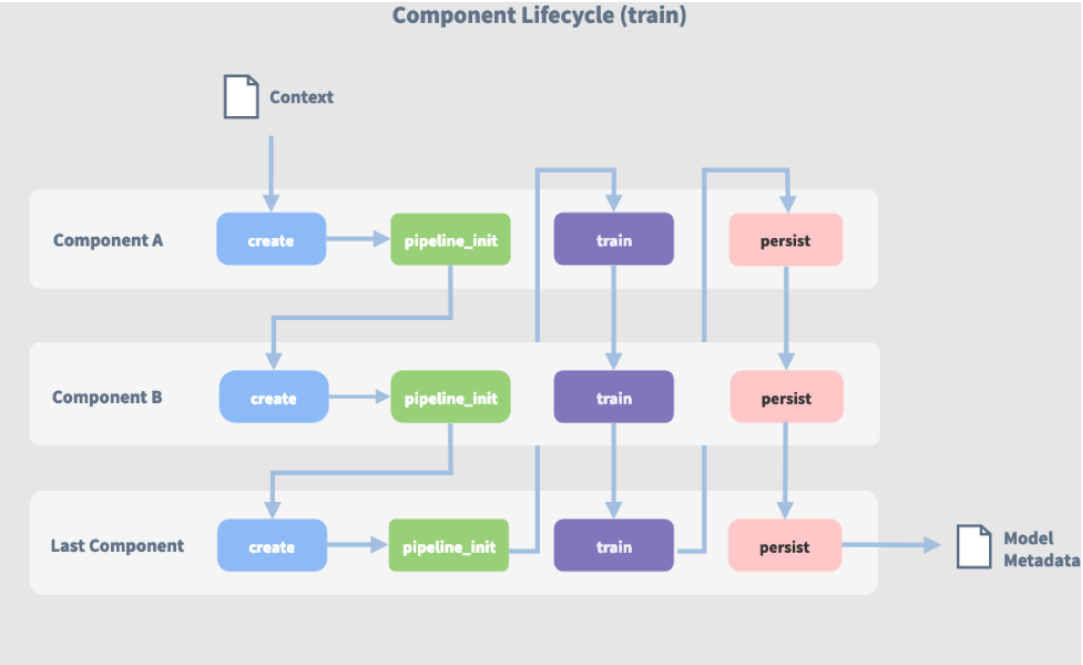

Figura 4 - Ciclo de vida componentes Rasa [Rasa, 2022]

3. Trabalhos relacionados

Os chatbots podem ser encontrados em várias áreas, onde haja necessidade de atendimento a pessoas ou prestação de serviços. Segundo Kuyven et al, em sua revisão de chatbots para saúde fornece um dado relevante para o projeto, onde a maioria dos chatbots foca na área do conhecimento da Ciência da Computação e são destinados para o ensino superior. Isso se deve ao fato de que há fatores a serem aprimorados no reconhecimento de linguagem, dada a variedade de formas de linguagem e suas demais variações por região, idade e principalmente nível de escolaridade [Kuyven et al, 2018].

Existem algumas instituições que vinham se destacando no desenvolvimento de chatbots até o ano de publicação do estudo referido, com presença forte entre os artigos estudados por instituições nacionais com destaque para as universidades e institutos federais. Os trabalhos relacionados dão uma perspectiva prática dos problemas enfrentados e objetivos alcançados, Torath e Jadhav em sua revisão

apontam alguns dos caminhos e possíveis pontos negativos na implementação de um chatbot. Sobre a implementação, existem dois caminhos que podem ser escolhidos para o desenvolvimento: [Torath e Jadhav, 2020]

- Frameworks

A utilização de frameworks agiliza o processo, uma vez que os métodos, funções e classes pré definidos, não é necessário trabalhar diretamente com os métodos matemáticos de análise e classificação para gerar algo do zero. Além disso, há métodos para fazer análise e armazenamento que não precisam ser implementados do zero, alguns exemplos de frameworks são Google dialogflow, IBM Watson, Rasa NLU, Api.ai e outros.

- Plataformas

São normalmente online, um sistema com preparo para interação e deploy do chatbot com um código base próprio onde é possível criar as regras baseadas em fluxo através de low code ou sem code, com blocos arrasta e solta, de forma que se simplifique o processo de implementação e amarre o sistema a plataforma. Essa abordagem costuma ser mais cara, pois as plataformas costumam ter um custo que varia de acordo com o uso do bot. Alguns exemplos de plataforma são: Chattypeople, Botsify, Blip, entre outros.

Em uma releitura de artigos relacionados ao chatbots voltados para a educação, Kuyven e autores mapearam os objetivos e principais resultados alcançados por cada um deles, com base nessa tabela é possível observar como vem sendo avaliados os chatbots criados para essa finalidade.

Tabela 1 - Chatbots, objetivos e principais resultados

ID	Autores	Objetivo	Principais resultados
1	Okita (2014)	Comparação entre auto aprendizado com e sem chatbot.	Os alunos que utilizaram o chatbot obtiveram melhores resultados na aprendizagem
2	Latham et al. (2012)	Proposta de tutor inteligente que se adapta ao estilo de aprendizagem do estudante.	Capacidade de predição do estilo de aprendizagem com acurácia superior a 61%.
3	Jia et al. (2012)	Avaliação individualizada	Melhoria significativa nas habilidades de

		do aprendizado de inglês como segunda língua.	compreensão, pronúncia, escrita e vocabulário.
4	Tegos, Demetriadis e Karakostas (2015)	Impacto do chatbot em atividades colaborativas entre alunos.	Maior engajamento dos alunos em diálogos mais produtivos; argumentação explícita nas atividades colaborativas; melhoria na aprendizagem com a intervenção do agente na argumentação entre os alunos.
5	Latham, Crockett e McLean (2014)	Proposta de sistema para adaptação ao estilo de aprendizagem do estudante.	Melhores resultados no desempenho dos alunos que utilizaram o sistema de tutoria com chatbot conforme seu estilo de aprendizagem.
6	Schouten et al. (2018)	Aplicação de técnicas afetivas em um tutor digital para alunos com baixa alfabetização.	Verificação da importância da personalização do chatbot para a aprendizagem
7	Dyke et al. (2013)	Uso de chatbots na mediação do aprendizado colaborativo.	Resultados significativos no aprendizado entre alunos que foram mediados pelo chatbot.
8	Aguiar, Taro uco e Reategui (2011)	Apoio aos estudantes em um sistema de aprendizagem autorregulada.	Aprimoramento de habilidades cognitivas e construção do conhecimento, porém sem aumento do engajamento
9	Moreno et al. (2015)	Proposta de chatbot para uso na área de linguística- Atlas Brasileiro.	Vantagem da utilização de chatbot através de aplicativos populares de mensagem instantânea. Aceitação positiva por parte dos alunos
10	Oliveira et al. (2010)	Proposta de chatbot com intenção e personalidade, usando ontologias, para	Aceitação positiva pelos alunos no apoio ao ensino e aprendizagem. Arquitetura escalável do agente possibilita conversação

		apoio ao ensino e aprendizagem em psiquiatria e psicologia.	com maior coerência através do uso de ontologias.
11	Lemos, Campos e Nunes (2012)	Apresentar a análise comportamental de um chatbot desenvolvido para ensinar algoritmos.	Identificação de fatores que aprimoram a consistência do diálogo do chatbot: linguagem informal, expressar emoções, identificar redundâncias de linguagem e possuir aspectos humanos para ser considerado consistente.
12	Shopf e Duarte (2005)	Proposta de um chatbot que emprega linguagem regional.	Demonstrou ganho cognitivo do aluno de forma significativa, motivação por parte do aluno pelo uso de linguagem bem humorada do chatbot.
13	Paschoal, Chicon e Falkembach (2016)	Proposta de chatbot integrado ao AVA Moodle, integrando conceitos de aprendizagem ubíqua.	Adaptação do chatbot ao nível de conhecimento e desempenho do estudante, e a diferentes tipos de dispositivos computacionais
14	Leonhardt et al. (2003)	Proposta de chatbot para ensino nas áreas de física e redes de computadores.	Participação mais ativa dos estudantes e foco do diálogo no interlocutor. Observação da necessidade de muitas interações pelos alunos, e consequente análise dos registros desta utilização para a criação de novas categorias, tornando a conversação mais natural e abrangente
15	Paschoal, Chicon e Falkembach (2017)	Proposta de chatbot integrado a ambiente virtual de aprendizagem.	Contribuição para a aprendizagem dos alunos. Identificação de fatores por parte dos alunos que deveriam ser aprimorados, como ampliação da base de conhecimento e

			tratamento de algumas sentenças de interação.
16	Gomes, Barbosa e Geyer (2005)	Proposta de agente pedagógico integrado a ambiente virtual de aprendizagem.	Contribuição do chatbot para tornar o ambiente virtual mais interativo e participativo.

