

Homer Chatbot

Marcos S. W. Landi, Frederico Messa Schwartzhaupt

Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

{mswlandi, fmschwartzhaupt}@inf.ufrgs.br

Abstract. *This paper presents our final work project, a chatbot, for the discipline of "Formal Languages and Automata", carried out in the first semester of 2019, at the Federal University of Rio Grande do Sul. The chatbot in question was developed to simulate the behavior, in any user-induced conversation, of Homer Simpson's character in the American animated series "The Simpsons," using the neural network models "Seq2Seq" and "Doc2Vec".*

Resumo. *Este artigo apresenta o nosso projeto de trabalho final, um chatbot, para a disciplina de "Linguagens Formais e Autômatos", cursada no primeiro semestre de 2019, na Universidade Federal do Rio Grande do Sul. O chatbot em questão foi desenvolvido para simular o comportamento, em qualquer conversação induzida pelo usuário, do personagem Homer Simpson da série de animação norte-americana "Os Simpsons", utilizando os modelos de redes neurais "Seq2Seq" e "Doc2Vec".*

1. Motivação



Figura 1. Homer e seu computador

Quando nos foi apresentado do projeto de trabalho final seria desenvolver um chatbot qualquer, sem restrições de tecnologia, logo tivemos a ideia de criar um modelo de redes neurais que aprendesse a conversar, claro, se conseguíssemos um conjunto de dados grande o bastante para treiná-la...

Então, um de nós se lembrou que existia um conjunto de dados de falas dos episódios de “Os Simpsons” desde 1989 até 2015 – e assim começou a nossa jornada para criar uma inteligência artificial cujo único propósito de existência é ser o Homer Simpson.

Me: hello homer
Homer: shut up, flanders

Figura 2. O propósito do Chatbot.

2. Introdução

Nós resolvemos desenvolver o chatbot utilizando um modelo de redes neurais conhecido como “Seq2Seq”. Posteriormente, após resultados não muito promissores, passamos a utilizar um segundo modelo, conhecido como “Doc2Vec”. Para entender porque nenhum dos modelos acabou dando muito certo, primeiro vamos dar uma olhada no conjunto de dados, e só mais tarde nos dois modelos.

Também fizemos experimentações com outras pessoas, e relacionamos o que foi feito aqui com o que foi visto na disciplina de “Linguagens Formais e Autômatos”.

Porém, antes que possamos prosseguir, já adiantamos que caso o leitor deseje rodar os códigos, é preciso que copie os arquivos do nosso projeto^[1], e os cole dentro de uma pasta com o nome "homer_chatbot", dentro da pasta principal do Google Drive de uma Conta Google, pois as células de código tomam este diretório como referência.

2.1. Conjunto de Dados

O conjunto de dados dos episódios de “Os Simpsons”^[2], disponibilizado virtualmente, contém falas de mais ou menos 600 episódios, totalizando mais de 100 mil falas.

