Daniel Terra Gomes, Andre do Valle Medeiros

Inteligência artificial e Chatbots: um estudo exploratório

Campos dos Goytacazes, RJ

21 de outubro de 2022, v1.0.0

Daniel Terra Gomes, Andre do Valle Medeiros

Inteligência artificial e Chatbots: um estudo exploratório

Relatório Atividade 1 apresentado ao Curso de Ciência da Computação da Universidade Estadual do Norte Fluminense Darcy Ribeiro, como requisito avaliativo da disciplina.

Universidade Estadual do Norte Fluminense Darcy Ribeiro

Ciência da Computação

INF01205 - IA 2022

Campos dos Goytacazes, RJ

21 de outubro de 2022, v1.0.0

Daniel Terra Gomes, Andre do Valle Medeiros

Inteligência artificial e Chatbots: um estudo exploratório

Relatório Atividade 1 apresentado ao Curso de Ciência da Computação da Universidade Estadual do Norte Fluminense Darcy Ribeiro, como requisito avaliativo da disciplina.

Campos dos Goytacazes, RJ
21 de outubro de 2022, v1.0.0

Este trabalho é dedicado às crianças adultas que,
quando pequenas, sonharam em se tornar cientistas.

"Behind me lies a farm.
I wonder if there is bread above the hearth
and if I will ever return.
(Pantheon, League of Legends)

Abstract

This article provides an introduction to AI and Chatbots. Communicating with customers through chat interfaces has, in recent years, become an increasingly popular means of providing real-time customer service in many e-commerce environments. Today, human chat service agents are often replaced by conversational software agents or chatbots, which are systems designed to communicate with human users using Machine Learning (ML) techniques such as Natural Language Processing (NLP), usually based on artificial intelligence (AI). Advances in these areas, particularly in Deep Learning, coupled with the availability of vast computational power and large amounts of data, have led to a new generation of dialog systems and conversational interfaces. Enabling even greater advances in AI. In the practical project developed in this article, we developed a simple Chatbot using 2 neural layers to process messages sent by a user. In this way, it was possible to achieve the desired objective of responding to users with responses from our training base.

Keywords: Chatbots. Artificial Intelligence. Machine Learning.

Resumo

Este artigo fornece uma introdução à IA e Chatbots. A comunicação com os clientes por meio de interfaces de bate-papo, nos últimos anos, tornou-se um meio cada vez mais popular de fornecer atendimento ao cliente em tempo real em muitos ambientes de comércio eletrônico. Hoje, os agentes de serviço de bate-papo humano são frequentemente substituídos por agentes de software de conversação ou chatbots, que são sistemas projetados para se comunicar com usuários humanos usando técnicas de Machine Learning (ML) como o Processamento de Linguagem Natural (PLN), geralmente baseada em inteligência artificial (IA). Avanços nessas áreas, particularmente em Deep Learning, juntamente com a disponibilidade de grande poder computacional e grandes quantidades de dados, levaram a uma nova geração de sistemas de diálogo e interfaces de conversação. Possibilitando avanços ainda maiores em IA. No projeto prático desenvolvido neste artigo elaboramos um Chatbot simples fazendo uso de 2 camadas neurais para o processamento de mensagens enviadas por um usuários. Desse modo, foi possível alcançar o objetivo desejado de responder os usuários com respostas da nossa base de treino.

Palavras-chave: Chatbots. Inteligência Artificial. Machine Learning.

Lista de ilustrações

Lista de tabelas

Listings

Lista de abreviaturas e siglas

NLP	Natural language processing
ML	Machine Learning
IA	Inteligência Artificial

Sumário

Introdução

Nosso projeto propõe a implementação de um Chatbot é um software ou programa de computador que simula conversas humanas por meio de interações de texto ou voz, permitindo que os humanos interajam com dispositivos digitais como se estivessem se comunicando com uma pessoa real. Os chatbots podem ser tão simples quanto programas rudimentares que respondem a uma consulta simples com uma resposta de linha única ou tão sofisticados quanto assistentes digitais que aprendem e evoluem para fornecer níveis crescentes de personalização à medida que coletam e processam informações.

