Implementação de Chatbot para Aprimorar a Comunicação com Usuários de Serviços Públicos

Mônica Rodrigues da Silva
Faculdade de Tecnologia SENAI Mato Grosso – FATEC SENAI MT
monikinha.r@gmail.com

Anita Maria da Rocha Fernandes
Universidade do Vale do Itajaí – UNIVALI
anita.fernandes@univali.br

Guilherme Falcão da Silva Campos
Faculdade de Tecnologia SENAI Mato Grosso – FATEC SENAI MT
guilhermefscampos@gmail.com

ABSTRACT

This paper describes the study and implementation of a chatbot to help users of public services with their most frequent questions. The chatbot was developed based on the TF-IDF (Term Frequency - Inverse Document Frequency) model, using the Python language and the Django framework. Functions such as registration of questions and answers, were implemented using the Java language with API Restful and Spring Boot, and the MongoDB database. Finally, to enable the interaction of internal and external users with the system, front ends were built using the TypeScript language and the Angular platform. The system is in the testing and validation phase.

KEYWORDS

Public Service, Chatbot, Natural Language Processing, TF-IDF

1 INTRODUÇÃO

Muitos usuários reclamam das dificuldades ao tentar obter informações em um determinado órgão público. Por outro lado, dentre outras limitações, a instituição possui a sede localizada na capital, o que dificulta o acesso às pessoas que moram no interior do Estado.

A comunicação do órgão pode ser melhorada automatizando as conversas com os contribuintes através de sistemas de diálogos ou agentes de conversação, mais conhecido como chatbots. Essa ferramenta de software simula através de texto e voz as conversas de seres humanos, podendo ser programado para executar tarefas repetitivas de forma eletrônica.

Sendo assim, foram analisados três soluções diferentes para a escolha do modelo de chatbot que pudesse responder de forma automatizada às perguntas mais frequentes dos usuários que buscam informações no órgão público.

Seguindo as tendências atuais, foi testado a implementação de um modelo de NLP (*Natural Language Processing* ou Processamento de Linguagem Natural) com Deep Learning (Aprendizagem Profunda), utilizando a arquitetura Seq2Seq com Python e Tensorflow. No entanto, devido à necessidade de grande quantidade de dados e recursos computacionais para que os treinamentos e testes pudessem ser realizados de forma efetiva, essa solução foi descartada.

Ao buscar outras tecnologias inovadoras, foi testada a ferramenta Dialogflow da Google, que possui a plataforma de serviços para chatbot e disponibiliza API's (*Application Programming Interface* ou Interface de Programação de Aplicações) que possibilitam a construção de software customizado e integrado aos serviços da nuvem. No entanto, devido à burocracia para contratação de serviços em órgãos públicos, essa solução foi deixada de lado.

Dessa forma, a solução mais adequada para o problema de comunicação no órgão público foi o processamento de linguagem natural clássico, sem o aprendizado independente e contínuo da máquina. Para isso, além da implementação do modelo TF-IDF (Frequência do Termo - Inverso da Frequência nos Documentos), foram desenvolvidos todos os módulos necessários para o funcionamento do sistema de conversação.

2 ARQUITETURA DO SISTEMA CHATBOT

A arquitetura foi escolhida levando em consideração os padrões utilizados nos projetos mais modernos da instituição no qual o sistema será implantado, visando facilitar a manutenção e evolução do software à longo prazo.

Seguindo o padrão, a linguagem Java foi utilizada para o desenvolvimento das funcionalidades da aplicação, e de forma inovadora para a organização, a linguagem Python foi utilizada para a construção das funções de inteligência artificial, conforme ilustra a Figura 1:

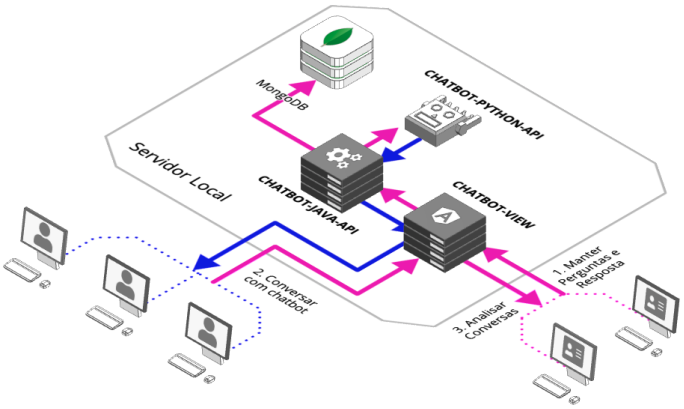

Figura 1: Arquitetura do Projeto

(1) CHATBOT-VIEW: projeto front-end que contém todas as interfaces gráficas da aplicação, desenvolvidas com as linguagens HTML (*HyperText Markup Language* ou Linguagem de Marcação de Hipertexto), CSS (*Cascading Style Sheets* ou Folhas de Estilo em Cascata) e TypeScript, através da plataforma Angular;

(2) CHATBOT-JAVA-API: projeto back-end onde estão centralizadas as implementações das regras de negócio, modelos de dados, acessos ao banco de dados e integração com o projeto chatbot-pyhton-api. Para isso, utilizou-se a arquitetura RESTful (*Representational State*

Transfer ou Transferência Representacional de Estado), linguagem Java, framework Spring Boot, e banco de dados MongoDB;

(3) CHATBOT-PYTHON-API: projeto back-end que possui funções para processamento de linguagem natural atráves de bibliotecas da linguagem Python, que foram disponibilizadas na Web em forma de APIs através da ferramenta Django REST Framework.

3 FUNCIONAMENTO DO SISTEMA CHATBOT

Através do diagrama da Figura 2, é possível obter uma visão geral de alto nível sobre o processo do Sistema Chatbot.

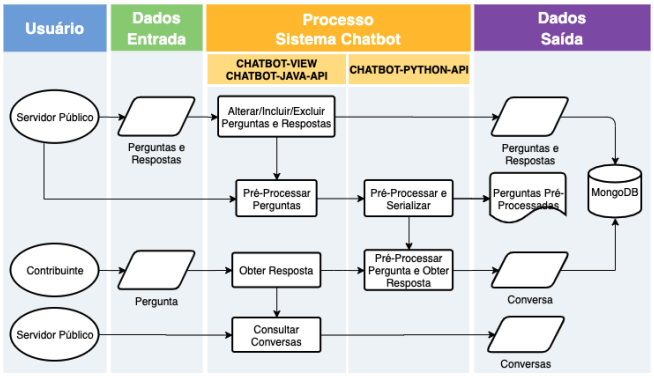

Figura 2: Diagrama

O processo se inicia com a manutenção das perguntas e respostas realizada pelo servidor público. Sendo assim, o usuário pode consultar, incluir, alterar e excluir um conjunto de dados dentro de um mesmo contexto, através da funcionalidade repesentada pela Figura 3. Além disso, o usuário deve solicitar o pré-processamento de todas as perguntas, para que um arquivo possa ser serializado pelo sistema a fim de agilizar o processamento dos dados.

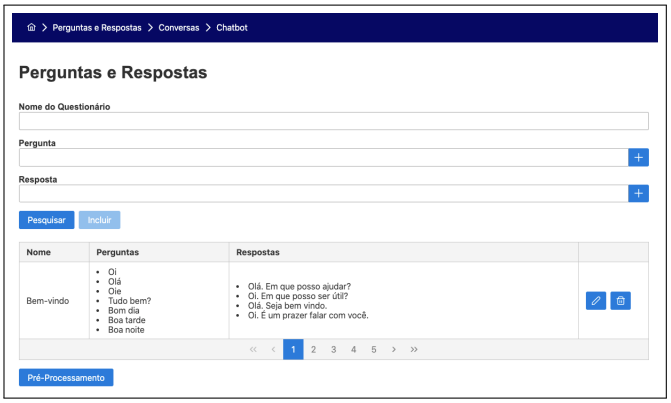

Figura 3: Interface Gráfica - Perguntas e Respostas

Após serializar as perguntas pré-processadas, o contribuinte poderá abrir a tela de chatbot, conforme mostra a Figura 4, onde o sistema irá capturar o texto da mensagem escrito pelo usuário, solicitar o processamento e obtenção da resposta, e por fim exibi-los de forma inversamente sequencial na tela. Além disso, a conversa é armazenada na base de dados, e atualizada a cada obtenção da resposta.