Fonte: Kuyven et al, 2018

A tabela 1 apresentou resultados muito positivos por parte dos projetos, que aumentaram o engajamento entre o público que utilizou, melhorando o desempenho para o caso de alunos que fizeram uso do chatbot tendo uma contribuição grande na educação. Tendo em vista esses resultados, espera-se obter resultados futuros positivos através deste projeto, que terá o papel de aumentar a produtividade no processo de ensino por meio da quitação de dúvidas sobre a lei de direito autoral, que costuma ser palco de perguntas frequentes.

4. Implementação

4.1 Planejamento

Criar um projeto de chatbot exige um bom planejamento e organização sobre as fases de desenvolvimento. A divisão do tempo gasto em cada etapa deve contar com imprevistos que podem ocorrer durante o processo. Ficando estruturado da seguinte forma:

4.4.1 Escolha da tecnologia e levantamento de requisitos

Essa etapa é a inicial no processo de desenvolvimento, pois servirá de base para consultas futuras e definirá qual tecnologia será usada, isso acaba trazendo os pontos negativos e positivos da escolha, portanto este demanda uma quantidade de pesquisa extensa.

4.4.2 Desenvolvimento e testes

A parte mais demorada é a etapa de desenvolvimento, pois está suscetível a problemas que podem surgir conforme a geração de código ocorre. É necessário que a etapa anterior seja o mais precisa possível para evitar que algum requisito não seja implementado devido a alguma limitação da tecnologia escolhida para a criação do bot.

A etapa de testes pode gerar um retrabalho de código ao verificar que o comportamento não esteja conforme esperado, remetendo ao ciclo de vida cíclico. Após o comportamento estar adequado é possível gerar métricas que asseguram a qualidade do software desenvolvido.

4.4.3 Avaliação de resultados e documentação

Documentar um software é garantir a sua longevidade, pois será mais fácil de fazer manutenções e futuramente para novos desenvolvedores que façam uso do projeto que possam entender facilmente o que foi criado e poder dar continuidade. Essa etapa é a última etapa e terá esse projeto como descrição do que foi realizado para a criação e configuração do chatbot.

A avaliação também é importante para garantir que o bot seja colocado em prática apenas se os requisitos foram satisfatórios e as métricas coletadas sejam altas para submeter o chatbot a etapa que não será coberta por esse projeto, que é a validação, com usuários finais.

4.2 Requisitos

4.2.1 Funcionais

Os requisitos funcionais são todos aqueles pensados no ambiente de desenvolvimento e hospedagem necessários para comportar o sistema. Podendo ser listados da seguinte forma:

- Windows ou Linux
- Docker
- Python 3
- Pip instalador de pacotes
- Rasa 3
- Venv

O sistema foi desenvolvido em windows utilizando o visual studio code em um ambiente windows. Após desenvolvimento e teste local, um componente Docker foi criado para que o deploy fosse viável em qualquer sistema de hospedagem em nuvem para exposição do serviço.

4.2.2 Não Funcionais

Os requisitos não funcionais dizem respeito àqueles que especificam sobre o que o sistema deve fazer de forma a não especificar o como será feito em questão de hardware ou software. Os requisitos não funcionais são os seguintes:

- Ser capaz de responder às perguntas pré cadastradas com possibilidade de variações
- Possuir interface de comunicação com várias plataformas
- Responder adequadamente a questões não existentes
- Possuir perguntas comuns fora do próprio contexto

4.3 Rasa

4.3.1 Configurações

O Rasa é composto por alguns arquivos .yml que definem as suas configurações e as definições que serão usadas no treinamento do modelo e mais a frente será o tratamento da mensagem para gerar um grau de certeza.

O Rasa dispõe de alguns modelos de chatbot, para o propósito do projeto de um chatbot Rule Based a configuração foi a de “RulePolicy”, que define que para cada input do bot haja uma regra específica de resposta já prevista. Sendo assim uma regra dentro do bot é algo que segue o seguinte formato:

```
- rule: nome_da_regra
steps:
  - intent: intenção_identificada
  - action: ação_de_esposta
```

Figura 5 - Regra do chatbot em yaml [Próprio autor. 2022]

O formato acima pode descrever por exemplo uma intenção de saudação, onde o usuário digitou um texto que teve a intenção de saudar o bot, com mensagens do tipo “oi”, “olá” ou “bom dia”. Logo abaixo da intenção referida, há uma ação prevista para quando for identificada essa intenção, que pode ser uma resposta em foto, um link, ou simplesmente um texto.

Idealmente o bot desenvolvido deve responder a esse tipo de intenção cordialmente respondendo e dando uma breve introdução ao escopo no qual está inserido, para que a próxima interação do usuário seja uma pergunta direcionada ao assunto de interesse.

Foram definidas 4 intenções ao bot, a primeira se refere a iniciação de uma conversa, chamada de “greet”, compondo frases como: “olá”, “boa tarde”, “ei”, entre

outros similares. A próxima intenção diz respeito à questão propriamente dita, chamada de “faq”, onde foram preenchidas as perguntas desejáveis e mais comuns para um bot específico para o tópico lei de direito autoral. A seguinte é semelhante ao citado anteriormente, que será utilizada para lidar com assuntos diversos e perguntas fora do contexto do bot, nomeada de “chitchat”. Por fim tem-se a intenção “init” que é utilizada para responder a primeira interação do usuário quando a mesma é feita por um canal como Telegram ou Slack.

Ainda inserido no arquivo de configuração, o pipeline ficou definido da seguinte forma:
Whitespace Tokenizer → Regex Featurizer → Lexical Syntactic Featurizer → Count Vectors Featurizer → Diet Classifier → Entity Synonym Mapper → Response Selector (chitchat) → Fallback Classifier. Seguindo um padrão para esse tipo de bot e recomendado pela própria documentação oficial, o começo da pipeline segue os mesmos princípios aplicados ao chatbot de exemplo. Algumas etapas a mais poderia melhorar a performance, porém isso aumentaria o tamanho do modelo e o tempo de cada treinamento, tornando o desenvolvimento mais demorado. Cada linguagem possui seus próprios sinônimos, ou abreviações que remetem a uma mesma palavra, uma etapa que ficou fora do pipeline foi o Spacy NLP, que contém um conjunto pré treinado de palavras com significado semelhante.

