UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE RUSSAS

CURSO DE GRADUAÇÃO EM ENGENHARIA DE SOFTWARE

JEFERSON DA SILVA JULIANI

GERANDO LETRAS MUSICAIS UTILIZANDO UMA REDE NEURAL RECORRENTE LSTM - LONG SHORT-TERM MEMORY

RUSSAS

2019

JEFERSON DA SILVA JULIANI

GERANDO LETRAS MUSICAIS UTILIZANDO UMA REDE NEURAL RECORRENTE LSTM - LONG SHORT-TERM MEMORY

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do Campus de Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Orientador: Prof. Dr. Alexandre Matos Arruda

Coorientador: Prof. Dr. Francisco Nauber Bernardo Gois

RUSSAS

2019

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

J89g Juliani, Jeferson da Silva juliani.
Gerando letras musicais utilizado uma rede neural recorrente LSTM - Long Short-Term Memory / Jeferson da Silva juliani Juliani. – 2019.
47 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Russas, Curso de Engenharia de Software, Russas, 2019.
Orientação: Prof. Dr. Alexandre Matos Arruda.
Coorientação: Prof. Dr. Francisco Nauber Bernardo Gois.

1. geração de letras musicais. 2. rede neural recorrente. 3. LSTM. 4. aprendizagem profunda. I. Título.

CDD 005.1

JEFERSON DA SILVA JULIANI

GERANDO LETRAS MUSICAIS UTILIZANDO UMA REDE NEURAL RECORRENTE LSTM - LONG SHORT-TERM MEMORY

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do Campus de Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Alexandre Matos Arruda (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Francisco Nauber Bernardo Gois (Coorientador)
Universidade Federal do Ceará (UFC)

Prof. Ms. Daniel Márcio Batista de Siqueira
Universidade Federal do Ceará (UFC)

À minha mãe, seu cuidado e dedicação foi que deram, em alguns momentos, a esperança para seguir. Pai, por sempre acreditar em mim. Aos meus amigos que sempre estiveram comigo nessa jornada.

AGRADECIMENTOS

A Deus! Pois sem ele nada disso seria possível, eu nem mesmo estaria aqui. Obrigado por sempre ter me dado forças quando eu mais precisava, e colocar pessoas maravilhosas em minha vida, e ter me proporcionado grandes experiências. Sou grato por tudo.

A minha mãe, Geny Ribeiro, por sempre me apoiar nas minhas decisões e sempre me ajudar em tudo que precisei, sem você eu não conseguiria. Obrigado por tudo, te amo muito.

Ao meu pai, Rogerio Juliani, que sempre quis o melhor para mim, e sempre batalhou muito para isso, você é uma inspiração para mim. Obrigado por tudo, te amo muito.

A minha namorada, Leyde Dayane, por ser uma pessoa maravilhosa e luz nos momentos em que eu precisei, por me fazer sorrir quando estava desanimado, e me ajudar a ser uma pessoa melhor. Obrigado pelo seu amor e carinho, te amo muito.

A minha amiga/irmã, Gabrielle Marques, por me guiar em vários momentos em que eu não sabia o que fazer, por me ajudar a me desenvolver como pessoa e me incentivar a fazer coisas que eu não acreditava ser capaz. Obrigado pelas aventuras, por tudo, você estará sempre no meu coração, te amo.

Ao meu irmão que a UFC me deu, Thiago Oliveira, por ser uma pessoa que sempre me incentivou, sempre tentou me colocar pra cima e sempre acreditou em mim. Obrigado pelas aventuras, conversas, zoeiras, memes e viagens. Te amo meu amigo, obrigado.

A família que a UFC me deu, que são meus amigos Alexandre Arruda, Neto Guimarães, Gabrielle Marques, Suzana Karen, Maria Eduarda, Matheus Oliveira, Yan Vancelis, Thiago Oliveira, Abner Carvalho, Amanda Lima e Paulo Victor. Obrigado vocês são incríveis nunca vou me esquecer de vocês.

Agradeço a todos aos professores, Alexandre Arruda, Marília Mendes, Viviane Menezes, Felipe Maciel e Ana Beatriz, que contribuíram enormemente com a minha graduação, não só em conhecimento acadêmico, sou muito grato a vocês. Vocês fizeram parte da minha história, não irei me esquecer. Por fim, gostaria de agradecer a todos que contribuíram para a realização deste trabalho e para a minha graduação.

“Nunca subestime o poder de sua visão para mudar o seu mundo.”

(Caio Carneiro)

RESUMO

O uso de aprendizagem de máquina vem sendo utilizada para diversos fins, inclusive a geração de vários tipos de arte, sendo elas: visuais, sonoras ou textuais. O uso desses algoritmos de aprendizagem possibilita desenvolver aplicações de geração de linguagem natural. Os problemas de geração de linguagem natural utilizam em sua maioria redes neurais recorrentes, um assunto bem pesquisado, com muitas provas de conceitos e variações. No entanto, uma aplicação mais específica para este problema é a das letras das músicas. Neste trabalho, foi apresentado um método e discutidos os resultados da geração automática de letras musicais utilizando uma rede LSTM (Long Short-Term Memory). Para a criação dos datasets foi utilizado um crawler desenvolvido para este trabalho, que capturou 562 músicas de 3 artistas de diferentes estilos. Os experimentos foram realizados separadamente com cada estilo. Os resultados dos experimentos mostraram uma estrutura gramatical diferente com o corpus de RAP, que passou a rimar em algumas das sentenças geradas. Também foi possível observar a geração de algumas sentenças musicais aceitáveis.

Palavras-chave: LSTM. geração de letras musicais. aprendizagem profunda. rede neural recorrente

ABSTRACT

The use of machine learning has been used for various purposes, including the generation of various types of art, such as visual, sound or textual. The use of these learning algorithms makes it possible to develop natural language generation applications. Problems of natural language generation mostly use recurrent neural networks, a well-researched subject, with much proof of concepts and variations. However, a more specific application for this problem is song lyrics. In this work, a method was presented and the results of the automatic generation of lyrics using a Long Short-Term Memory (LSTM) network were discussed. For the creation of the datasets a crawler developed for this work was used, that captured 562 songs of 3 artists of different styles. The experiments were performed separately with each style. The results of the experiments showed a different grammatical structure with the RAP corpus, which began to rhyme in some of the generated sentences. It was also possible to observe the generation of some acceptable musical sentences.

Keywords: LSTM. Song-lyrics generation. deep learning. recurrent neural networks

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CNN	rede neural convolucional
HTML	Linguagem de Marcação de Hipertexto
LSTM	Unidades de Memória de Longo-Curto Prazo
RNA	redes neurais artificiais
RNN	redes neurais recorrentes

SUMÁRIO

1 INTRODUÇÃO

As redes neurais artificiais (RNA) são algoritmos que se baseiam nos processos biológicos do cérebro, com o intuito de resolver determinados problemas. Com o avanço da tecnologia e o aumento do poder computacional, essas redes vêm sendo utilizadas cada vez mais para resolver problemas diversos.