Você provavelmente já interagiu com um chatbot, sabendo ou não. Por exemplo, você está em seu computador pesquisando um produto e uma janela aparece na tela perguntando se você precisa de ajuda. Ou talvez você esteja a caminho de um show e use seu smartphone para solicitar uma carona via chat. Ou você pode ter usado comandos de voz para pedir um café no café do bairro e receber uma resposta informando quando seu pedido estará pronto e quanto custará. [1].

Esses sistemas de Chats podem ser divididos entre dos tipos; orientados a objetivo (Chatbots) que seriam chats que ajudam o usuários a atingir um certo objetivo e com um campo de conhecimento restrito e, também há os Chit Chat cujo trabalham com perguntas abertas, respostas mais naturais e campo de conhecimento aberto de modo a ajudar a encontrar uma informação (QA System) [2].

Um aspecto crítico da implementação do chatbot é selecionar o mecanismo correto de processamento de linguagem natural (NLP) [3]. Esse mecanismo conhecido como NLP O refere-se ao ramo da ciência da computação – e mais especificamente, o ramo da inteligência artificial ou IA – preocupado em dar aos computadores a capacidade de entender texto e palavras faladas da mesma maneira que os seres humanos. A NLP combina linguística computacional – modelagem baseada em regras da linguagem humana – com modelos estatísticos, de aprendizado de máquina e aprendizado profundo. Juntas, essas tecnologias permitem que os computadores processem a linguagem humana na forma de texto ou dados de voz e 'compreendam' seu significado completo, completo com a intenção e o sentimento do falante ou escritor [4].

Sendo assim, propomos realizar uma implementação de um Chatbot Simples (jump to define) capaz de auxiliar um usuário durante uma compra em um Website. Durante o desenvolvimento deste projeto estaremos realizando uma pesquisa exploratória a fim de

[1] <https://www.oracle.com/chatbots/what-is-a-chatbot/>
[2] <https://youtu.be/Mu1N-akGL78>
[3] <https://www.techtarget.com/searchcustomerexperience/definition/chatbot/>
[4] <https://www.ibm.com/cloud/learn/natural-language-processing>

entender o funcionamento de um Software que aplica NLP. Por consequência, ao final deste projeto teremos implementado e apresentado o nosso sistema de Chatbot, e compreendido as tecnologias para o seu funcionamento. Assim alcançando uma aprendizagem eficaz, autodidata, e exploratória.

1 Fundamentação Teórica

1. Como contribuição teórica inicial para o nosso projeto, temos o material do canal no Youtube 'Python Engineer' a playlist de vídeo aulas, 'Chat Bot With PyTorch - NLP Beginner Tutorial' [1]. A partir desse conteúdo poderemos dar início aos nossos desenvolvimentos e pesquisas. No tutorial, é desenvolvido um chatbot simples usando PyTorch e Deep Learning. Também fornecendo uma introdução a algumas técnicas básicas de Processamento de Linguagem Natural (PLN).

2. Seguiremos, também, os conteúdos já aprendidos nas aulas de 'Inteligência Artifical 2022/2 - UENF' para nos auxiliar na formulação teórica do nosso projeto.

3. Salientamos que com o decorrer do projeto nossas referências de materiais, para esse projeto, tenderá a aumentar devido a novas descobertas.

[1] <https://www.youtube.com/playlist?list=PLqnslRFeH2UrFW4AUgn-eY37qOAWQpJyg>

2 Metodologia

Baseado no “Project-based learning” (KRAJCIK; BLUMENFELD, 2006). Seguiremos os estudos através de um projeto que aborda problemas do mundo real, cujo muitos não tem resposta única. Ao longo desse projeto será possível fazer novas perguntas e encontrar suas possíveis respostas por meio de uma investigação sustentada.

Este Plano de Pesquisa também utilizará as seguintes metodologias:

- *Pesquisa Exploratória; visando promover o enriquecimento do conhecimento sobre os diferentes assuntos relacionados a IA, ML, e Chatbots:*
  - *Levantamento Bibliográfico;*
  - *Levantamento documental;*
  - *Minicursos e Vídeo aulas;*
  - *Obtenção de experiências.*

3 Construindo um Chatbot com Pytorch

Já pensou em como Alexa, Siri ou assistente de voz do Google funcionavam? Neste capítulo daremos início à construção de um chatbot para interação com usuários de uma loja virtual.