id	episode_id	speaking_line	character_id	location_id	raw_character_text	raw_location_text	spoken_words	word_count
9549	32	True	464.0	3.0	Miss Hoover	Springfield Elementary	No, actually, it was a little of both. Some	31.0
9550	32	True	9.0	3.0	Lisa Simpson	Springfield Elementary	Where's Mr. Bergstrom?	3.0
9551	32	True	464.0	3.0	Miss Hoover	Springfield Elementary	I don't know. Although I'd sure like to ta	22.0
9552	32	True	9.0	3.0	Lisa Simpson	Springfield Elementary	That life is worth living.	5.0
9553	32	True	40.0	3.0	Edna Krabappel-Flar	Springfield Elementary	The polls will be open from now until th	33.0
9554	32	True	38.0	3.0	Martin Prince	Springfield Elementary	I don't think there's anything left to say.	8.0
9555	32	True	40.0	3.0	Edna Krabappel-Flar	Springfield Elementary	Bart?	1.0
9556	32	True	8.0	3.0	Bart Simpson	Springfield Elementary	Victory party under the slide!	5.0
9557	32	False		374.0		Apartment Building		
9558	32	True	9.0	374.0	Lisa Simpson	Apartment Building	Mr. Bergstrom! Mr. Bergstrom!	4.0
9559	32	True	469.0	374.0	Landlady	Apartment Building	Hey, hey, he Moved out this morning. H	19.0
9560	32	True	9.0	374.0	Lisa Simpson	Apartment Building	Do you know where I could find him?	8.0
9561	32	True	469.0	374.0	Landlady	Apartment Building	I think he's taking the next train to Capit	10.0
9562	32	True	9.0	374.0	Lisa Simpson	Apartment Building	The train, how like him... traditional, ye	9.0
9563	32	True	469.0	374.0	Landlady	Apartment Building	Yes, and it's been the backbone of our c	19.0
9564	32	True	9.0	374.0	Lisa Simpson	Apartment Building	I see he touched you, too.	6.0
9565	32	False		3.0		Springfield Elementary	School	
9566	32	True	8.0	3.0	Bart Simpson	Springfield Elementary	Hey, thanks for your vote, man.	6.0
9567	32	True	101.0	3.0	Nelson Muntz	Springfield Elementary	I didn't vote. Voting's for geeks.	6.0
9568	32	True	8.0	3.0	Bart Simpson	Springfield Elementary	Well, you got that right. Thanks for your	10.0
9569	32	True	467.0	3.0	Terri/sherri	Springfield Elementary	We forgot.	2.0

Figura 3. O banco de dados.

Como se pode ver, cada linha corresponde a uma fala e possui 13 colunas (algumas omitidas na imagem), com diversas informações que podem ser potencialmente úteis, como o local, quem fala, e claro, a própria fala.

Você pode estar pensando que um conjunto de 100 mil falas é um conjunto enorme, mas iremos explicar porque não é o caso:

- No tutorial de tradução neural do “TensorFlow”^[3], um conjunto de dados considerado pequeno possui 133 mil pares de falas.
- No caso do modelo “Seq2Seq”, é preciso separar uma parte do conjunto de dados para ser o conjunto de validação (mais sobre isso a seguir), e não é usado para treinar a inteligência artificial.
- Nem todas as falas são falas utilizáveis para nossos propósitos:
 - Algumas não são de diálogo.
 - Como o chatbot vai imitar o Homer, só podemos usar as falas dele, pareadas com as falas de outros personagens, as quais ele está respondendo. (no caso do modelo “Doc2Vec” podemos usar todas falas para treinar, mais sobre isso adiante).
 - Como ambos modelos recebem frases de entrada, devemos concatenar as falas adjacentes de um mesmo personagem em uma única fala.

Após todos estes filtros, nos resta apenas 15 mil pares de falas, o que pode ser considerado insuficiente para treinar uma rede neural do zero para simular um personagem como o Homer.

2.2. Conjunto de Treino

Usando dois scripts em Python (que estão no repositório), organizamos o conjunto de dados da seguinte forma:

- “Seq2Seq”: (os arquivos “train” foram usados no treino e os arquivos “test” foram usados na validação)
 - “train.a” e “test.a” – cada linha possui uma fala direcionada ao Homer.
 - “train.b” e “test.b” – cada linha possui uma resposta do Homer.
- “Doc2Vec”: (semelhante aos arquivos do “Seq2Seq”, porém com diálogos entre outros personagens abaixo)
 - questions.txt – mesma formatação que os arquivos de extensão “.a” do “Seq2Seq”.
 - answers.txt – mesma formatação que os arquivos de extensão “.b” do “Seq2Seq”.

```
# Fazendo o Drive estar disponível como pastas acessíveis para o python
e shell
from google.colab import drive
drive.mount('/content/gdrive', force_remount=True)

%%bash
# Copiando os dados de treino e teste do Drive para a pasta de trabalho
- Seq2Seq
```

```

cp /content/gdrive/My\ Drive/homer_chatbot/homer_train.a ./train.a
cp /content/gdrive/My\ Drive/homer_chatbot/homer_train.b ./train.b
cp /content/gdrive/My\ Drive/homer_chatbot/homer_test.a ./test.a
cp /content/gdrive/My\ Drive/homer_chatbot/homer_test.b ./test.b

%%bash
# Copiando os dados de treino e teste do Drive para a pasta de trabalho
- Doc2Vec
cp /content/gdrive/My\ Drive/homer_chatbot/questions.txt\
/content/questions.txt
cp /content/gdrive/My\ Drive/homer_chatbot/answers.txt\
/content/answers.txt

```

3. Modelo “Seq2Seq”

Possuindo os arquivos organizados em seus devidos formatos, vamos falar sobre o primeiro modelo – “Seq2Seq”^[4].

“Seq2Seq” é um modelo criado pela Google para traduções automáticas, sumarização de textos, modelagem conversacional, legendas automáticas de imagens, e mais – segundo a mesma. Ele é um modelo generativo, ou seja, gera novos dados a partir do aprendizado que ele absorveu com os dados de entrada.

Na verdade, nós vamos utilizar um modelo já construído a partir do “Seq2Seq”, o “TensorFlow-nmt”, feito para tradução neural, mas ao invés de traduzir frases de uma língua para a outra, vamos traduzir frases para o Homer para frases do Homer.

```

%%bash
# Baixando o nmt (tf-seq2seq)
rm -rf /content/nmt_model
rm -rf nmt
git clone https://github.com/tensorflow/nmt/

```

Isso funciona porque tradução neural consiste em abstrair o significado de uma frase em uma língua e traduzir este significado em outra língua, mas o mesmo princípio pode ser usado para chatbots: a rede neural abstrai o significado de uma fala, e a partir dele, busca responder de acordo com este significado, na mesma língua.

Na verdade, são duas redes neurais em uma - uma "encoder" - que aprende o significado da frase de entrada, representado por um vetor, e uma "decoder", que recebe esse vetor, e aprende a responder a ele:

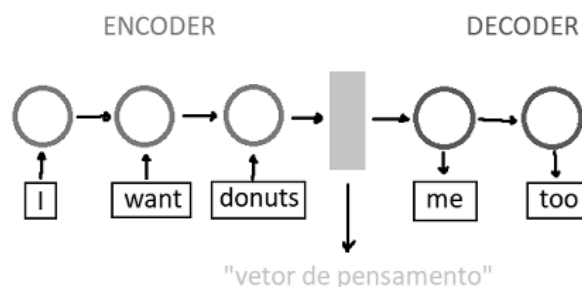


Figura 4. O aprendizado da rede neural.

Nós, então, dividimos o conjunto de entrada entre treino e validação para evitar o chamado "overfitting" – quando o modelo "aprende demais". A seguir uma imagem que explica intuitivamente como um modelo de predição pode aprender demais:

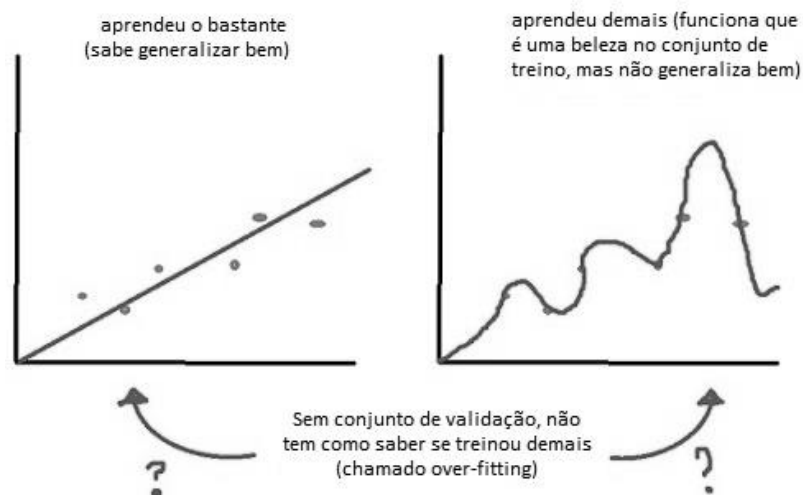


Figura 5. O resultado de um aprendizado excessivo.