- Whitespace Tokenizer, de modo geral os “tokenizers” criam frações do texto em tokens para a criação do modelo, nesse caso como uma etapa, servindo de entrada para a próxima etapa. Este componente utiliza o “espaço” como um separador na geração de tokens. Além disso, qualquer caractere que não esteja dentro do padrão do seguinte regex: “a-zA-Z0-9_#@&”, será substituído por espaços em branco e então gerado o token, dessa forma os seguintes exemplos se aplicam:

$$"!palavra" \Rightarrow "palavra"$$
$$"palavra!" \Rightarrow "palavra"$$

Além disso a seguinte regra também se aplica: qualquer caractere que não esteja dentro do seguinte regex: “a-zA-Z0-9_#@&.~:\/?[]()!$*+,;=-” será substituído por espaços e então gerados os tokens, exceto se o caractere não estiver entre números. Dessa forma os seguintes exemplos também se aplicam:

$$"vinte\ e\{um" \Rightarrow "vinte", "e", "um"$$
$$"10\{1!" \Rightarrow "10\{1"$$

- Regex Featurizer, de forma simplificada esse componente cria uma representação em forma de vetor da entrada utilizando expressões regulares. Durante o treinamento e criação do modelo, cria expressões regulares que representam a entrada do usuário, dessa forma reduz o espaço ocupado pelo modelo e cria uma forma mais eficiente de comparar strings (programaticamente falando), portanto essa etapa já é contabilizada no

reconhecimento de intents como parte do score final. Algumas configurações podem ser aplicadas para otimizar o reconhecimento e geração de expressões regulares, uma delas é a de ignorar letras maiúsculas no começo das palavras, sendo ativada para o projeto criado, algumas configurações são específicas e podem ser utilizadas quando o projeto é desenvolvido para outras línguas.

- Lexical Syntactical Featurizer, assim como o anterior, os “Feturizers” criam recursos para o reconhecimento de intenções. Utilizando a técnica computacional “Window Sliding” na qual se percorre um loop de forma ordenada, gerando anotações sintáticas para o modelo de acordo com as configurações.
  Esse passo é necessário devido a natureza de como a linguagem humana é feita, especialmente considerando o escopo de multi linguagens. Diferente do que funciona para o computador, onde não há ambiguidades, cada bit é simplesmente 1 ou 0, na linguagem humana, uma palavra como “meia” pode significar uma peça de roupa ou metade, dependendo do contexto em que se insere, e isso não é um exemplo isolado, é possível apontar palavras como letra, boca, manga, banco, entre outros.
  Então é preciso separar os tokens de forma que coisas como a posição, e outros tokens influenciam no que seria o resultado final, evitando que o NLU tenha um entendimento incorreto da intenção do usuário.
  Os algoritmos utilizados para esse fim não necessariamente fazem um entendimento do que cada palavra realmente tem de significado, porque, para um humano é simples distinguir essas colocações: “meia dúzia" e “pé de meia”, mas para o algoritmo interpretar isso é como lidar com um saco de palavras de forma ordenada. Supondo um algoritmo de aproximação de cossenos, imagina-se que a distância entre as frases “cachorro morde gato” e “gato morde cachorro” sejam próximas, pois utilizam exatamente as mesmas palavras, todavia as frases não têm o mesmo sentido.
  Entre as possibilidades de configuração disponíveis existem avaliações para verificar títulos (palavras que iniciam com a primeira letra em maiúscula e o restante em minúscula), BOS (é uma verificação que diz respeito a posição do token), entre outros. Porém não foi necessário alterar a configuração padrão do componente para esse projeto.

- Count Vectors Feturizer, essa etapa cria uma representação da mensagem em uma espécie de “saco de palavras” utilizando a biblioteca scikitlearn, as configurações desse componente são vinculadas às configurações da biblioteca. Olhando de forma detalhada para o componente, a ideia de “saco de palavras” são apenas representações utilizando n-grams. Essa etapa do processo foi configurada para limitar a criação de no máximo 4-gram, para evitar um grande processamento na criação dos vetores. A própria documentação da função apresenta como exemplo o seguinte resultado:

$array(['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this'], \ldots) =>$
[[0 1 1 1 0 0 1 0 1] [0 2 0 1 0 1 1 0 1] [1 0 0 1 1 0 1 1 1] [0 1 1 1 0 0 1 0 1]]

- DIET Classifier (Dual Intent and Entity Transformer), que em sua tradução literal quer dizer transformador de entidades e intenções duplas, criado pelo Rasa para ser um componente de reconhecimento de entidades e intenções, utilizando multi-tarefas. O DIET é o componente mais complexo mencionado até agora. A arquitetura é compartilhada por ambas as tarefas. Apesar disso, é possível configurar o DIET para realizar apenas uma das tarefas, restringindo tanto a classificação de intenção, quanto a de entidade.

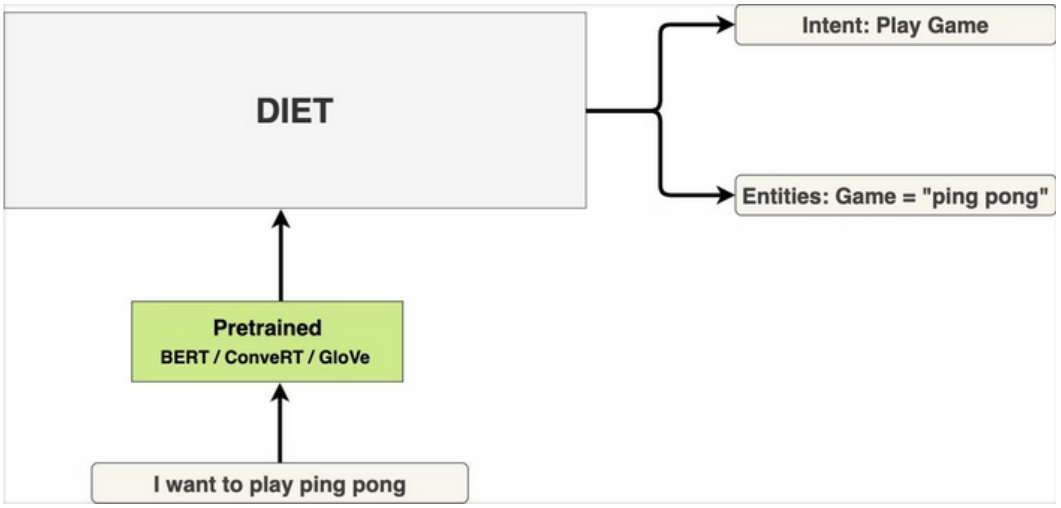

Figura 6 - Dual Intent and Entity Classifier [Mantha, 2020]

A figura 6 apresenta uma ideia visual do objetivo do DIET Classifier, o texto em sua tradução quer dizer “eu quero jogar pingue pongue”, passando pelo componente são reconhecidas a intenção do usuário “jogar” e a entidade “pingue pongue”. Essa arquitetura é muito útil em aplicações que desempenham tarefas para o usuário, como por exemplo um bot bancário que recebe a seguinte entrada: “extrato de janeiro”, entende-se que a intenção do usuário é “tirar extrato bancário” e a entidade é “janeiro”.

O DIET tem depende de no mínimo dois “featurizers” para ter um bom desempenho que tem o papel de converter as mensagens em vetor. Por conta disso o DIET é um dos últimos componentes no pipeline. Foram aplicadas duas configurações para essa etapa: epochs, o número de vezes que o algoritmo verifica o conteúdo de treinamento, uma epoch corresponde a uma vez que o algoritmo percorre todos os exemplos cadastrados, por padrão setado em 300, porém foi configurado para 100, sem que isso afetasse sua performance. A segunda aplicada foi a tag “constrain similarities”, que quando definida em verdadeiro passa a ignorar as similaridades entre os exemplos e extrai apenas das diferenças mais marcantes de cada um.

- Entity Synonym Mapper, esse componente pode mapear sinônimos para o mesmo valor, porém é preciso definir nas informações de treinamento, caso uma expressão como “O que é a Câmara Brasileira do Livro?” fosse cadastrada no bot, seria possível definir que um possível sinônimo para Câmara Brasileira, que pode também ser mencionada como CBL, isso é útil na maioria das vezes para descrever siglas, o que é comum quando falamos de órgãos da organização brasileira ou instituições, é comum ver a Universidade Federal do Maranhão sendo chamada apenas de UFMA.