Figura 4: Interface Gráfica - Chatbot

A fim de validar os resultados do chatbot, o servidor público poderá visualizar as conversas através da funcionalidade exibida na Figura 5.

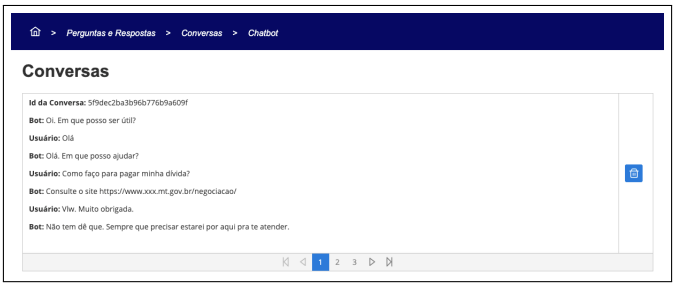

Figura 5: Interface Gráfica - Conversas

4 PROCESSAMENTO DE LINGUAGEM NATURAL

Ao trabalhar com processamento de linguagem natural, os textos devem ser convertidos em números para que os algoritmos possam realizar as operações lógicas. Dessa forma, diferentemente do modelo de classificação Bag-of-Words (Saco de Palavras) que considera somente os números da contagem da repetição de cada palavra, o modelo TF-IDF considera a importância de uma palavra dentro de um documento, com base nas propriedades lexicais e morfológicas do texto [4].

Considerado um dos métodos de ponderação mais populares para recuperação de informações e mineração de dados [4], o cálculo do modelo TF-IDF se baseia no resultado TF que indica a importância do termo, e no valor IDF que atua como um fator de ajuste aumentando o peso dos termos que ocorrem raramente. Com isso, palavras frequentes que muitas vezes não representam informação sobre o contexto, perdem sua importância devido aos resultados dos cálculos abaixo [3]:

- **TF (*Term Frequency* ou Frequência do Termo):** calcula a frequência da palavra no documento atual, através da divisão do número de vezes que um termo aparece pelo total de termos. Exemplo: Um documento com 100 palavras em que a palavra "Big Data" aparece 8 vezes, logo TF = (8/100) = 0.08

- **IDF (*Inverse Document Frequency* ou Inverso da Frequência nos Documentos):** calcula a raridade de uma palavra nos documentos, ou seja, o logaritmo do número de documentos dividido pelo número de documentos que o termo aparece. Exemplo: No total de 100 documentos, a palavra "Big Data" aparece em 20 documentos, logo IDF = log(100/20) = 0.69
- **TF-IDF:** é cálculo da multiplicação dos valores TF e IDF. Conforme o resultado dos exemplos anteriores, TF-IDF = (0.08*0.69) = 0.055

Os cálculos do modelo possuem implementações disponíveis na biblioteca Scikit-learn, que consiste em uma coleção de classes e funções para programas Python [1]. Sendo assim, o processamento do chatbot se dá em etapas conforme a codificação da Figura 6:

```python
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import random


def obter_resposta(pergunta_usuario: str, perguntas_preprocessadas_respostas: list):
    perguntas_preprocessadas = list(perguntas_preprocessadas_respostas.keys())
    pergunta_usuario_preprocessado = preprocessamento_por_texto(pergunta_usuario)
    perguntas_preprocessadas.append(pergunta_usuario_preprocessado)
    # Converte base de dados em uma matriz TF-IDF
    tfidf = TfidfVectorizer()
    palavras_vetorizadas = tfidf.fit_transform(perguntas_preprocessadas)
    print(palavras_vetorizadas)
    # Calcula similaridade do último texto com a base de dados
    similaridade = cosine_similarity(palavras_vetorizadas[-1], palavras_vetorizadas)
    # Retorna índice de maior valor / mais similar
    indice = similaridade.argsort()[0][-2]
    # Converte frase similar em vetor / matriz numpy
    vetor_similar = similaridade.flatten()
    vetor_similar.sort()
    vetor_encontrado = vetor_similar[-2]

    resposta_consulta = ''
    if vetor_encontrado != 0:
        resposta_consulta = random.choice(list(perguntas_preprocessadas_respostas.values())[indice])
    return resposta_consulta
```