Antes de treinar a rede neural, precisamos construir um vocabulário, um conjunto de todas as palavras que a rede neural pode utilizar para entender e responder. Podemos fazer isso apenas dando o conjunto de todas as palavras vistas no conjunto de dados, mas fazer isto criaria um vocabulário potencialmente muito grande, e sem certas conexões úteis entre palavras similares.

Uma maneira de resolver isto é dividir as palavras em subpalavras, por exemplo "loved" seria uma composição de "lov" e "ed", e "loving" seria uma composição de "lov" e "ing". Isso faz com que o modelo possa generalizar melhor para palavras novas e ainda diminui o tamanho do vocabulário.

Existem vários métodos de fazer isso, e o que vamos usar é o “Byte-Pair-Encoding” (BPE). Para isso, usamos o repositório “subword-nmt”:

```
%%bash
# Clonando o repositório subword-nmt
rm -rf subword-nmt
git clone https://github.com/b0noI/subword-nmt.git
cd subword-nmt
git checkout dbe97c8f95f14d06b2e46b8053e2e2f9b9bf804e
cd /content/
# Criando o vocabulário de palavras únicas a partir dos dados de treino
subword-nmt/learn_joint_bpe_and_vocab.py --input ./train.a ./train.b -s 50000 -o code.bpe --write-vocabulary vocab.train.bpe.a vocab.train.bpe.b
# Removendo os tabs inúteis dos vocabulários
sed -i '/\t/d' ./vocab.train.bpe.a
sed -i '/\t/d' ./vocab.train.bpe.b
# Fazendo novos arquivos de vocabulário (sem as frequências, pra usar no tf-seq2seq)
cat vocab.train.bpe.a | cut -f1 --delimiter=' ' > revocab.train.bpe.a
cat vocab.train.bpe.b | cut -f1 --delimiter=' ' > revocab.train.bpe.b
```

Tendo criado os vocabulários (um para os .a e outro para os .b), aplicamos ele nos conjuntos de treino.

```
%%bash
# Aplicando os vocabulários com sub-palavras em todos os arquivos (treino e teste)
subword-nmt/apply_bpe.py -c code.bpe --vocabulary vocab.train.bpe.a --vocabulary-threshold 5 < ./train.a > train.bpe.a
subword-nmt/apply_bpe.py -c code.bpe --vocabulary vocab.train.bpe.b --vocabulary-threshold 5 < ./train.b > train.bpe.b
subword-nmt/apply_bpe.py -c code.bpe --vocabulary vocab.train.bpe.a --vocabulary-threshold 5 < ./test.a > test.bpe.a
subword-nmt/apply_bpe.py -c code.bpe --vocabulary vocab.train.bpe.b --vocabulary-threshold 5 < ./test.b > test.bpe.b
```

Esta próxima célula de código só deve ser executada, em caso de interesse em carregar, do Google Drive, um modelo já treinado anteriormente, para continuar o treinamento do mesmo, ou testar o modelo já treinado.

```
# Carregando modelo treinado anteriormente
!rm -rf /content/nmt/nmt_model
!cp -r /content/gdrive/My\ Drive/homer_chatbot/nmt_model /content/nmt/nmt_model
!echo -e "\nloaded\n"
```