As redes neurais recorrentes são um tipo de RNA projetada para reconhecer padrões em sequências de dados, como texto, bolsa de valores, dados climáticos, DNA, dentre outros. Essas redes possuem loops que permitem que as informações persistam. O fato delas possuírem um certo tipo de memória oferece uma vantagem sobre as redes neurais tradicionais, porém existe um problema associado a essas redes, conhecido como *Vannishing Gradient*. Para resolver esse problema, esse trabalho utilizou um tipo de redes neurais recorrentes (RNN), chamado de Unidades de Memória de Longo-Curto Prazo (LSTM).

As redes LSTM têm um estado adicional que através do qual faz ajustes no fluxo das informações que são passadas de uma rede para outra. Esse estado tem uma vantagem de poder esquecer ou lembrar as informações mais seletivamente. Essas redes vem obtendo sucesso em muitas aplicações, como tradução de textos (DOğAN *et al.*, 2019), geração de textos e poemas (LUO; HUANG, 2017; Yizhe Zhang, Zhe Gan, 2016), geração de notas músicas (CHOI *et al.*, 2016), dentre outras.

Quando se trabalha com aplicações de geração de linguagem natural, é fundamental entender as estruturas das sentenças, e sua gramática. Uma das características relevantes ao realizar essas gerações é a criatividade, o que não é uma tarefa tão simples para as máquinas (BODEN, 2014).

Uma aplicação mais específica para os problemas de geração de linguagem natural utilizando-se destas redes neurais, são a de geração de letras musicais. Pois ao se compor uma música, o artista leva em consideração diversos fatores. Esses fatores, podem variar de acordo com o gênero, à época, cultura, dentre outros aspectos. Muitas das vezes, os artistas querem passar um sentimento, uma sensação, ou até mesmo uma crítica em suas composições. Portanto, o processo de compor a melodia e letra de uma música não é uma tarefa simples. Tarefa esta, que até então, era considerada inerente aos seres humanos.

Neste trabalho, procurou-se entender melhor como uma rede LSTM pode agir em um processo criativo de criacriação de letras musicais. Além de criar um modelo que possa criar letras inspiradas em obras de artistas de diversos gêneros, utilizando redes neurais LSTM.

1.1 Objetivos

1.1.1 Objetivo geral

Desenvolver um modelo capaz de gerar letras musicais de diferentes estilos, utilizando redes LSTM e analisar os resultados gerados.

1.1.2 Objetivo específico

- Obter Corpora Criar para treinamento da rede.
- Desenvolver um protótipo para realização de testes.
- Desenvolver uma estrutura para a geração de letras musicais.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo é apresentada fundamentação teórica sobre os tópicos envolvidos neste trabalho, sendo eles: modelos sequenciais, redes neurais artificiais, redes neurais recorrentes, unidades de memória de longo-curto prazo e Crawler. Estes tópicos estão descritos abaixo.

2.1 Modelos Sequenciais

Segundo (HAJ-YAHIA, 2018) sequências são uma estrutura de dados em que cada exemplo pode ser visto como uma série de pontos de dados. Por exemplo, na frase: "O inverno está chegando"é um exemplo em que várias palavras dependem umas das outras para que haja sentido. Modelos sequenciais são utilizados atualmente para resolver diversos problemas. Segundo (BROWNLEE, 2017) alguns desses problemas são:

• Previsão do mercado de ações: dada uma entrada com o comportamento das ações ao longo do tempo, a sua saída será uma previsão do comportamento nos próximos dias.

• Classificação de sequência de DNA: dada uma sequência de DNA de valores A, C, G e T, preveja se a sequência é para uma região codificante ou não codificante.

• Análise de sentimentos: dada uma sequência de texto como entrada, preveja se o sentimento do texto é positivo ou negativo.

• Geração de texto: dado um corpus de texto, gere como saída parágrafos baseadas no corpus de entrada.

• Geração de música: dado um corpus com exemplos de notas musicais, gere uma sequência de notas musicais que tenham as propriedades do corpus.

• Geração de legenda para imagens: dado uma imagem como entrada, gere uma sequência de palavras que descrevem a imagem.

• Tradução automática de textos: dado um texto em uma linguagem, converta para uma outra linguagem.

A estrutura utilizada nesse projeto, que veremos adiante, utiliza um modelo sequencial para o processamento de dados sequenciais.

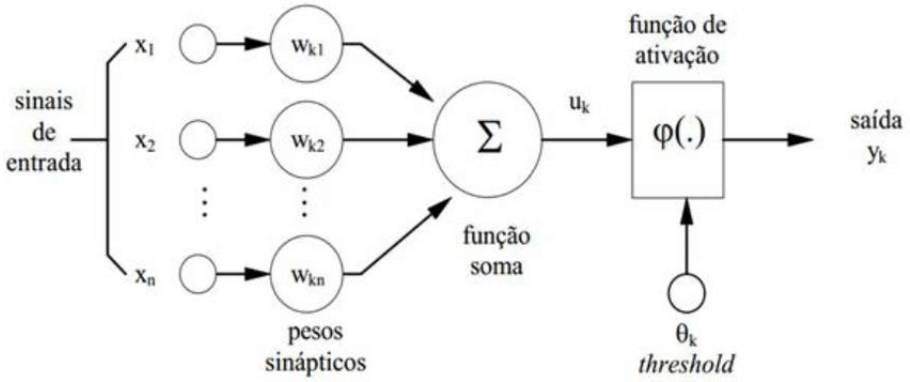
2.2 Redes Neurais Artificiais

As RNA tinham como objetivo originalmente, criar sistemas computacionais capazes de resolver determinados problemas, baseando-se nos processos biológicos do cérebro

(GURNEY, 1997) . Atualmente, vem sendo desenvolvidas para tarefas específicas, como reconhecimento de fala, visão computacional, diagnóstico médico, dentre outras. Existem diferentes tipos de RNA, porém o foco deste trabalho será com as RNN.

RNN são basicamente algoritmos que são capazes de reconhecer padrões. Esses algoritmos formam uma rede que possui um conjunto de elementos ou nós. Quando conectados, são semelhantes aos neurônios humanos. Esses algoritmos interpretam os dados que estão em uma forma numérica. Estes dados, são previamente contidos em vetores, que foram convertidos a partir de dados do mundo real, como: imagens; sons; textos; ou séries temporais.