Figura 6: Via PyCharm - Obter Resposta

- **Conversão de dados:** através da classe *TfidfVectorizer* contida no módulo de extração de texto, e do método *fit_transform*, a lista de perguntas pré-processadas é convertida em uma matriz de recursos do TF-IDF. Para analisar os processamentos e resultados das operações, é possível invocar métodos para obter os dados dos cálculos, como o *vocabulary_* para obter os ids únicos criados para cada palavra, *idf_* para obter os valores da matriz com os pesos idf calculados, *todense()* para obter os valores da matriz com os resultados dos cálculos TF-IDF de cada palavra, dentre outros [5].
- **Cálculo de similaridade do cosseno:** através da função *cosine_similarity*, são realizados os cálculos da similaridade entre os documentos representados como vetores TF-IDF [2]. Dessa forma, através do texto do usuário e das perguntas em forma de vetor, é obtido o índice da pergunta mais similar.
- **Resposta:** caso encontre um texto similar, o sistema retorna de forma aleatório uma das respostas vinculadas.

5 CONSIDERAÇÕES FINAIS

Após estudos, implementações e testes de alguns modelos de chatbot, foram realizados juntamente com a equipe de TI do órgão público, alinhamentos técnicos para aprovação de um modelo para implementação do robô, das arquiteturas e tecnologias adotadas, das questões frequentes que primordialmente precisam ser respondidas de forma automatizada, além de algumas sugestões para trabalhos futuros.

O entendimento geral foi de que um chatbot simples seria capaz de resolver boa parte dos problemas de comunicação existentes, podendo inclusive proporcionar uma maturidade das áreas de negócio para esse tipo de meio de comunicação que vem se tornando cada vez mais comum. Dessa forma, a arquitetura foi planejada para possibilitar melhorias e mudanças futuras, considerando que o órgão pode decidir fazer investimentos para evoluir o projeto.

Caso resolva migrar para uma plataforma de serviço na nuvem, o projeto CHATBOT-PYTHON-API deixará de existir e o projeto CHATBOT-JAVA-API sofrerá mudanças para consumir as API's da nuvem escolhida. Por outro lado, caso decida investir em aprendizagem profunda, ou mesmo adotar outras técnicas de NLP na linguagem Python, o projeto CHATBOT-PYTHON-API poderá ser evoluído.

Além disso, os dados dos diálogos serão armazenados para que futuramente possam ser realizadas análises de dados e sentimentos, pois ao obter informações relevantes, será possível avaliar o sistema, entender o público e favorecer tomadas de decisão. Outra ideia de melhoria, seria utilizar as API's de outros sistemas da organização, para automatizar operações realizadas por colaboradores da instituição, como por exemplo calcular a dívida do contribuinte e informar o valor e as formas de pagamento.

Por enquanto, o trabalho a médio prazo será o de integrar o projeto CHATBOT-VIEW e CHATBOT-JAVA-API com o Portal de Acesso da instituição via API's, para oferecer uma maior segurança às funcionalidades que serão disponibilizadas somente ao público interno. Além disso, será realizada a fase de homologação junto aos gestores do órgão, para que após a aprovação ocorra a implantação no ambiente de produção.

REFERENCES

[1] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. 2013. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 108–122.

[2] Cosine-similarity. 2020. 6.8. Pairwise metrics, Affinities and Kernels — scikit-learn 0.23.2 documentation. Disponível em: <https://scikit-learn.org/stable/modules/metrics.html#cosine-similarity>. Acesso em 19 Out. 2020.

[3] Jones Granatyr. 2020. Chatbots com Python e Dialogflow: O Guia para Iniciantes. Disponível em: <https://www.udemy.com/course/chatbots-python-dialogflow-iniciantes/learn/lecture/18321698#overview>. Acesso em 01 Ago. 2020.

[4] Manal Mohammed and Nazlia Omar. 2020. Question classification based on Bloom's taxonomy cognitive domain using modified TF-IDF and word2vec. *PLoS ONE* 15, 3 (2020), e0230442.

[5] Scikit-learn.idf. 2020. 6.2. Feature extraction — scikit-learn 0.23.2 documentation. Disponível em: <https://scikit-learn.org/stable/modules/feature_extraction.html?highlight=idf_>. Acesso em 19 Out. 2020.