Antes de continuar, abaixo encontra-se a lista de requisitos para o nosso projeto:

- *Python 3*
  - *Define: Python* é uma linguagem de programação interpretada, orientada a objetos e de alto nível com semântica dinâmica. Suas estruturas de dados embutidas de alto nível, combinadas com tipagem dinâmica e vinculação dinâmica, o tornam muito atraente para o Desenvolvimento Rápido de Aplicativos e para uso como uma linguagem de script ou cola para conectar componentes existentes [1].
- *Dictionaries e Lists*
  - *Define: Listas* são tipos de dados mutáveis em Python. Lists é um tipo de dados de índice baseado em 0, o que significa que o índice do primeiro elemento começa em 0. As listas são usadas para armazenar vários itens em uma única variável. As listas são um dos 4 tipos de dados do Python, ou seja, Listas, Dicionário, Tupla e Conjunto [2].
  - *Define: Dicionários* Dicionários são a implementação do Python de uma estrutura de dados que é mais conhecida como uma matriz associativa. Um dicionário consiste em uma coleção de pares chave-valor. Cada par de chave-valor mapeia a chave para seu valor associado[3].
- *NumPy*
  - *Define: NumPy* é o pacote fundamental para computação científica em Python. É uma biblioteca Python que fornece um objeto array multidimensional, vários objetos derivados (como arrays e matrizes mascarados) e uma variedade de rotinas para operações rápidas em arrays, incluindo matemática, lógica, manipulação de formas, classificação, seleção, E/S , Fourier discreto transforma álgebra linear básica, operações estatísticas básicas, simulação aleatória e muito mais[4].

[1] <https://www.python.org/doc/essays/blurb/>
[2] <https://www.analyticsvidhya.com/blog/2021/06/working-with-lists-dictionaries-in-python/>
[3] <https://realpython.com/python-dicts>
[4] <https://numpy.org/doc/stable/user/whatisnumpy.html>

- *Pandas*
  - *Define: Pandas* é um pacote Python de código aberto que é mais amplamente usado para ciência de dados/análise de dados e tarefas de aprendizado de máquina. Ele é construído em cima de outro pacote chamado Numpy, que fornece suporte para arrays multidimensionais[5].
- *Pytorch*
  - *Define: PyTorch* é uma biblioteca de aprendizado de máquina para Python usada principalmente para processamento de linguagem natural. O software de código aberto foi desenvolvido pelas equipes de inteligência artificial do Facebook Inc. em 2016. O PyTorch oferece dois recursos significativos, incluindo computação de tensor, bem como redes neurais profundas funcionais.
- *Natural Language Processing (Bag of Words)*
  - *Define: NLP ou PLN* O processamento de linguagem natural usa aprendizado de máquina para revelar a estrutura e o significado do texto. Com aplicativos de processamento de linguagem natural, as organizações podem analisar texto e extrair informações sobre pessoas, lugares e eventos para entender melhor o sentimento da mídia social e as conversas com os clientes [6].

3.1 Importando Bibliotecas Relevantes

```
1
2   import numpy as np
3   import random
4   import json
5   import nltk
6   import torch
7   import torch.nn as nn
8   from torch.utils.data import Dataset, DataLoader
```

Listing 3.1 – Python Bibliotecas

3.2 Criando funções personalizadas

Criaremos funções personalizadas para que seja fácil implementá-las posteriormente.

[5] <https://www.activestate.com/resources/quick-reads/what-is-pandas-in-python-everything-you-need-to-know/>

[6] <https://cloud.google.com/learn/what-is-natural-language-processing>

```
def tokenize(sentence):
    return nltk.word_tokenize(sentence)

def stem(word):
    return stemmer.stem(word.lower())
```

Listing 3.2 – Python Funções personalizadas

Nltk ou kit de ferramentas de linguagem natural é uma biblioteca realmente útil que contém classes importantes que serão úteis em qualquer uma de suas tarefas de PNL.