Figura 1 – Modelo de um neurônio artificial



Fonte: (HAYKIN, 2001)

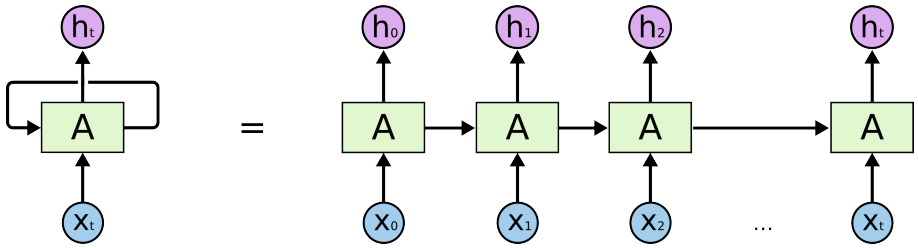
A Figura 1 é uma representação de um modelo de um neurônio artificial. Os sinais de entrada *{x1, x2, ..., xn}* são os dados que servem de entrada para a rede. Esses sinais são ponderados em pesos sinápticos *{wk1, wk2, wk3, ..., wkn}*, e depois multiplicados antes de serem enviadas para o corpo da célula. Segundo (GURNEY, 1997) esses sinais são somados por uma adição aritmética simples, para fornecer uma ativação ao nó. Esse potencial de ativação Uk é enviado para uma função de ativação e em seguida é realizada a saída Yk.

2.3 Redes Neurais Recorrentes

As RNN são um tipo de RNA que são específicas para problemas de dados sequenciais. Uma RNN pode ser imaginada como múltiplas cópias da mesma rede, cada uma passando uma mensagem a um sucessor (OLAH, 2015). RNNs possuem um mecanismo de looping que atua como um caminho que permite que as informações fluam de uma rede para um outra.

A Figura 2 demonstra o funcionamento deste mecanismo de looping. A entrada está sendo representada por X_t , a rede neural por A, e o valor de saída por h_t. A ideia de conectar

Figura 2 – Desdobrar de uma rede neural recorrente



Fonte: (OLAH, 2015)

informações anteriores, tem por objetivo compreender o quadro atual baseado nos anteriores. Quando isso ocorre, há uma relação entre duas entradas sequenciais. Porém existe um problema quando um intervalo entre as entradas é muito grande. Considere, por exemplo uma entrada do tipo texto:

"Comprei as passagens, porém tenho medo de voar de..."

A rede pode adivinhar que a próxima palavra que complete a frase é "avião", pois o intervalo é pequeno. Porém na seguinte frase:

"Ao chegar na Rússia, Suzana percebeu que não entendia uma palavra que os russos diziam. Já o amigo que estava com ela conseguia entender. Ele é fluente em ..."

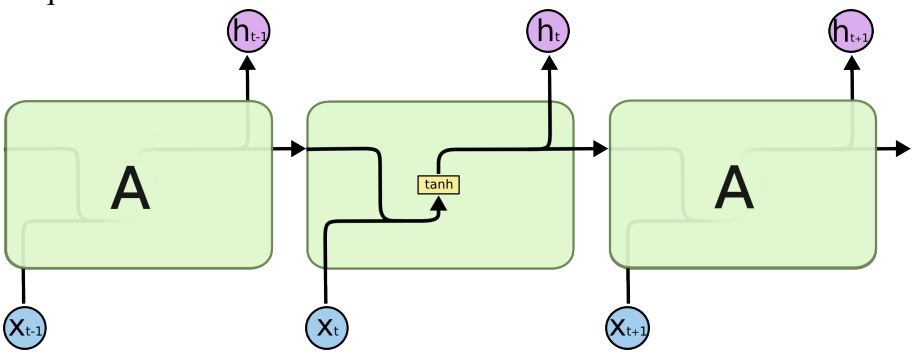
Na frase acima, a rede poderia ter mais dificuldades para inferir a palavra que complete a frase. Por mais que ela considere que a próxima palavra é um idioma, a rede precisa levar em consideração as informações passadas. As palavras "Rússia" e "russos" foram referenciadas no texto, porém com o crescimento do intervalo, fica difícil acessar essas informações que estão em passados muito distantes. Em teoria essas redes são capazes de lidar com dependências de sequências longas. Porém (Bengio *et al.*, 1994), mostraram através de experimentos, que na prática isso muitas vezes não é possível.

2.4 Unidades de Memória de Longo-Curto Prazo (LSTM)

A ideia inteligente de introduzir *auto-loops* para produzir caminhos nos quais o gradiente pode fluir por longos períodos é uma contribuição central do modelo inicial de memória de longo prazo (HOCHREITER; SCHMIDHUBER, 1997). Como mostrado na Figura 3, logo abaixo, o módulo de repetição em uma RNN padrão contém apenas uma única camada de rede neural, representada na caixa amarela.

Uma rede LSTM, representada na Figura 4, contem um fluxo de controle similar a uma RNN. A diferença entre elas esta nas operações com as celulas, já que o módulo de repetição em uma LSTM contém quatro camadas de interação. Essas operações são usadas para permitir

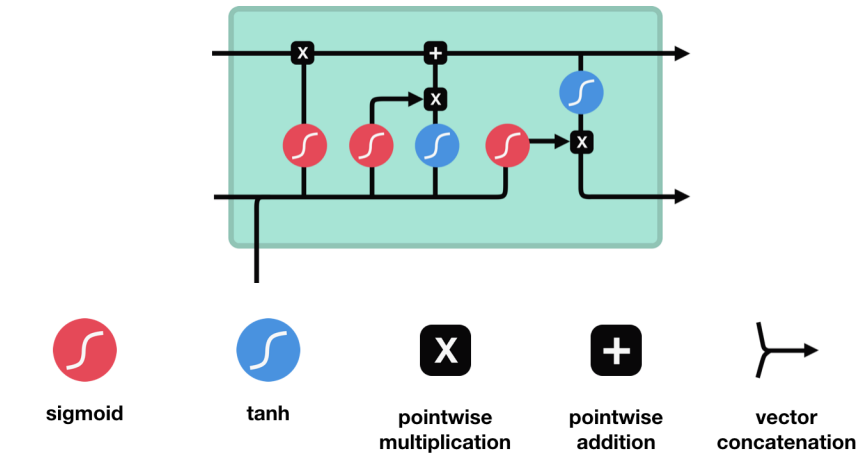
Figura 3 – Arquitetura básica de uma rede neural recorrente



Fonte: (OLAH, 2015)

que a rede mantenha ou esqueça informações.

Figura 4 – Arquitetura básica de uma rede LSTM



Fonte: (NGUYEN, 2018)

A ativação tanh, representada na Figura 4, é usada para ajudar a regular os valores que fluem pela rede entre -1 e 1. As RNN utilizam essa ativação para gastar menos recursos computacionais.