E agora, finalmente, o treinamento!

```
# Treinando o modelo seq2seq
import json
# Loop para treinar por 10.000 passos de cada vez,
# salvando no drive entre iterações.
for i in range(1,50):
    # Atualizando Parâmetro de Passos
    filename = '/content/nmt/nmt_model/hparams'
    with open(filename, 'r') as f:
        data = json.load(f)
        data["num_train_steps"] = i*20*500 + 457690
    get_ipython().system("rm " + filename)
    with open(filename, 'w') as f:
        json.dump(data, f, indent=4)
    # Treinando de fato
    !cd nmt && python3 -m nmt.nmt \
      --src=a --tgt=b \
      --vocab_prefix=../revocab.train.bpe \
      --train_prefix=../train.bpe \
      --dev_prefix=../test.bpe \
      --test_prefix=../test.bpe \
      --out_dir=nmt_model \
      --num_layers=2 \
      --num_gpus=1
    # Salvando o progresso
    !rm -rf /content/gdrive/My\ Drive/homer_chatbot/nmt_model
    !cp -r /content/nmt/nmt_model /content/gdrive/My\ Drive/homer_chatbot
    !echo -e "\nsaved\n"
```

Após o treinamento, já podemos criar uma função para conversar com a rede neural, fazendo inferências novas a partir de um arquivo “input.txt”:

```
def chatbot_seq2seq():
    quit = False
    while(quit == False):
```

```

text = input("Me: ")
if(text == "quit()"):
    quit = True
else:
    with open("/content/input.txt", "w") as input_file:
        input_file.write(text)
        !/content/subword-nmt/apply_bpe.py -c /content/code.bpe --
        vocabulary /content/vocab.train.bpe.a --vocabulary-threshold 5 <
        /content/input.txt > /content/input.bpe
        !cd /content/nmt && python -m nmt.nmt \
        --out_dir=nmt_model \
        --inference_input_file=/content/input.bpe \
        --inference_output_file=/content/output.txt > /dev/null 2>&1
    with open("/content/output.txt", "r") as output:
        print('Homer: ', output.read().replace('@@', ' '),
        '').strip("@@\n"))
        print('\n')

```

4. Modelo “Doc2Vec”

O modelo “Doc2Vec” é semelhante ao “Seq2Seq” no sentido de percepção de “sentido” para frases, mas a semelhança para por aí. Dentro de si, ele utiliza o modelo “Word2Vec”, que atribui significado a palavras em vetores numéricos, e vai um passo além, usando os significados de múltiplas palavras para inferir o significado de uma frase – um *documento*.

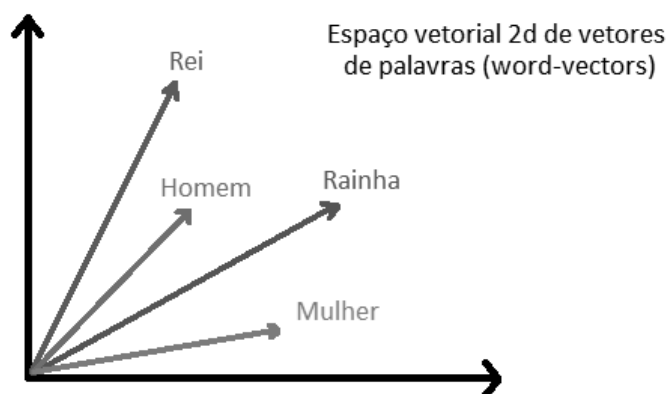


Figura 6. A vetorização da linguagem.

Possuindo o vetor que representa a frase de entrada, procuramos entre as falas de “Os Simpsons” que foram faladas para o Homer, qual que é a mais parecida, ou seja, qual vetor que é o mais similar (ou o mais próximo) ao vetor gerado pelo modelo, e retornamos a resposta que o Homer deu para esta frase. Isso configura um modelo de diálogo “retrieval-based”, ou seja, baseado em busca de frases já prontas, diferentemente dos modelos generativos, que geram texto novo.