3.3 Stemming

Se tivermos as seguintes palavras como Andei, Ande, Andarei, Andamento, Andando, Andante, podem parecer palavras diferentes, mas geralmente têm o mesmo significado e também a mesma forma base; "and". O processo de stemização (do inglês, stemming) consiste em reduzir uma palavra ao seu radical. A palavra “meninas” se reduziria a “menin”, assim como “meninos” e “menininhos”. As palavras “gato”, “gata”, “gatos” e “gatas” reduziriam-se para “gat” [7]. Então, para que nosso modelo entenda todas as formas diferentes das mesmas palavras, precisamos treinar nosso modelo com essa forma. Isso é chamado de Stemming. Existem diferentes métodos que podemos usar para derivação. Aqui usaremos o modelo Porter Stemmer da nossa biblioteca NLTK.

```
from nltk.stem.porter import PorterStemmer
stemmer = PorterStemmer()
```

Listing 3.3 – Python Stemming

3.4 Bag of Words

Vamos dividir cada palavra em frases e adicioná-la a um array. Nós estaremos usando um saco de palavras. Que inicialmente será uma lista de zeros com um tamanho igual ao comprimento do array all-words. Se tivermos um array de ‘ frases = ["ola", "como", "voce", "esta"]‘ e um array de total ‘ words = ["oi", "olá", "eu", "você, "tchau", "obrigado", "legal"]‘ então seu conjunto de palavras será ‘bog = [0 , 1 , 0 , 1 , 0 , 0 , 0]‘. Faremos um loop sobre cada palavra no array all-words e o array bog correspondente a cada palavra. Se uma palavra da frase for encontrada no array all words, 1 será substituído naquele índice/posição no array bag.

[7] <https://www.computersciencemaster.com.br/como-reduzir-uma-palavra-ao-seu-radical-em-python-stemming/>

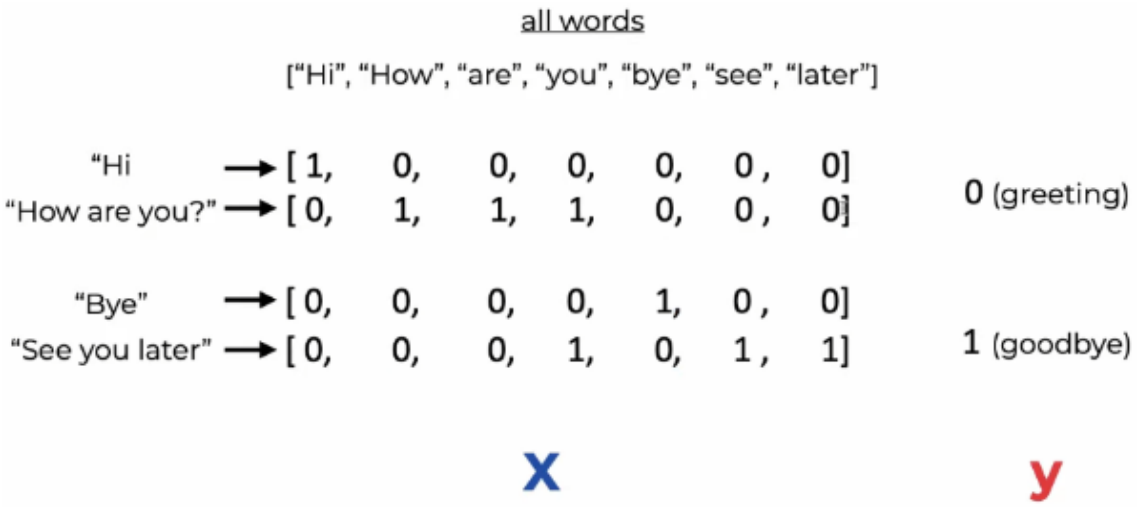

Figura 1 – Bag of Words

Usa Bag of Words para separar uma frase em várias palavras. Imagem retirada da Fundamentação Teórica 1.

```
def bag_of_words(tokenized_sentence, words):
    """
    return bag of words array:
    1 para cada palavra conhecida que existe na frase,
    0 caso contrario.


    Exem,plo:


    sentence = ["ola", "como", "esta", "voce"]
    words = ["oi", "ola", "eu", "voce", "tchau",
     "obrigado", "legal"]
    bag   = [  0 ,    1 ,    0 ,   1 ,    0 ,    0 ,      0]
    """
    # stem cada palavra
    sentence_words = [stem(word) for word in tokenized_sentence]
    # initialize bag com 0 para cada palavra
    bag = np.zeros(len(words), dtype=np.float32)
    for idx, w in enumerate(words):
        if w in sentence_words:
            bag[idx] = 1

    return bag
```

Listing 3.4 – Python Bag of Words

Durante o processo, também usaremos **nltkwordtokenize()** que converterá uma única string de sentença em uma lista de palavras. Por exemplo, se você passar **"Ola, como voce**

esta?", ele retornará **"ola", "como", "voce", "esta"**. **Observação:** Passaremos palavras em minúsculas para o **Stemmer** para que palavras como Bom e bom (em maiúsculas) não sejam rotuladas como palavras diferentes.