Uma ativação sigmoid, representada em vermelho na Figura 4, também é utilizada para regular os valores. Diferentemente da ativação tanh, ela regula os valores entre 0 e 1. Isso faz com que os valores possam ser esquecidos ou mantidos, visto que, todo número multiplicado por 0 é 0, e multiplicado por 1 é o mesmo valor, portanto são lembrados.

Primeiramente, na célula LSTM, nós temos o portão do esquecimento que utiliza justamente uma ativação sigmoid. Logo após, o portão do esquecimento, temos o portão de entrada. Este portão é responsável por atualizar o estado da célula, recebendo o estado oculto anterior e a entrada atual. Esses dados recebidos são passados para uma ativação sigmoid e tanh.

Em seguida temos já informações suficientes para calcular o estado da célula. Basta multiplicar o vetor que passou pelo portão do esquecimento, pelo estado atual da célula e então somar a saída do portão de entrada.

No final, temos o portão de saída, que decide qual será o próximo estado oculto. Primeiro é passado o estado oculto anterior mais a entrada atual em uma função de sigmoid e passado o estado atual que foi modificado para uma função tanh. Esses vetores são multiplicados, e o resultado da operação é o novo estado oculto.

2.5 Crawler

Crawlers são softwares desenvolvidos com o objetivo de realizar uma varredura sistemática, vista a informação na qual se busca. O processo que um crawler executa é chamado de Web Crawling ou Spidering. Segundo (OLSTON; NAJORK, 2010), esses softwares são conhecidos também como bot(robôs) ou spider, e são usados para uma variedade de propósitos. Dentre alguns dos propósitos, podemos citar:

• Motores de busca (Search Engine), que são utilizados para auxiliar a procura das informações armazenadas em uma rede, seja ela global ou não. Uma aplicação famosa que utiliza esses motores de busca é o Google;

• Mineração de dados, onde os dados coletados (textos, imagens, números, etc) são armazenados para que sejam posteriormente processados e analisados;

3 TRABALHOS RELACIONADOS

Nesta seção são apresentados alguns estudos que aplicam as redes LSTM para a geração de dados. A busca dos artigos foi realizada entre 10/03/2019 e 18/05/2019 em duas das principais bibliotecas digitais de artigos científicos na área de computação: ACM Digital Library e IEEExplore. Essas duas bases de dados foram escolhidas devido a quantidade de trabalhos científicos na área.

Uma string de busca foi utilizada para buscar os trabalhos em ambas bibliotecas, sendo ela: (LSTM AND Generation) AND (Text OR Music OR Dialogue OR Lyrics). O resultado desta busca obteve 72 trabalhos das duas bibliotecas, o período das publicações ficou entre 2015 – 2019. Foram selecionados os trabalhos que tinham mais semelhança com este trabalho, visto que a maior parte deles se tratava apenas de geração de texto em geral.

Em (Yizhe Zhang, Zhe Gan, 2016) foi proposto uma estrutura genérica que emprega uma rede LSTM e o sistema de rede neural convolucional (CNN) para a geração de sentenças realistas. Foram utilizados, dois datasets para o treinamento, sendo que um continha sentenças formais de livros e o outro sentenças diversas de um determinado site. O treinamento continha 1 milhão de sentenças de ambos datasets. O gerador e o discriminador tiveram um treinamento de forma iterativa. Dado que o gerador de LSTM tipicamente envolve mais parâmetros e é mais difícil de treinar do que o discriminador de CNN, foi executada então, as etapas de otimização para o discriminador a cada 5 passos do gerador. Como resultado eles observaram que o discriminador poderia distinguir as sentenças sintéticas das reais, com uma probabilidade de 0,08. Foi observado que as sentenças geradas possuíam uma gramática razoável e apresentavam palavras adequadas, porém provavelmente a CNN conseguia identificar certos nuances nas sentenças. Neste trabalho, diferentemente de (Yizhe Zhang, Zhe Gan, 2016), será aplicado uma rede LSTM para datasets menores e de gêneros musicais diferentes, para analisar como ela se comporta no processo criativo de geração.

Em (SON; LEE, 2019) foi implementado um modelo que utiliza uma rede neural LSTM básica para criar letras de músicas que se encaixasse no contexto de músicas coreanas K-pop. Neste trabalho concentrou-se na geração de letras musicais baseadas no predicado, devido as características do coreano, como língua aglutinante . Devido a essas características, toda vez que uma barra de letras foi criada, gerou-se uma barra a partir do predicado, que é invertido e colocado na frente da sentença, para que o modelo possa aprender. Para o treinamento, foi utilizado um dataset contendo letras de balada, coletadas 10.000 músicas por um web crawler, em

um site de músicas chamado 'MELÃO'. Em suas primeiras tentativas de treinamento, ocorreram overfitting, as músicas geradas estavam idênticas aos dados de treinamento. Visto isso, concluiu-se que o tamanho de incorporação de um caractere era muito grande, e a camada da rede LSTM foi mais profunda do que o necessário. Portanto, os hiper-parâmetros foram alterados. Para avaliar se o texto gerado era diferente das entradas, foi feita uma avaliação humana e usado uma pontuação BLEU (bilingual evaluation understudy). Como resultado, observou-se que o modelo de inversão gera uma barra correspondente ao valor de entrada de uma determinada música, e gera um resultado semelhante à estrutura das letras de entrada originais, porém com uma semântica diferente.

No trabalho de (LUO; HUANG, 2017) foi utilizado um modelo LSTM Encoder-Decoder para a geração de poemas clássicos chineses. O corpus utilizado foi coletado da dinastia Tang, contendo 74.474 quadras. Para os experimentos foram utilizadas 2.000 quadras como conjunto de validação e o restante para treinamento. Durante o treinamento, foram selecionados os 8000 caracteres mais frequentes como vocabulário. Para avaliar os poemas gerados, foram levados em consideração dois fatores: naturalidade do texto gerado; e a taxa de incorporação. Esses fatores foram avaliados entre diferentes algoritmos de esteganografia. Visto que a avaliação de poesias feitas por máquinas é um problema, foi utilizado o Teste de Turing para avaliar o modelo. Uma pessoa julgou se esses poemas foram criados por uma máquina, ou escritos por um poeta. Como resultado, observou-se que o modelo fracassou no teste, e apesar produzir poemas coerentes e semanticamente consistentes, aplicá-lo diretamente à esteganografia não funciona muito bem.

Este trabalho se difere dos demais pelo fato de aplicar uma rede LSTM para a geração de letras musicais, de diferentes gêneros e na língua portuguesa. A Tabela 1 abaixo traz algumas informações que mostram a diferença entre os trabalhos citados.

Tabela 1 – Diferença entre os trabalhos relacionados

Trabalho	Tipo de dados gerados	Rede neural	Idioma
(Yizhe Zhang, Zhe Gan, 2016)	Texto	LSTM + CNN	inglês
(SON; LEE, 2019)	Letra musical	LSTM	Coreano
(LUO; HUANG, 2017)	Poema	LSTM	Chinês
Este trabalho	Letra musical	LSTM	Português

Fonte: o autor.