- Response Selector, esse componente é responsável apenas por coletar os dados das predições após as etapas percorridas e selecionar a que possui um grau de confiança de maior valor. Um detalhe importante de levantar é sobre a flexibilidade deste componente, pois é possível agrupar um conjunto de “sub intenções” dentro de uma intenção maior a fim de selecionar apenas respostas dentro daquele contexto. Essa funcionalidade foi incluída na nova versão do Rasa e é recomendado seu uso para chatbots que tenham como objetivo dar respostas únicas independentemente do contexto anterior. Apesar de se encaixar perfeitamente no contexto desse projeto, a configuração de agrupar foi desativada, pois compromete o próximo componente, que trata de erros.

- FallBack Classifier, essa é a última etapa do pipeline, responsável por classificar a mensagem recebida como um fallback, ou seja, nenhuma das intenções cadastradas foi selecionada com um grau de confiança alto o suficiente para satisfazer a seleção. Um threshold (limiar) é definido nesse componente para determinar o número que define o limite para o grau de confiança mínimo. Não existe um valor default para o threshold desse componente, porém foi definido um valor de 90% de certeza, tendo em vista os testes realizados internamente com pequenas variações das perguntas cadastradas. Adicionar um valor alto ao threshold faz com o que o bot evite dar respostas que não tenha certeza absoluta, porém pode dar uma resposta negativa para uma variação muito grande de perguntas, para solucionar esse problema seria necessário um acervo de exemplos grande para cada uma das perguntas cadastradas com no mínimo 2 exemplos.

4.3.2 F.A.Q

O chatbot foi projetado para responder a perguntas comuns e tirar dúvidas de forma que o usuário inserisse sua dúvida e o bot pudesse reconhecer aquela pergunta ou equivalente forma de inserir a mesma pergunta e responder adequadamente. Inicialmente foram pensadas em 21 perguntas sobre o tema lei de

direito autoral. As perguntas listadas abaixo na tabela 2 (Base de perguntas do bot), são apenas um modelo único para cada pergunta, porém o Rasa torna possível expressá-las de várias formas diferentes em sua configuração.

Tabela 2 - Base de perguntas do bot

PERGUNTAS	**RESPOSTAS**
O que é Propriedade intelectual?	São obras intelectuais protegidas as criações do espírito, expressas por qualquer meio ou fixadas em qualquer suporte, tangível ou intangível, conhecido ou que se invente no futuro. (BRASIL, 1998)
É obrigatório o registro de uma obra	Não. O artigo 7º traz que as obras precisam ser exteriorizadas em qualquer meio ou fixadas em qualquer suporte
O registro é gratuito?	Não. Para ter uma obra registrada, deve-se entregar documentação necessária e pagar as taxas estabelecidas.
Onde posso registrar a obra?	Através do escritório de Direitos Autorais com sede na Biblioteca Nacional. Confira em: https://www.bn.gov.br/servicos/direitos-autorais
Qual o nome do documento expedido que garante a autoria da obra?	Certidão. Este documento é emitido após cumpridas as exigências relacionadas ao registro da obra.
Porque devo registrar uma obra?	É uma segurança jurídica que ajudará a solucionar futuros conflitos judiciais e extrajudiciais relacionados a autoria e exploração da obra
O que é direito moral	"São direitos morais aqueles que dizem respeito a uma ideia de relaçao de paternidade entre o autor e sua obra, e que se justificam por essa ligação pessoal, e não por imperativos de aproveitamento econômico. " VALENTE, Mariana Giorgetti; FREITAS, Bruna Castanheira de. **Manual de direito autoral para museus, arquivos e bibliotecas**. Rio de Janeiro : FGV Editora, 2017. Disponível em: https://bibliotecadigital.fgv.br/dspace/handle/10438/19038. Acesso em: 20 mar. 2022.
O que é direito patrimonial	"Os direitos patrimoniais de autor dizem respeito propriamente à exploração econômica das obras intelectuais e irrenunciáveis." VALENTE, Mariana Giorgetti; FREITAS, Bruna Castanheira de. **Manual de direito autoral para museus, arquivos e bibliotecas**. Rio de Janeiro :

	FGV Editora, 2017. Disponível em: https://bibliotecadigital.fgv.br/dspace/handle/10438/19038. Acesso em: 20 mar. 2022.
O direito moral e o patrimonial podem ser explorados economicamente?	Somente o direito patrimonial pode ser explorado economicamente.
Porque somente o direito patrimonial pode ser cedido temporariamente ou em definitivo?	Porque o direito patrimonial está relacionado com a exploração econômica da obra.
Porque o direito moral não pode ser vendido?	Porque são direitos inalienáveis e irrenunciáveis.
Qual procedimento para registrar uma obra?	Preparar a uma cópia física da obra; Pagar a GRU (Guia de recolhimento da União); Preencher o formulário específico; Protocolar o requerimento e acompanhar a finalização do processo. Para maiores informações consulte: https://www.bn.gov.br/servicos/direitos-autorais
O que é ISBN?	A sigla corresponde ao seguinte termo: International Standard Book Number . Sua tradução corresponde ao Padrão Internacional de Numeração de Livro. É o número que todo livro publicado por uma editora recebe para que seja identificado internacionalmente.
Como e onde devo solicitar o ISBN?	A Câmara Brasileira do Livro - CBL, é a instituição responsável por emitir o ISBN no Brasil. Para mariores informações consulte: https://www.cblservicos.org.br/isbn/
O que é a ficha catalográfica de uma obra?	Elemento pré-textual obrigatório para a publicação de um livro. Contém todas as informações relativas à obra.

Quais as obras intelectuais protegidas pela Lei de Direito Autoral?	os textos de obras literárias, artísticas ou científicas; II - as conferências, alocuções, sermões e outras obras da mesma natureza; III - as obras dramáticas e dramático-musicais; IV - as obras coreográficas e pantomímicas, cuja execução cênica se fixe por escrito ou por outra qualquer forma; V - as composições musicais, tenham ou não letra; VI - as obras audiovisuais, sonorizadas ou não, inclusive as cinematográficas; VII - as obras fotográficas e as produzidas por qualquer processo análogo ao da fotografia; VIII - as obras de desenho, pintura, gravura, escultura, litografia e arte cinética; IX - as ilustrações, cartas geográficas e outras obras da mesma natureza; X - os projetos, esboços e obras plásticas concernentes à geografia, engenharia, topografia, arquitetura, paisagismo, cenografia e ciência; XI - as adaptações, traduções e outras transformações de obras originais, apresentadas como criação intelectual nova; XII - os programas de computador; XIII - as coletâneas ou compilações, antologias, enciclopédias, dicionários, bases de dados e outras obras, que, por sua seleção, organização ou disposição de seu conteúdo, constituam uma criação intelectual. (BRASIL, 1998)
De acordo com a Lei de Direito Autoral, o que não pode ser protegido?	I - as idéias, procedimentos normativos, sistemas, métodos, projetos ou conceitos matemáticos como tais; II - os esquemas, planos ou regras para realizar atos mentais, jogos ou negócios; III - os formulários em branco para serem preenchidos por qualquer tipo de informação, científica ou não, e suas instruções; IV - os textos de tratados ou convenções, leis, decretos, regulamentos, decisões judiciais e demais atos oficiais; V - as informações de uso comum tais como calendários, agendas, cadastros ou legendas; VI - os nomes e títulos isolados;

	VII - o aproveitamento industrial ou comercial das idéias contidas nas obras. (BRASIL, 1998)
O que é uma obra em co-autoria?	Obra em co-autoria é aquela criada em comum, por dois ou mais autores. (BRASIL, 1998)
O que é uma obra anônima?	É uma obra publicada sem a indicação do nome do autor, por sua vontade ou por ser desconhecido. (BRASIL, 1998)
O que é uma obra póstuma?	Uma obra que é publicada após a morte do autor. (BRASIL, 1998)
Quem auxilia o autor com o Prefácio, correção de portugês e normas da ABNT, tradutor ou ilustrador, pode ser considerado co-autor?	Não. O co-autor é a pessoa que participa efetivamente na criação da obra intelectual.