3.5 Carregando os dados e Data Cleaning

Usaremos um conjunto de dados chamado **intents.json** que tem a estrutura mostrada no campo abaixo. Estaremos limpando esses dados de acordo com as nossas necessidades usando as funções que criamos anteriormente.

```
{
  "intents": [
    {
      "tag": "saudacao",
      "patterns": [
        "Oi",
        "Ei",
        "Como voce esta",
        "Tem alguem ai?",
        "Ola",
        "Bom dia"
      ],
      "responses": [
        "Ei :-)",
        "Ola, obrigado pela visita",
        "Ola, o que posso fazer por voce?",
        "Ola, como posso ajudar?"
      ]
    }
  ]
}
```

Listing 3.5 – Data intents.json

Agora vamos simplesmente carregar o arquivo json usando a função **jsonload**.

```
with open('intents.json', 'r') as f:
    intents = json.load(f)
```

Listing 3.6 – Python Carregando dados

Para obter as informações corretas, iremos descompactá-las com o seguinte código:

```
all_words = []
tags = []
xy = []
# loop atraves de cada frase no nosso intents patterns
for intent in intents['intents']:
```

```
    tag = intent['tag']
    # adicionar a tag list
    tags.append(tag)
    for pattern in intent['patterns']:
        # tokenize cada palavra na frase
        w = tokenize(pattern)
        # adicionar a nossa lista de palavras
        all_words.extend(w)
        # adicionar ao par xy
        xy.append((w, tag))
```

Listing 3.7 – Python Loop nos dados

Isso separará todas as tags e palavras em suas listas.

3.6 Limpeza e preparação dos dados

Estaremos usando nossas funções personalizadas e limpando os dados implementando as funções que criamos em nossas células anteriores.

```

# stem e lower cada palavra
ignore_words = ['?', '.', '!']
all_words = [stem(w) for w in all_words if w not in ignore_words]
# remover duplicados e sort
all_words = sorted(set(all_words))
tags = sorted(set(tags))

print(len(xy), "patterns") # padroes
print(len(tags), "tags:", tags) # tags
print(len(all_words), "unique stemmed words:", all_words) # palavras
    derivadas unicas
```

Listing 3.8 – Data Cleaning

Criando dados de treinamento: transformaremos os dados em um formato que nosso modelo PyTorch possa entender facilmente

```

# criar dados de treinamento
X_train = []
y_train = []
for (pattern_sentence, tag) in xy:
    # X: bag de palavras para cada pattern_sentence
    bag = bag_of_words(pattern_sentence, all_words)
    X_train.append(bag)
    # y: PyTorch CrossEntropyLoss precisa apenas class labels, nao one-
    hot
```

```
    label = tags.index(tag)
    y_train.append(label)

X_train = np.array(X_train)
y_train = np.array(y_train)
```

Listing 3.9 – Training Data

3.7 Modelo PyTorch

Aqui estaremos fazendo uma classe para implementar nossa rede neural personalizada. Será uma Rede neural feed-forward que terá 3 Camadas Lineares e usaremos a função de ativação "ReLU". Nota: Usamos a função super() para herdar as propriedades de sua classe pai. Este é um conceito de Programação Orientada a Objetos (OOP).

- *Define: ReLU* é uma função de ativação não linear que é usada em redes neurais multicamadas ou redes neurais profundas. Esta função pode ser representada como:

$$f(x) = \max(0, x) \qquad (1)$$

Figura 2 – ReLu

onde x = um valor de entrada.