4 METODOLOGIA

Neste trabalho, foram realizadas algumas etapas, dentre elas: desenvolvimento do crawler; criação da corpora; criação do protótipo inicial; e análise dos experimentos iniciais. As etapas descritas acima serão melhor apresentadas abaixo.

4.1 Desenvolvimento do Crawler

O desenvolvimento do crawler foi necessário, pois não há corpus em português para letras musicais disponíveis. Para o desenvolvimento, utilizou-se a linguagem de JavaScript e um interpretador de código Node.js. O algoritmo foi dividido em duas partes, são elas: captura dos links e captura das letras. O algoritmo encontra-se no Apêndice A.

A primeira parte de captura dos links, é responsável por capturar todos links de um determinado cantor. Primeiramente, o algoritmo necessita de uma entrada com um link da página, onde contém todos os links das músicas. Um exemplo de link utilizado seria: “https://letras.mus.br/racionais-mcs”. Após varrer toda a página e ler os links, o algoritmo exibe os links, e estes são salvos em um arquivo.

A segunda parte de captura das letras, é responsável por capturar toda a música da página. A cada link visitado, o algoritmo salva as letras em um arquivo de texto.

4.2 Criação da Corpora

O algoritmo de captura das letras musicais, é responsável apenas por salvar as letras. Após essa etapa, é necessário fazer uma leitura manual em todo o arquivo, para verificação de possíveis erros.

Cada página, pode ter uma formatação diferente na estrutura, o que faz com que o arquivo de texto, fique formatado de diferentes formas. No caso do site letras.mus.br , algumas das páginas capturadas, apresentavam letras com palavras unidas. Visto isto, foi necessário percorrer todo o arquivo, em busca dessas páginas. Após identificada a letra da música contendo essa formatação incorreta, as palavras foram separadas de forma manual.

Neste trabalho, foram criados três corpus para testes dos protótipos criados. O primeiro, foi criado a partir das músicas da banda Racionais MCs. Este primeiro corpus criado contém 116 músicas e 66.933 palavras. O Segundo, foi a partir da banda Henrique e Juliano, e contem 206 músicas e 33.186 palavras. O terceiro, foi criado a partir do grupo músical Raça

Negra, e contem 240 músicas e 27.422 palavras.

4.3 Criação de um protótipo inicial

Para a criação do protótipo inicial, utilizou-se a linguagem Python juntamente da API do Keras e seu modelo de geração de textos. O algoritmo realiza um pré-processamento dos dados, treina a rede neural e gera as sentenças. O algoritmo encontra-se no Apêndice B.

Primeiramente o corpus é carregado e passa por um processo de limpeza, onde são removidas algumas pontuações e espaços em branco. Em seguida é feito um mapeamento dos caracteres para inteiros. Após o mapeamento, são criadas subsequências do texto, e elas são vetorizadas para os dados de treino e teste.

Logo após ser feito o tratamento com os dados, é definida a rede neural e ela passa por um processo de treinamento.Após esse treinamento é possível gerar as sentenças por meio de uma função. Essa função gera as sentenças com uma semente aleatória do corpus, e é definido um tamando para a saída. Os hiper-parâmetros podem ser vistos na Tabela 2.

Tabela 2 – Hiper-parâmetros da rede neural

Tamanho do Batch	60
Épocas	60
Camadas LSTM	1
Unidades de Memória	128
Tamanho das sequências	40

Fonte: o autor.

4.4 Criação do Segundo protótipo

Visto a necessidade de melhoria do primeiro protótipo, foi proposto um novo algoritmo. Este se encontra no Apêndice C, e apresenta uma característica diferente na forma de criar os tokens. O algoritmo funciona da seguinte forma:

Primeiramente o corpus passa por um processo de limpeza, onde são removidas as pontuações e letras minúsculas nas frases;

Logo após é preciso que haja uma tokenização para que seja possível prever a próxima palavra(token). Esse processo extrai as palavras do corpus e as transformam em tokens. Para esse pré-processamento do texto, foi utilizado um modelo da própria biblioteca Keras chamado Tokenizer. Ao final desse processo, temos todos as palavras em tokens, que pode ser

Tabela 3 – Exemplo de tokens criados algoritmo 2

Ngram	**Sequência de tokens**
Essa barra	[200, 30]
Essa barra que	[200, 30, 150]
Essa barra que é	[200, 30, 150, 56]
Essa barra que é gostar	[200, 30, 150, 56, 803]
Essa barra que é gostar de	[200, 30, 150, 56, 803, 22]
Essa barra que é gostar de você	[200, 30, 150, 56, 803, 22, 48]

Fonte: o autor.

Tabela 4 – Exemplo de preditores e rótulos

Preditores	**Rótulos**
Então	me
Então me	ajude
Então me ajude	a
Então me ajude a	segurar

Fonte: o autor.

visto melhor na Tabela 3;

Depois de gerar todos os tokens, as sequências foram preenchidas e tiveram seus comprimentos igualados. Para isso foi utilizado a função "pad_sequence"do Keras. E antes de passar para o processo de treinamento do modelo de aprendizagem, foi criado os preditores(predictors) e rótulos(labels). Um exemplo desse processo pode ser visto meslhor na Tabela 4;

Ao final, o modelo é criado e os dados são passados para o modelo para que seja feito o treinamento. Os hiper-parâmetros podem ser vistos na Tabela 5. Ao finalizar esse processo é possível gerar as sentenças. Ao contrário do primeiro protótipo que gerava uma sentença por completa, no segundo foi utilizado uma abordagem diferente. Uma semente é passada para a função, e a partir dela é gerada uma pequena sentença, então essa sentença gerada passava a ser semente da próxima chamada da função.

Tabela 5 – Hiper-parâmetros da rede neural do segundo protótipo

Tamanho do Batch	60
Épocas	60
Camadas LSTM	1
Unidades de Memória	256
Tamanho das sequências	40

Fonte: o autor.