Fonte: Autoria própria

O arquivo do Rasa responsável por conter essas informações é o NLU, nele são inseridas as frases que futuramente serão reconhecidas pela entrada de texto do usuário. A estrutura ainda é em yaml, e organizada da seguinte maneira: a primeira linha chamada intent, identifica qual tipo estamos tratando e a segunda linha em diante são exemplos.

```
- intent: faq_docname
  examples: |
    - Qual o nome do documento expedido que garante a autoria da obra?
    - Que documento garante a autoria da obra?
```

É possível observar no exemplo acima, uma das perguntas descrita na tabela 2, onde fora adicionada uma pergunta extra que possui a mesma resposta e portanto representa a mesma intenção, que no caso acima é descobrir o nome do documento que, quando emitido assegura ao autor a autoria da obra de forma legal. O mesmo modelo segue para todas as perguntas e deve ser melhorada conforme são identificadas novas formas como o usuário pode realizar essa pergunta.

4. Experimentos

Essa seção vai descrever os métodos utilizados para verificar o comportamento do chatbot e os métodos de avaliação utilizados para garantir um bom nível assertivo nas respostas desejadas e no reconhecimento pela NLU criada pelo chatbot desenvolvido.

4.1 Métodos de avaliação

A avaliação de um chatbot pode ser feita por meio de alguns métodos, no entanto alguns demandam um esforço maior, devido a necessidade de uma equipe para desenvolvimento, e usuários ativamente testando. Segundo Wari et al, é possível considerar três maneiras principais de avaliação:

4.1.1 Automática

É possível obter essas métricas de forma automática, utilizando alguns algoritmos para isso, naturalmente exige um processo de desenvolvimento para preparar o bot para responder e reconhecer algumas frases selecionadas ou para que faça isso em tempo de execução. Esse é um dos processos mais rápidos para gerar um resultado, pois não exige um usuário para testar, tudo é feito pela própria inteligência do código baseado nos próprios outputs.

Nesse cenário é importante pontuar duas métricas comumente utilizadas para avaliação de NLP, o ROUGE (Recall-Oriented Understudy for Gisting Evaluation) e o BLEU (Bilingual Evaluation Understudy). O Rouge diferente do BLEU é na verdade um conjunto de métricas mais pontuais que, combinadas geral esse resultado, portanto é essencial discorrer sobre recall, precisão e f1.

- Rouge Recall

Essa métrica vai ser gerada, para a análise de forma automática, para isso é considerado o número de n-grams. O conceito de n-grams se aplica a área de processamento de texto e linguagem natural, e funciona da seguinte forma, considerando a frase “eu sou alguém”:
1-gram: [eu, sou, alguém]
2-gram: [eu sou, eu alguém, sou alguém]

Assim ocorre sucessivamente até o máximo de n-grams possíveis de acordo com a sentença. Dada a definição acima é possível descrever o recall como a contagem de n-grams encontradas no modelo e na referência dividida pelo número de n-grams

de referência: $número\ de\ n\ -\ grams\ no\ modelo\ e\ na\ referência\ /\ n\ -\ grams\ referência.$

Essa métrica é utilizada principalmente para garantir que o modelo gerado está capturando todas informações ou boa parte delas que estão contidas na referência. É apenas necessária uma certa atenção, pois não é possível identificar com essa métrica que o modelo não está gerando uma quantidade excessiva de palavras para alcançar um bom resultado.

- Rouge Precisão

O problema supracitado pode ser contornado utilizando a métrica de precisão, que tem uma fórmula semelhante, porém a divisão leva em consideração o número de n-grams no modelo, ficando da seguinte forma: $número\ de\ n\ -\ grams\ no\ modelo\ e\ na\ referência\ /\ n\ -\ grams\ modelo$

- Rouge F1-Score

Essa métrica exige conhecimento das duas mencionadas anteriormente, pois sua fórmula contém os valores calculados tanto da precisão quanto do recall. A mesma é utilizada para dar maior confiança ao modelo gerado, pois leva em consideração tanto o modelo, capturando o máximo de palavras possível (recall), desconsiderando as palavras sem utilidade (precisão).

A fórmula fica descrita semelhante ao dobro de uma média ponderada entre os dois valores: $2\ *\ [(precisão\ *\ recall)\ /\ (precisão\ +\ recall)]$.

- Rouge-L

O Rouge-L vai medir a subsequência comum mais longa entre o modelo e a referência, ou seja, a sequência de tokens que é compartilhada entre ambos. Essa métrica é utilizada para indicar similaridade entre as sequências, é possível aplicar as métricas citadas anteriormente utilizando o Rouge-L alterando as entradas

- BLEU - Bilingual Evaluation Understudy

O BLEU foi pensado inicialmente para avaliar a tradução entre duas linguagens feita por um sistema textual, porém não necessariamente precisa ser um sistema que faça tratamento de texto em multilinguagens, basta desconsiderar a tradução do texto de referência, pois o BLEU é calculado utilizando. Apesar de não ser uma métrica sem falhas, é muito simples de ser calculada e pode dar um outro ponto de vista sobre como o sistema vai se comportar.

O cenário de cálculo do BLEU é bem simples, considerando as referências R1 e R2, e o nosso modelo M1, são contadas o número de palavras de M1 contidas em R1 e R2 e dividido pelo número de palavras em M1 para obter um valor entre

zero e um. Vale mencionar que essa métrica também se utiliza de n-grams, ou seja, é possível testar o resultado considerando a mesma referência mais de uma vez.

Essa métrica é recente, criada em 2002 por Kishore Papineni, onde o autor diz que o objetivo do BLEU é comparar n-grams entre candidato e referência para contar o número de ocorrências, quanto maior a quantidade de palavras em comum, melhor é o modelo.

- Matriz de confusão

A matriz de confusão é uma tabela utilizada para medir a performance de um algoritmo de classificação, nela é possível visualizar de forma sintética os resultados obtidos pelo modelo. A imagem mostra o exemplo de uma matriz, onde é possível ver 3 classes do modelo, identificadas nas cores amarela, azul e cinza. A matriz vai observar quando duas classes se identificarem ou que o resultado seja adequado e a classificação seja a mesma prevista pelo modelo, o resultado ideal é uma linha (caracterizada por uma matriz identidade), porém no exemplo existem dois erros cometidos, ao identificar a classe amarela como azul uma vez, e a classe azul como cinza, entretanto a maioria das previsões obteve um resultado esperado.

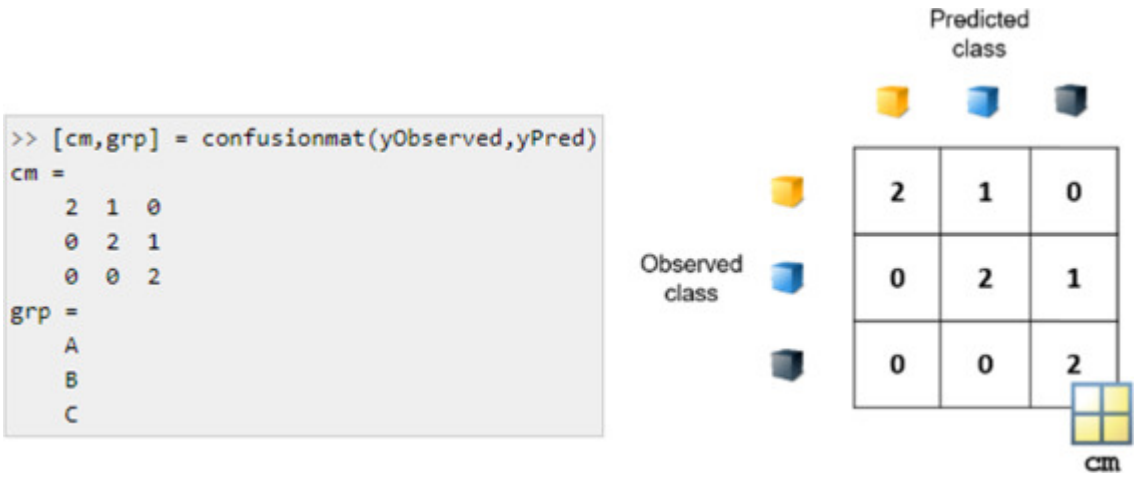

Figura 7 - Matriz de confusão [Singh, 2021]