A biblioteca que utilizamos para isto é a “gensim”^[5], também originalmente publicada pela Google, que serve para implementação de modelos “Word2Vec” e “Doc2Vec”.

Nessa versão, como não iremos gerar conteúdo novo, criamos um vocabulário com palavras inteiras, onde as palavras que aparecem apenas uma vez são ignoradas e tratadas como palavras desconhecidas.

Execute a próxima célula de código se e somente se quiser carregar o modelo já treinado, do Google Drive:

```
# Carregando modelo treinado anteriormente
import gensim
from gensim.models.doc2vec import Doc2Vec, TaggedDocument
from gensim.utils import simple_preprocess
import multiprocessing
import os
!rm -rf /content/doc2vec.model
!rm -rf /content/doc2vec.model.docvecs.vectors_docs.npy
!cp /content/gdrive/My\ Drive/homer_chatbot/doc2vec.model
/content/doc2vec.model
!cp /content/gdrive/My\
Drive/homer_chatbot/doc2vec.model.docvecs.vectors_docs.npy
/content/doc2vec.model.docvecs.vectors_docs.npy
doc2vec_model = Doc2Vec.load("/content/doc2vec.model")
!echo -e "\nloaded\n"
```

Já caso queira treinar, execute isto:

```
# Treinando o modelo
import gensim
from gensim.models.doc2vec import Doc2Vec, TaggedDocument
from gensim.utils import simple_preprocess
import multiprocessing
import os
# Só tendo certeza de que será usado o compilador C (para treinar mais rápido)
assert gensim.models.doc2vec.FAST_VERSION > -1
print('Fazendo uns paranauês matemáticos...')
# Precisamos do número de palavras no conjunto de treino
with open("questions.txt", "r") as f:
    n_words = len(f.read().split())
cores = multiprocessing.cpu_count()
# Criando o modelo com 200 dimensões de vetores, palavras que
# aparecem no mínimo 2 vezes e com o número certinho de cores
doc2vec_model = Doc2Vec(vector_size=200, min_count=2, workers=cores)
doc2vec_model.build_vocab(corpus_file="questions.txt")
doc2vec_model.train(corpus_file="questions.txt",
                    total_words=n_words,
                    epochs=100)
if not os.path.exists("models"):
    os.makedirs("models")
doc2vec_model.save('models/doc2vec.model')
# Salvando o modelo
!rm -rf /content/gdrive/My\ Drive/homer_chatbot/doc2vec.model
!cp /content/models/doc2vec.model /content/gdrive/My\
Drive/homer_chatbot/doc2vec.model
!cp /content/models/doc2vec.model.docvecs.vectors_docs.npy
/content/gdrive/My\
Drive/homer_chatbot/doc2vec.model.docvecs.vectors_docs.npy
!echo -e "\nsaved\n"
print('Feito!')
```


Para testar e ver o funcionamento do “Doc2Vec”, fazemos um “teste de sanidade” – analisando as palavras consideradas parecidas entre si.

```
# 'Sanity Test'
import warnings
# Um filtro de warnings chatos, pra não atrapalharem a apresentação...
with warnings.catch_warnings():
    warnings.simplefilter("ignore")
    # vetor com palavras mais parecidas
    print(doc2vec_model.wv.most_similar(['nice'])[0:2])
```

A função de chatbot é feita utilizando o método “most_similar” no vetor inferido de uma nova frase:

```
import warnings
with open("answers.txt") as f:
    answers = f.read().split("\n")
# A função de ChatBot em si
def chatbot_doc2vec():
    quit=False
    while quit == False:
        text = input('Me: ').lower()
        # Um comando de quit opcional
        if text == 'quit()':
            quit=True
        else:
            tokens = text.split()
            # Infere vetor para o texto que o modelo pode não ter visto
            new_vector = doc2vec_model.infer_vector(tokens)
            with warnings.catch_warnings():
                warnings.simplefilter("ignore")
                # 15248 é o último índice das linhas com respostas do
                # é até onde o modelo pode pegar frases para responder.
                index = doc2vec_model.docvecs.most_similar([new_vector], topn=1, clip_end=15248)
            # index é uma lista de tuplas (index no arquivo de treino, similaridade)
            print("Homer: ", answers[index[0][0]])
            print('\n')
```