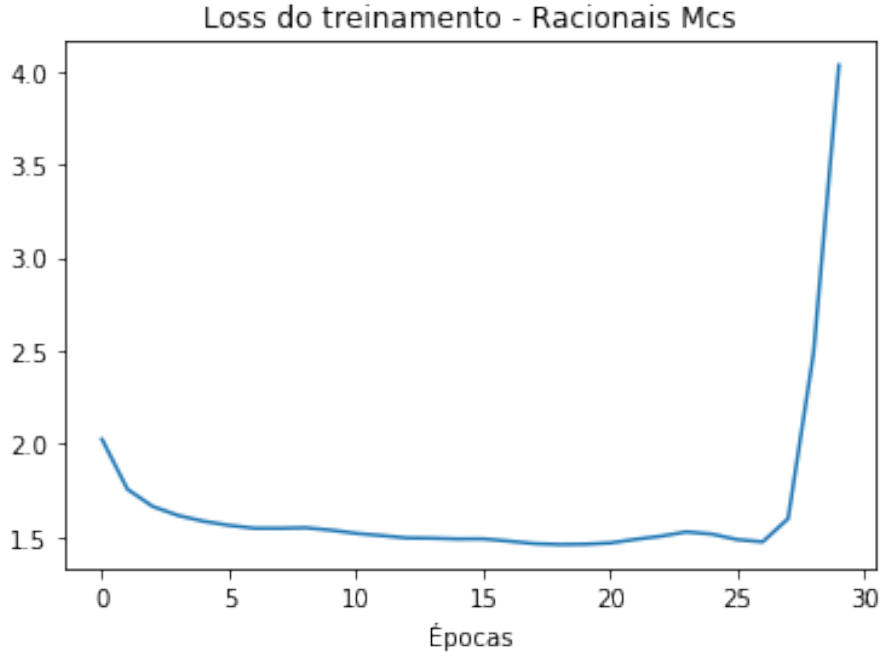
5 RESULTADOS

Os resultados que serão apresentados nas próximas sessões, foram obtidos utilizando os dois protótipos. Para se chegar nesses resultados foram feitas varias execuções do algoritmo, com diferentes hiper-parâmetros, afim de se chegar em um resultado que fosse considerado mais aceitável, e que houvesse mais sentido nas sentenças geradas.

5.1 Primeiro protótipo

Como resultado dos experimentos do primeiro protótipo, pode-se observar uma diferença no treinamento da mesma rede neural para diferentes corpuss. O corpus de músicas de RAP apresentaram um crescimento do Loss (perda) a partir da época 25. O menor valor foi atingido na época 19 com um Loss de 1,457. Os hiper-parâmetros utilizados são os da Tabela 2. Um gráfico contendo o crescimento pode ser visto na Figura 5. Algumas sentenças foram geradas a partir deste treinamento, elas podem ser vistas na Tabela 6.

Figura 5 – Gráfico de loss do treinamento de RAP



Fonte: o autor

Ao final do treinamento com o corpus de Raça Negra, obteve-se um Loss de 0.83 na época 60. O mesmos hiper-parâmetros presentes na Tabela 2 foram utilizados. Um gráfico apresentado na Figura 6 demonstra o Loss durante as épocas. A Tabela apresenta algumas das sentenças que foram geradas.

Tabela 6 – Sentenças geradas com a rede neural - RAP

corpus	**Sentenças geradas**
Racionais MCs (RAP)	o vendedor de pedras la na zona sul era o mundo e sera mais e o mal ai jao e a mao de cara e a vida
	quem nao quer brilhar quem nao mostra que e a mais da manha aqui e o que e nao e glonte a porta
	não é capaz ele nasce cresce e o que acontece a minha cara é pra acreditar e tem que ser e va
	ha aquele campo olha olha quanta gente em qual mentira vou acreditar o que vou encontrar

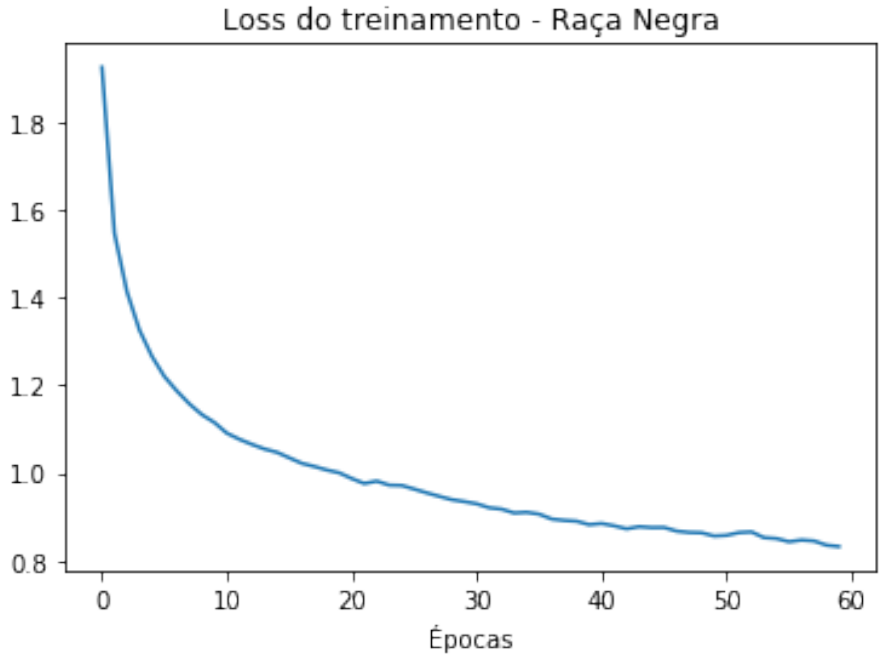
Fonte: o autor.

Tabela 7 – Sentenças geradas com a rede neural - Samba

corpus	**Sentenças geradas**
Raça Negra (Samba)	sou teu te adoro voce sabe que eu gosto de dizer que eu sou um meu coracao esta dificil de voce
	o fogo do meu coracao e fica so jogando vou me apaguendo e meu coracao nao vivo sonhar e amor
	com voce so com voce to a fim de amor com voce vem melhor de voce vou telefova o que eu tenho medo
	so juramos amor de verdade nao brinque com voce vem melhor de voce pra mim diana voce minha vida

Fonte: o autor.

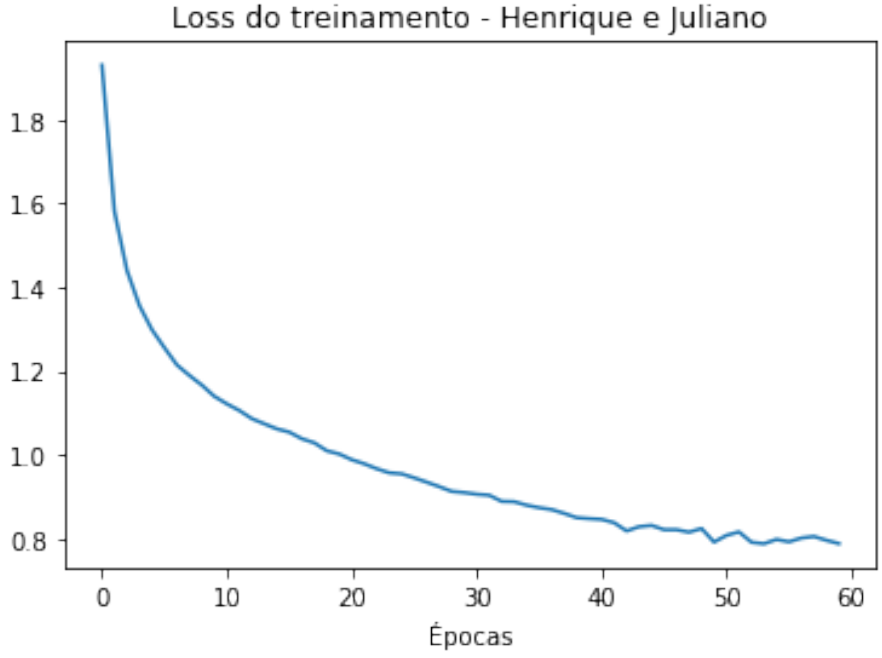
Figura 6 – Gráfico de loss do treinamento de Samba