A visualização da matriz acima (Figura 7) pode ser de difícil visualização quando se trata de mais classes, portanto é possível ativar o mapa de calor para avaliar onde foi causado algum erro de reconhecimento (miss match). A tabela fica colorida onde há destaques para um maior número de predições, observe a figura 8 a seguir, que será o modelo escolhido para analisar o chatbot criado.

```
>> [cm,grp] = confusionmat(yObserved,yPred)
cm =
     2     1     0
     0     2     1
     0     0     2
grp =
     A
     B
     C
>> heatmap(grp,grp,cm,...
    'Colormap','summer')
```

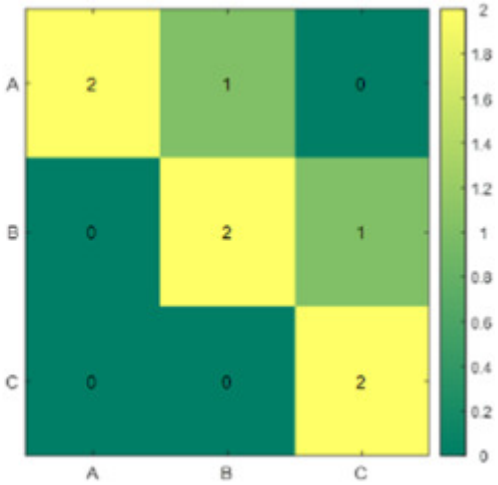

Figura 8 - Matriz de confusão mapa de calor [Singh, 2021]

4.1.2 Manual (Expert)

Essa forma de avaliação de um chatbot exige que um especialista faça os procedimentos necessários para a avaliação do desempenho do bot. Não existe um método específico para uma avaliação manual, Adampou menciona que os projetos que utilizaram dessa abordagem, fizeram em conjunto com a avaliação automática, pois aliada a isso um humano poderia dizer que a resposta era adequada, inadequada ou neutra ao que foi solicitado, afinal, um especialista sabe exatamente qual resposta esperar do chatbot e entender uma variação da mesma resposta como certa. Sendo assim uma métrica resultando dessa avaliação poderia ser dada pela expressão: $(Nadequadas + Nneutras)/Ntotal$. Sendo algo próximo a uma métrica de precisão.

É interessante que esse teste seja feito por um especialista que não fez parte do processo de desenvolvimento, pois o teste poderia ser tendencioso, visto que há um conhecimento sobre as perguntas e respostas contidas no software, portanto esse método não será utilizado nesse projeto.

4.1.3 Satisfação do usuário

Atualmente, é comum que qualquer pessoa já tenha recebido algum questionário para preencher e dar o nível de satisfação quanto ao produto, serviço ou etc. As instituições, ou empresas que enviam esse tipo de avaliação do usuário buscam melhorar seu serviço com críticas e apontamentos de erros, além de ressaltar a qualidade do serviço prestado com respostas positivas dos seus usuários.

Segundo Adampou, esse tipo de avaliação no contexto de chatbot pode ser medido de duas formas. Considerando que o usuário vai interagir com bot, é

possível que isso seja feito por turnos ou por sessão. A avaliação por sessão é uma métrica de avaliação mais geral, pois o usuário vai avaliar ao final da utilização do bot como foi a sua experiência, em seu artigo é mencionado que foram utilizados formulários variados para os chatbots referenciados no estudo, os mais comuns são avaliações por estrelas em uma escala de zero a cinco com categorias quanto a usabilidade, qualidade das respostas, entendimento do bot e nível de satisfação.

A avaliação por turno é um tipo objetivo de avaliação, porém pode ser mais cansativo ao usuário responder individualmente a cada interação que é feita com o bot. A implementação desse método exige um bloco de código no bot para armazenar essas respostas dadas pelo usuário. Ainda de acordo com Adampou em seu artigo menciona que há variações na forma como o usuário avalia a resposta, o mais comum de ocorrer é uma pergunta ao final de cada saída do bot com algo próximo a “essa pergunta foi útil” acompanhada de uma resposta binária (sim ou não) e ainda uma pergunta sobre melhorias.

4.2 Integrações

É importante estabelecer ampla possibilidade para que o chatbot se integre às mais diversas plataformas, permitindo ao usuário que faça uso do bot em diversos cenários. Para o propósito desse projeto, a viabilidade de integrações mais relevantes vai ser a com websites e mensageiros, uma vez que será de uso da biblioteca da universidade.

O Rasa disponibiliza algumas integrações em seu código fonte, são elas:

- Websites / Apps - A integração com websites pode ser feita de através de dois canais principais, a primeira e mais simples é através de requisições REST, esse método é consolidado pela navegação web e tem compatibilidade com todos os navegadores existentes, uma vez que os browsers são capazes de fazer requisições por default. Utilizar requisições Rest é o método mais tradicional, portanto pode ser mais lento no caso de múltiplas requisições e sessões ativas, isso de deve ao fato de que o servidor vai manter uma sessão ativa que pode ter terminado há algum tempo, já que as requisições não possuem uma ponte de presença entre o browser e o servidor, ou seja, caso o usuário feche a página, o servidor só encerrará a sessão algum tempo após. O segundo método para websites ou aplicativos próprios, é o uso de WebSockets, esse canal faz uso de uma tecnologia avançada que mantém uma sessão ativa entre o browser e o servidor, de forma a criar um túnel de comunicação, a troca de mensagens via WebSockets é extremamente rápida uma vez que estabelecida a conexão.
- Facebook Messenger - O facebook tem se consolidado uma grande ferramenta de mensagens instantâneas desde o lançamento, como rede social, em 2009 fez muito sucesso entre o público jovem, hoje em dia é palco

de diversas idades e empresas que estão presentes na plataforma. A integração com o facebook é viabilizada pelo Rasa em seu arquivo de configuração, e da abertura a alcançar o público que já está na plataforma e utilizar da interface do facebook de mensagens.

- Slack - Essa plataforma é focada especialmente no ambiente de trabalho, tendo como principal diferencial entre os outros mensageiros, a integração com o slack é prevista na documentação do Rasa e é feita de forma semelhante a integração com o Facebook Messenger, e traz os mesmos pontos positivos sobre o público e com relação a interface.
- Telegram - O Telegram é um serviço de mensagens instantâneas assim como o Facebook Messenger e Slack, porém para um público um pouco diferente. O mesmo possui diversas funcionalidades e tem um enorme suporte para desenvolvedores que desejam interagir com o Telegram e utilizá-lo como plataforma de comunicação com seus usuários.

Existem outras plataformas com as quais o Rasa é habilitado a se integrar de forma fácil, são elas o Twillio, Microsoft Bot Framework, Cisco Webex, Rocket Chats, Mattermost, Google Hangouts e além disso é possível fazer uso de canais personalizados utilizando webhooks e canais de áudio.

A integração com o mensageiro Telegram foi a escolhida para implementação devido a facilidade de configuração via WebSocket, e portanto foi utilizada para verificar o comportamento do chatbot em uma prova de conceito. A integração via Web Requests foi testada em ambiente local, porém alguns ajustes precisam ser feitos para evitar problemas de segurança.

4.3 Estruturação

Organizar as frases que serão reconhecidas pelo bot pode ser feito de algumas maneiras distintas no que diz respeito a alguns arquivos. É importante ressaltar que o bot possui uma coleção de perguntas selecionadas com base nos assuntos mais comumente perguntados. Tendo isso em vista, a primeira abordagem é feita tratando as perguntas como um conjunto único de perguntas, cada grupo de perguntas tem características em comum, e o código reconhece o grupo (intent), e depois seleciona a pergunta que mais se encaixa com o escopo perguntado. Esse comportamento deve ser analisado e testado, tendo em vista que o melhor cenário é reconhecer uma pergunta com uma boa taxa de assertividade.