5. Experimentação

Com o chatbot em mãos, testamos as capacidades do nosso chatbot com outras pessoas, para validar o quão bom ele é. Fizemos o seguinte: apresentamos as duas versões do chatbot, explicando a diferença entre elas, mas sem dizer qual personagem é, e pedimos para a pessoa tentar descobrir qual é. Afinal, se o único propósito deste robô é ser o Homer Simpson, vamos testar o quão Homer Simpson ele é.

Infelizmente na primeira experimentação, esquecemos informar a nossa colega que o nosso querido chatbot não se dá muito bem interpretando letras maiúsculas (o que o impede de interpretar bem quais palavras foram de fato ditas).

Todos os diálogos completos da parte de experimentação estão disponíveis no repositório do Github. ^[6]

5.1 Experimentação I – Bruna

5.1.1 “Seq2Seq”

Ficamos envergonhados com o que o Homer nos decide falar, já na primeira resposta – mas tudo bem, a nossa colega entendeu que ele não faz muito sentido, e conversando um pouco com esta primeira versão, ela viu que seria melhor tentar com a outra, que poderia fazer mais sentido.

```
Me: Hi
chatbot: ohh. so hottt.

Me: My cat is beautiful
chatbot: well, it's a cat -- I'm not sure it's the cat.

Me: My cat is cute
chatbot: well, it's a cat -- I'm not sure it's the cat.

Me: The eraser is blue
chatbot: all right! you caught us. what are you gonna do about it? throw us out of town again?
```

Figura 7. Primeira conversa com a Bruna.

5.1.2 “Doc2Doc”

Foi nesse momento que uma outra colega que estava assistindo teve a ideia de perguntar qual o nome dele, e ele respondeu de acordo (mas note a desconsideração da máquina para com sua nova amiga no final):

```
Me: What is your name?
chatbot: simpson.

Me: My cat is beautiful
chatbot: well, it's a cat -- I'm not sure it's the cat.

Me: The teacher is tall
chatbot: the moody blues!

Me: I am gonna die
chatbot: hey, that's great, now let's get to work! now, if "the flintstones
has taught us anything, it's that pelicans can be used to mix cement.
```

Figura 8. Segunda conversa com a Bruna.

5.2 Experimentação II (“Seq2Seq”) – Vicente

Literalmente na primeira resposta, o Homer já se entrega, gritando 'Bart!!' – isto que o nosso colega parecia superinteressado em conversar com o chatbot, e descobrir sua verdadeira identidade...

```
Me: hello there
chatbot: bart!
```

Figura 9. Conversa com o Vicente.

5.3 Experimentação III (“Seq2Seq”) – João

Ele parece gostar de chamar a pessoa com quem está interagindo de "Marge", o que faz sentido, pela alta porcentagem das falas do Homer que foram com a Marge.

```
Me: what are you doing?
chatbot: saving our marriage!

Me: whats happening in your marriage?
chatbot: waiting a good way to go, marge. i'll just turn the belt and the t

•••

Me: your nickname?
chatbot: oh, marge! don't you have to drive to the bathroom!
```

Figura 10. Primeira parte da conversa com o João.

Ele também parece não entender abreviações direito, respondendo melhor às frases mais gramaticamente corretas.

```
Me: whats your name?
chatbot: all the last guy's goin' housed.

•••

Me: what is your name?
chatbot: homer simpson.

Me: BOA
chatbot: exactly. up to hell!
```

Figura 11. Segunda parte da conversa com o João.