Fonte: o autor

O corpus de músicas de sertanejo, utilizou os hiper-parâmetros da Tabela 2, e apresentou nos testes um crescimento do Loss a partir da época 60. O menor valor foi atingido na época 54 com um Loss de 0.786. Um gráfico contendo o crescimento pode ser visto na Figura 7. As sentenças geradas a partir deste treinamento, podem ser visualizadas na Tabela 8.

Figura 7 – Gráfico de loss do treinamento de sertanejo



Fonte: o autor

Tabela 8 – Sentenças geradas com a rede neural - sertanejo

corpus	**Sentenças geradas**
Henrique e Juliano (Sertanejo)	e confesso ta dificil de esquecer eu te amo demais e so tem murtorendo tempo te esquecer amor
	lembro de nos dois meu coracao sempre chama de verdade e o meu chapéu pro luar pra ver se eu nao so
	ei de novo falei desse amor que e lindo que eu te amo demais e o que eu nao disfarca de nada de nada
	dias passam devagar eu vou andando sem voce eu termina mas e so tem mudicao tem um amor de lado

Fonte: o autor.

5.2 Segundo protótipo

O segundo protótipo, foi testado com os corpuss de Raça negra, Henrique e Juliano e de Racionais MCs. Esses corpuss foram escolhidos por conter uma característica bem diferente na estrutura das sentenças, pois apesar de samba (Raça negra) e sertanejo (Henrique e Juliano) serem um pouco semelhantes, ambas divergem do estilo de RAP (Racionais Mc's). Além dessa característica, um outro fator é o vocabulário utilizado nas composições, por se tratar de Rap, Samba e Sertanejo. Diferentemente do outro protótipo, as sentenças geradas neste apresentam palavras que existem, e não há palavras que a rede neural tentou criar. Os resultados apresentados foram bastante satisfatórios, levando em consideração as estruturas das frases.

Durante a execução do corpus de Raça Negra, foram utilizados o hiper-parâmetros da Tabela 5. O menor Loss foi foi obtido na época 60 com o valor de 1.23 e pode ser melhor visualizado na Figura 8. As sentenças geradas a partir deste treinamento, podem ser visualizadas na Tabela 9.

Durante a execução do corpus de Henrique e Juliano, foram utilizados os mesmos hiper-parâmetros da Tabela 5, que também foram usados no treinamento do corpus de Raça Negra. O menor Loss foi foi obtido na época 60 com o valor de 0.94 e pode ser melhor visualizado na Figura 9. As sentenças geradas a partir deste treinamento, podem ser visualizadas na Tabela 10.

Tabela 9 – Sentenças geradas com a rede neural - Samba 2

corpus	**Sentenças geradas**
Raça Negra (Samba)	Amo Ter Voce Nao Quero Mais Alguem Dela E Que Eu Vou Chorar Sofrer Te Chama Demais
	Voce E Perfeita O Meu Vicio Todo Dia Eu Te Conquisto Mais Pra Chorar Chorar Chorar 2X
	Estou Perto De Voce Voltar Pra Mim Se Vai Se Magoar Esquecer De Mim Eu Nao Quer
	Nao Vou Mais Ter Tua Voz Me Cansei De Te Amar So Te Amar So Me Deixar Dia Meu

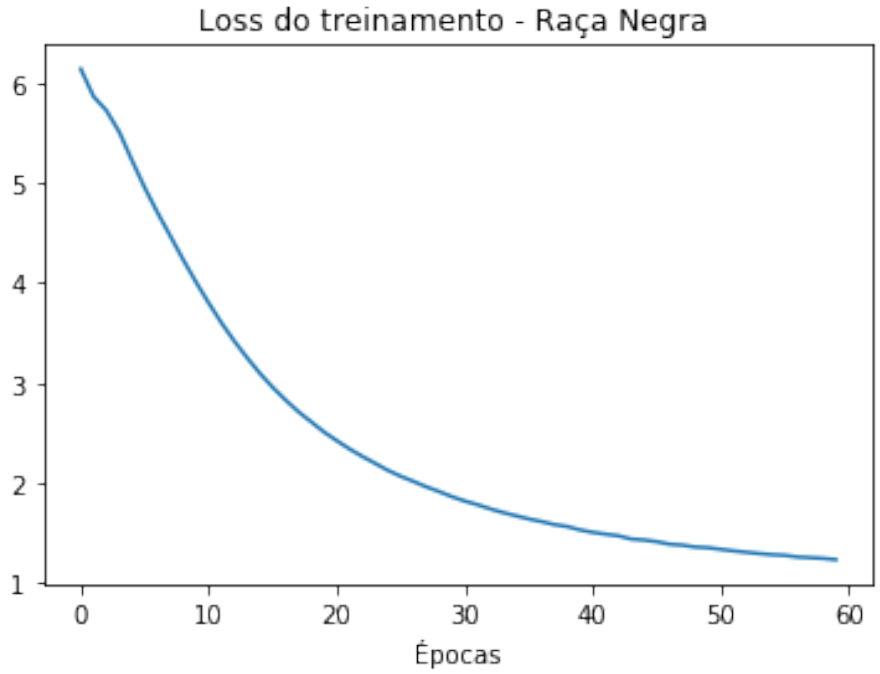
Fonte: o autor.