De acordo com a equação 1, a saída de ReLu é o valor máximo entre zero e o valor de entrada. A saída é igual a zero quando o valor de entrada é negativo e o valor de entrada quando a entrada é positiva. Assim, podemos reescrever a equação 1 da seguinte forma:

$$f(x) = \begin{cases} 0, if\ x < 0 \\ x, if\ x \geq 0 \end{cases} \qquad (2)$$

Figura 3 – ReLu

onde x = um valor de entrada.

Exemplos de ReLu: Dadas diferentes entradas, a função gera diferentes saídas. Por exemplo, quando x é igual a -5, a saída de f(-5) é 0. Por outro lado, a saída de f(0) é 0 porque a entrada é maior ou igual a 0. Além disso, o resultado de f (5) é 5 porque a entrada é maior que zero [8].

[8] <https://deepai.org/machine-learning-glossary-and-terms/relu/>

3.7.1 Feed Forward Neural Network

Define: Feed Forward Neural Network Uma Rede Neural Feed Forward é uma rede neural artificial na qual as conexões entre os nós não formam um ciclo. O oposto de uma rede neural feed-forward é uma rede neural recorrente, na qual certos caminhos são ciclados. O modelo feed-forward é a forma mais simples de uma rede neural, pois a informação é processada apenas em uma direção. Embora os dados possam passar por vários nós ocultos, eles sempre se movem em uma direção e nunca para trás [9].

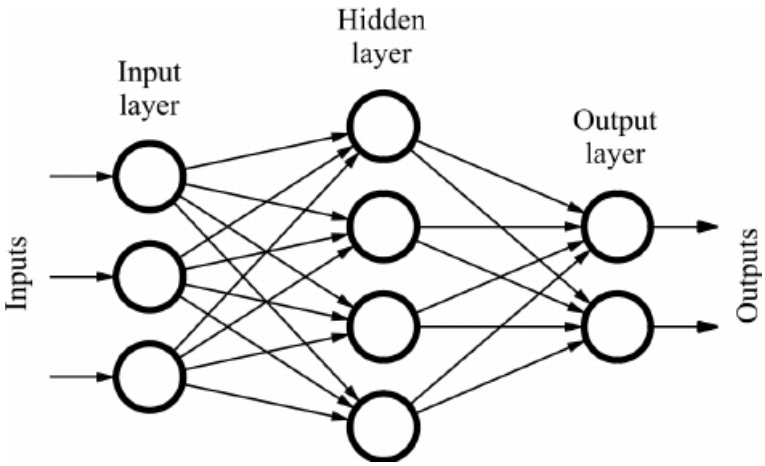

Figura 4 – Feed Forward Neural Network

Exemplo de uma rede neural feed-forward.

Desse modo, uma representação válida para o nosso projeto seria a FFNN abaixo:

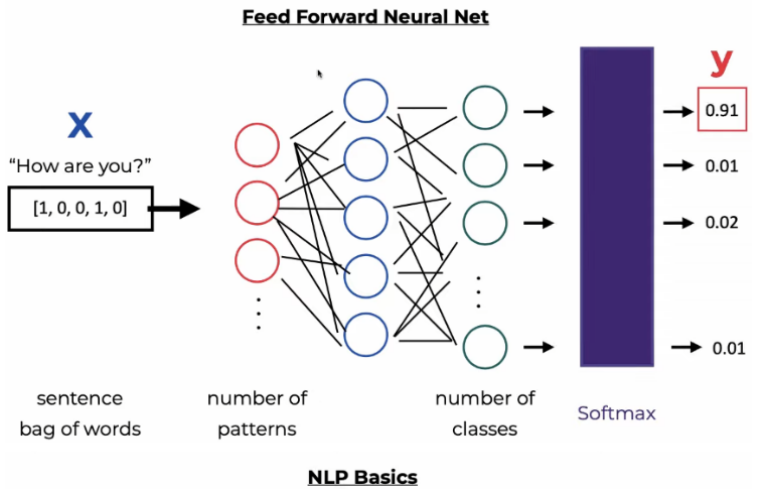

Figura 5 – Feed Forward Neural Network

Exemplo de uma rede neural feed-forward para o nosso projeto. Imagem retirada da Fundamentação Teórica 1.

[9] <https://deepai.org/machine-learning-glossary-and-terms/feed-forward-neural-network>

Referências

KRAJCIK, J. S.; BLUMENFELD, P. C. *Project-based learning*. [S.l.]: na, 2006. Citado na página 25.