A segunda abordagem é tratar individualmente cada pergunta, criando uma intent para cada pergunta, isso pode ser mais trabalhoso do ponto de vista de escrita, pois cada intent tem que ser configuradas no domínio da aplicação e como regra para o caso de um chatbot Rule Based, no entanto para reconhecimento é estritamente direcionado para a intent que foi cadastrada.

Segundo a documentação do Rasa é recomendado que se use da primeira abordagem para chatbots que tem como objetivo responder a perguntas de forma direta, no estilo de um FAQ (Frequently Asked Questions), isso é válido também para mensagens chamadas de “chitchat”, algo como “papo-furado”, que pode ser entendido pelo chatbot em um momento de descontração do usuário, perguntas como: “qual seu nome?”, “o que é você?”, “quem te criou?”. Além de perguntas, também é possível tratar mensagens agressivas ou amigáveis do usuário e dar uma resposta pré-definida.

Além do que foi mencionado acima, é possível escrever uma série de variações para a mesma pergunta, ou seja, ao identificar que houve uma saudação ao bot, ela pode ser identificada como: “oi”, “olá”, “bom dia”, entre outras formas que podem ser acrescentadas. Fazer uso disso posteriormente trará impactos na matriz de confusão, pois uma frase não pode causar ambiguidade para outra intenção cadastrada.

Após alguns testes, foram identificadas algumas inconsistências que precisaram ser tratadas de uma forma diferente da apresentada na documentação oficial. Uma delas diz respeito ao “fallback”, ou, tratamento de exceções, isso ocorre quando não é identificada com um grau de certeza alto para nenhuma das intenções existentes. É recomendado o uso de um componente chamado “Response Selector”, que foi criado para tratar de bots direcionados para perguntas frequentes e assuntos relacionados, porém ao utilizar dessa abordagem a intenção era acionada, no entanto apesar de nenhuma resposta possuir um nível de confiança suficiente, o bot não entrava em fallback. Esse problema é tema de discussão em fóruns do Rasa e será corrigido em futuros lançamentos, portanto para o projeto foi necessário separar as intenções e não utilizar o Response Selector para esse fim.

4.4 Resultados

As métricas do Rasa são feitas utilizando a biblioteca scikit learn, uma das mais populares feitas na área da inteligência artificial escrita em Python. É possível fazer testes de duas maneiras, a primeira é criar “histórias”, que são uma simulação de conversa com perguntas e respostas previstas. Essa abordagem é mais utilizada para testar fluxos se os fluxos se permanecem consistentes conforme o bot sofre alguma alteração no seu código. A segunda abordagem é uma análise do próprio modelo independente de entradas e saídas. Os resultados obtidos foram os seguintes (Figura 9):

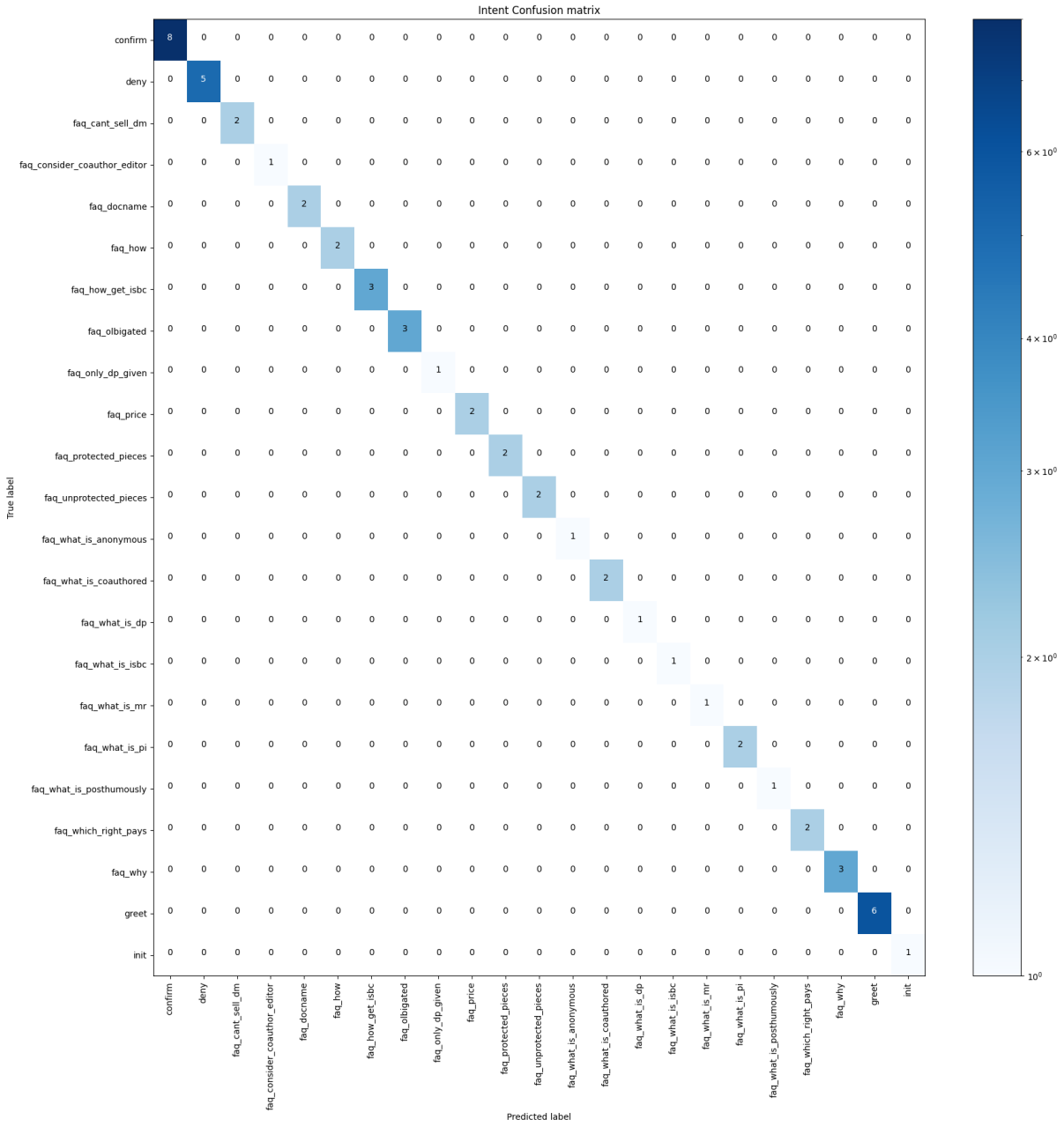

Figura 9 - Matriz de confusão

A matriz de confusão apresentada na figura 9 mostra se houve alguma intenção que, em sua definição original, não foi reconhecida por outra intenção com grau de certeza maior. É possível observar que algumas intenções possuem um número maior de resultados do que outras, isso se deve ao fato de que foi estruturado um maior número de exemplos de frases para as intenções mais comuns. As intenções com mais exemplos são as de “greet” e “affirm”, podemos citar exemplos como: “sim”, “com certeza”, “afirmativo”, “s”. É levado em consideração que podem haver abreviações no caso de uso da digitação como o caso da palavra “sim” ser escrita como apenas “s”.

Como não houveram erros entre as intenções, os scores para essa avaliação automática tiveram resultados um para as métricas calculadas. Não foi realizado um

teste para cada história, evitando criar um teste tendencioso por conta do conhecimento dos caminhos de sucesso e falha do bot.

Através de testes manuais de usabilidade, o chatbot respondeu a abreviações de forma satisfatória, ou seja, frases cadastradas que continham perguntas com a expressão “por que”, poderia ser reconhecida com apenas “pq”, “porque”, e algumas outras variações menores. Isso ocorreu devido a configuração de pipeline das etapas da NLU, especialmente o “Entity Synonym Mapper” que é utilizado para para mapear entidades como: “São Paulo” para “sp”, “sampa” e outros (não sendo restrito para pronomes).