5.4 Experimentação IV – Bruno

Mais alguns momentos relevantes:

5.4.1 “Seq2Seq”

```
Me: what do you think about theology?
chatbot: i'llgetit.

Me: physics is fun?
chatbot: uh, now no. i'm sure you're holding out on me.

Me: is physics fun?
chatbot: uh, now death are complaining to me, and you stole the rest of my life.

Me: uh no
chatbot: all right then
```

Figura 12. Primeira conversa com o Bruno.

5.4.2 “Doc2Doc”

```
Me: what's your favourite sport?
chatbot: when a man and a woman love each other very much, they--
```

Figura 13. Segunda conversa com o Bruno.

6. Conclusão

Com os resultados no mínimo estranhos, concluímos que o conjunto de dados era pequeno demais para uma boa aprendizagem de contexto de palavras e as suas relações. Também concluímos que existem modelos de redes neurais que são mais adequados para diferentes conjuntos de dados e objetivos de processamento de linguagem natural.

Na experimentação, vimos que os chatbots, apesar de tudo, têm uma semelhança com o Homer Simpson, com uma personalidade parecida e de vez em quando mencionando outros personagens da série. Isso é óbvio no caso do modelo Doc2Vec, mas ficamos surpresos com estes resultados do modelo Seq2Seq.

Para fazer uma ligação com o que aprendemos na disciplinas, vamos comparar funcionamento de modelos de aprendizado de máquina, com expressões regulares e autômatos:

Autômatos são criados a partir de algoritmos, que, assim como expressões regulares, podem ser criados de maneira racional para prever contextos diferentes de um diálogo. Porém isto está limitado à imaginação e observação do programador, de identificar e prever programaticamente tais contextos.

Por outro lado, os modelos de aprendizado de máquina, como as redes neurais, são diferentes, pois aprendem de fato, munidos de seus parâmetros de aprendizado. Neles, o trabalho do programador (além de passar noites tentando entender como implementar) é apenas escolher quais são os parâmetros certos, aqueles em que potencialmente serão relevantes para o resultado final, e controlar o período de treinamento, para que o modelo treine pela quantidade certa de tempo.

7. Links Úteis

- Nosso repositório do Google Drive.^[1]
- Nosso repositório do Github.^[6]
- Conjunto de dados.^[2]
- Tutorial do “Tensorflow”.^[3]
- “Gensim”.^[5]

7.1 Seq2Seq

- Nossa referência para o modelo “Seq2Seq”.^[7]
- Página da Google do “Seq2Seq”.^[4]

7.2 “Doc2Vec”

- Nossa referência para o modelo “Doc2Vec”.^[8]
- A gentle Introduction to “Doc2Vec”.^[9]
- Tutorial de “Doc2Vec”.^[10]

8. Referências

- [1] https://drive.google.com/drive/folders/1cs3x5OpAML9lGeDbHjby7GSifKu0X5P_
- [2] https://www.kaggle.com/wcukierski/the-simpsons-by-the-data#simpsons_script_lines.csv
- [3] <https://github.com/tensorflow/nmt>
- [4] <https://google.github.io/seq2seq/>
- [5] <https://radimrehurek.com/gensim/tutorial.html>
- [6] <https://github.com/mswlandi/homer-chatbot>
- [7] <https://blog.kovalevskyi.com/how-to-create-a-chatbot-with-tf-seq2seq-for-free-e876ea99063c>
- [8] <https://towardsdatascience.com/how-to-build-an-easy-quick-and-essentially-useless-chatbot-using-your-own-text-messages-f2cb8b84c11d>
- [9] <https://medium.com/scaleabout/a-gentle-introduction-to-doc2vec-db3e8c0cce5e>
- [10] <https://github.com/RaRe-Technologies/gensim/blob/develop/docs/notebooks/doc2vec-lee.ipynb>