5. Conclusão

O desenvolvimento desse projeto mostrou o nível de maturação que os bots podem chegar até os dias de hoje, servindo como parâmetro para consultas futuras. Além disso, o chatbot criado será utilizado para promover um acesso facilitado para sanar dúvidas sobre direito autoral na biblioteca da Universidade Federal do Maranhão, podendo alcançar todo o público que queira fazer essa consulta via web site ou Telegram.

O uso da biblioteca Rasa se mostrou muito importante para a produção desse software de forma a acelerar o processo de desenvolvimento, apenas com ressalvas a inconsistências no padrão de desenvolvimento do modelo de chatbot para perguntas frequentes que não conta com suporte ao componente de falha de forma natural, apesar disso foi possível obter bons resultados utilizando de outras técnicas de desenvolvimento.

A matriz de confusão e as métricas obtidas mostraram que nenhuma das mensagens cadastradas causou ambiguidades, apesar de ter algum ponto em comum entre elas, uma alteração precisou ser feita para obter esse resultado. Devido a quantidade de similaridade entre algumas perguntas, o algoritmo precisou restringir as similaridades no momento do treino do modelo, isso aumentou a performance do bot de modo geral.

Uma parte do trabalho precisou ser dedicada à integração e conforme mencionado, o Rasa possui integração com diversos canais de mensagem e voz. Os canais escolhidos para desenvolvimento foram os canais de Socket (Web e Mobile) e o Telegram. A escolha se deu devido a celeridade no processo de teste e finalização do desenvolvimento do bot, pois a integração é uma das últimas etapas no processo.

Para realizar o deploy da aplicação foi utilizada a plataforma de serviço em nuvem Heroku, que permite utilização gratuita para serviços simples com algumas limitações, essa etapa do projeto foi feita através da tecnologia de containers do Docker, que permite criar imagens reproduzíveis em qualquer ambiente, essa etapa de configuração precisou ser escrita criteriosamente para possibilitar o deploy.

A etapa de testes do bot automática se mostrou promissora para a quantidade de perguntas cadastradas e o bot respondeu bem a elas. O limiar de erro foi definido com base em desenvolvimento e testes de usabilidade feitos através da verificação, apesar disso, para uma assertividade maior é necessário fazer uma validação com usuários sem conhecimento sobre as perguntas cadastradas. Outro ponto importante no teste com usuários é identificar outras perguntas que não estejam cadastradas e podem ser motivo de dúvida frequente sobre o assunto.

Referências

ADAMOPOULOU, Eleni; MOUSSIADES, Lefteris. **Chatbots: History, technology, and applications, Machine Learning with Applications**, Volume 2, 2020, ISSN 2666-8270, Disponível em: https://www.sciencedirect.com/science/article/pii/S2666827020300062 Acesso em: 13 de junho. 2022

ADAMOPOULOU, Eleni; MOUSSIADES, Lefteris. **An Overview of Chatbot Technology**. In: Maglogiannis I., Iliadis L., Pimenidis E. (eds) Artificial Intelligence Applications and Innovations. AIAI 2020. IFIP Advances in Information and Communication Technology, vol 584. Springer, Cham. Disponível em https://doi.org/10.1007/978-3-030-49186-4_31 Acesso em 13 de junho. 2022

BATES M. **Models of natural language understanding.** Proceedings of the National Academy of Sciences Volume 92, ISSN 0027-8424, 1995. Disponível em: **https://doi.org/10.1073/pnas.92.22.9977** Acesso em: 17 de maio de 2022

HÖHN, S; BONGARD-BLANCHY, K. **Heuristic Evaluation of COVID-19 Chatbots**. Følstad A. et al. (eds) Chatbot Research and Design. CONVERSATIONS 2020. Lecture Notes in Computer Science, vol 12604. Springer, Cham, 2021. Disponível em : https://doi.org/10.1007/978-3-030-68288-0_9. Acesso em: 28 de maio de 2022.

LARISANE KUYVEN, N.; ANDRÉ ANTUNES, C.; JOÃO DE BARROS VANZIN, V.; LUIS TAVARES DA SILVA, J.; LOUREIRO KRASSMANN, A.; MARGARIDA ROCKENBACH TAROUCO, L. **Chatbots na educação: uma Revisão Sistemática da Literatura**. **RENOTE**, Porto Alegre, v. 16, n. 1, 2018. DOI: 10.22456/1679-1916.86019. Disponível em: https://seer.ufrgs.br/index.php/renote/article/view/86019. Acesso em: 25 jun. 2022.

MANTHA, **Introducing DIET: state-of-the-art architecture that outperforms fine-tuning BERT and is 6X faster to train**. Disponível em: https://rasa.com/blog/introducing-dual-intent-and-entity-transformer-diet-state-of-the-art-performance-on-a-lightweight-architecture/, Acesso em 17/06/2022.

MAROENGSIT, Wari; PIYAKULPINYO, Thanarath; POLYIAM, Korawat; PONGNUMKUL, Suporn; CHAOVALIT, Pimwadee; THEERAMUNKONG, Thanaruk. **A Survey on Evaluation Methods for Chatbots**. 2019 Disponível em: https://www.researchgate.net/publication/333524709_A_Survey_on_Evaluation_Methods_for_Chatbots. Acesso em 20 de maio de 2022

MISISCHIA, Chiara Valentina; POECZE, Flora; STRAUSS, Christine, **Chatbots in customer service: Their relevance and impact on service quality**, Procedia Computer Science,
Volume 201, 2022, Pages 421-428, ISSN 1877-0509, Disponível em: https://www.sciencedirect.com/science/article/pii/S1877050922004689. Acesso em 29 de março de 2022

MOLNÁR, G.; SZÜTS, Z.. **The Role of Chatbots in Formal Education,** *2018 IEEE 16th International Symposium on Intelligent Systems and Informatics (SISY)*, 2018, pp. 000197-000202, doi: 10.1109/SISY.2018.8524609. Disponível em: https://ieeexplore.ieee.org/document/8524609. Acesso em 21 de maio de 2022

PAPINENI, Kishore; ROUKOS, Salim; WARD, Todd; ZHU, Wei-Jing. **BLEU: a Method for Automatic Evaluation of Machine Translation**. Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, pp. 311-318, July 2002. Disponível em: https://dl.acm.org/doi/10.3115/1073083.1073135. Acesso em 19 de maio de 2022

PUSHPA, Singh; NARENDRA, Singh; KRISHNA, Kant Singh; AKANSHA, Singh. **Chapter 5 - Diagnosing of disease using machine learning**. Machine Learning and the Internet of Medical Things in Healthcare, Academic Press, 2021, Pages 89-111, ISBN 9780128212295, Disponível em: https://www.sciencedirect.com/science/article/pii/B9780128212295000033. Acesso em 20 de maio de 2022

RASA, **Architecture Overview.** 30 de março de 2022. Disponível em: https://rasa.com/docs/rasa/arch-overview. Acesso em 30 de março de 2022

SANTOS, G. A.; ANDRADE, G. G; SILVA, G. R. S; DUARTE, F. C. M; COSTA, J. P. J. D; SOUSA, R. T. **A Conversation-Driven Approach for Chatbot Management,** in *IEEE Access*, vol. 10, pp. 8474-8486, 2022. Disponível em: https://ieeexplore.ieee.org/abstract/document/9681834. Acesso em: 19 de maio de 2022.

THORAT, Sandeep A; JADHAV, Vishakha, **A Review on Implementation Issues of Rule-based Chatbot Systems**. Proceedings of the International Conference on Innovative Computing & Communications (ICICC) 2020, Disponível em SSRN: https://ssrn.com/abstract=3567047 or http://dx.doi.org/10.2139/ssrn.3567047. Acesso em: 20 de maio de 2022