Tabela 10 – Sentenças geradas com a rede neural - Sertanejo 2

corpus	**Sentenças geradas**
Henrique e Juliano (Sertanejo)	Eu Te Quero De Qualquer Jeito E Ela Vai Demorar Pra Mim Que Eu Quero Viver Disso
	Alcool Em Mim Deixa Eu Te Mostrar Meu Amor Quem Eu Sou Ai Ai Ai Ai
	Hoje Eu Sei Que Achei Quem Eu Tanto Procurava Meu Amor E Voce Diz Voce Vem Vem
	Vamos Viver O Meu Lado Eu So Inteiro Bobo Vendo Meu Amor De Mim Faz Pouco De

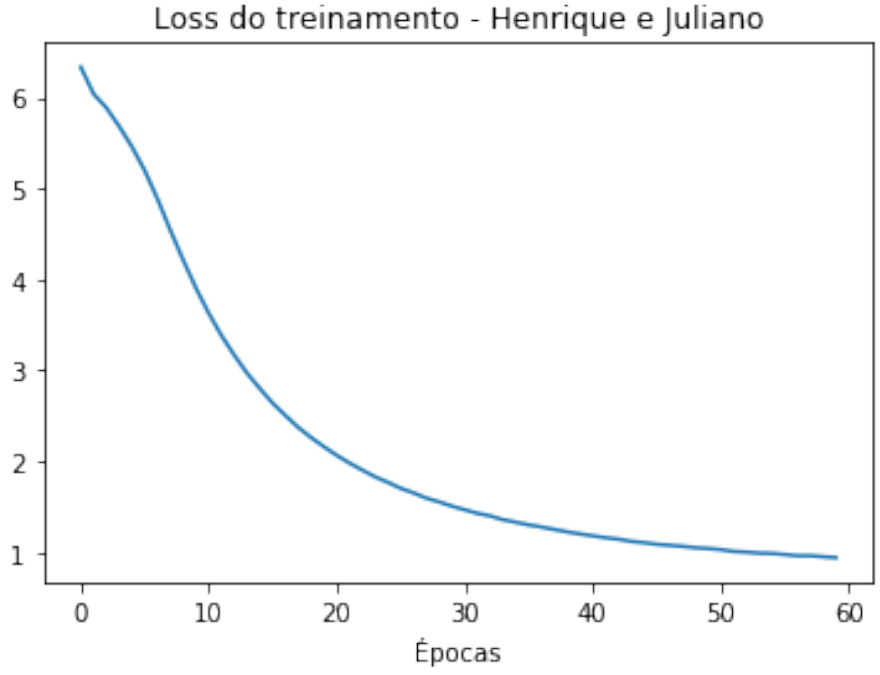
Fonte: o autor

Figura 8 – Gráfico de loss do treinamento de Samba 2



Fonte: o autor

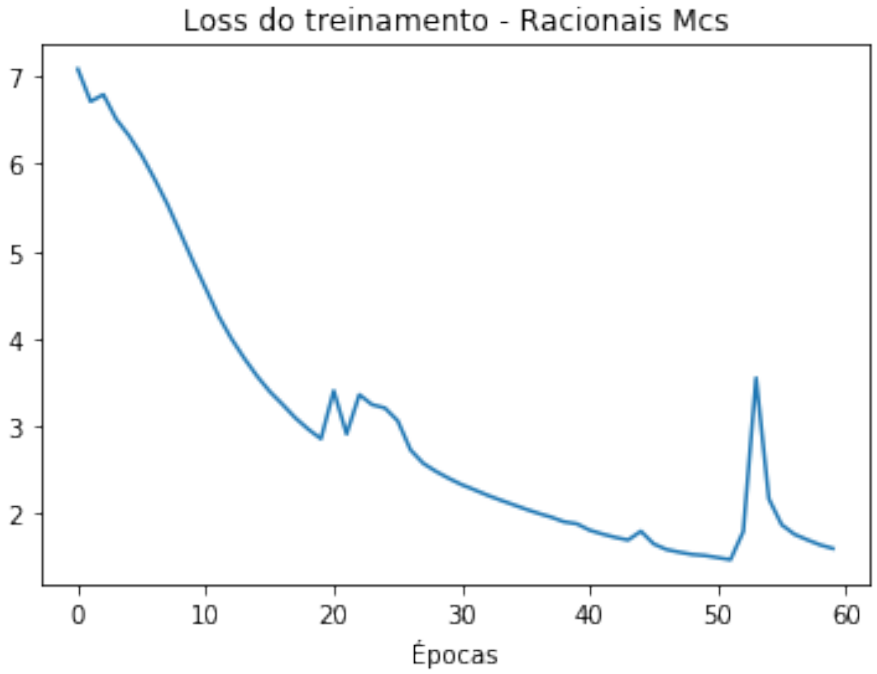
Figura 9 – Gráfico de loss do treinamento de Sertanejo 2



Fonte: o autor

Durante a execução do corpus de Racionais MC's, foram utilizados os mesmos hiper-parâmetros da Tabela 5. O menor Loss foi obtido antes de finalizar o treinamento, na época 51 e com o valor de 1.47 e pode ser melhor visualizado na Figura 10. As sentenças que foram geradas, podem ser visualizadas na Tabela 11.

Figura 10 – Gráfico de loss do treinamento de RAP 2



Fonte: o autor

Tabela 11 – Sentenças geradas com a rede neural - RAP 2

Corpus	**Sentenças geradas**
Racionais MCs (RAP)	Olho Por Dinheiro E Aquilo O Que Sempre Quer Ver Quem Eu Eh A Verdade 2X
	Tenho Medo De Tomar Um Ar Que Eu Tambem Sei O Cara E O Ken E Assim
	Noite Na Estica Cabelo Escovinha Nao Dava Na Rua E O Capao E Feliz E Ele Ta
	A Droga E O Funk Jamais Vao Morrer Na Festa No Que Deus E De Responsa Ta

Fonte: o autor

5.3 Conclusão

Os resultados obtidos com os dois protótipos mostraram que é possível gerar letras músicas com as redes LSTM. Este resultado pode obter sentenças musicais consideradas bastante satisfatórias, levando em consideração critérios como: sentido lógico; criatividade; e estrutura.

Tabela 12 – Comparativo do Loos dos protótipos

	Loss	
corpus	**Protótipo 1**	**Protótipo 2**
Raça Negra	0.83	1.23
Henrique e Juliano	0.78	0.94
Racionais MC's	1.45	1.47

Um comparativo do Loos apresentado na Tabela 12 mostra que no primeiro protótipo foram obtidos valores bem menores do que no segundo. É necessário levar em consideração que foram utilizados diferentes hiper-parâmetros e diferentes formas de se trabalhar com os dados dos corpuss.

Uma característica importante a ser analisada é a questão de no primeiro protótipo algumas sentenças rimarem. Um exemplo disso foram nas sentenças apresentadas na Tabela 6, "em qual mentira vou acreditar" e "o que vou encontrar", nelas é possível encontrar uma rima com a palavra "acreditar". Esse resultado podef ser observado apenas no primeiro protótipo.

6 DISCUSSÃO

Nesta seção são apresentados alguns pontos a serem discutidos sobre os experimentos. Os pontos a serem discutidos são: tamanho do corpus, gênero musical, métodos de avavaliação da rede e das letras.

A questão levantada com relação ao tamanho do corpus, é se o tamaho é realmente suficiente para o treinamento da rede neural. Visto que, em outros trabalhos (LUO; HUANG, 2017; Yizhe Zhang, Zhe Gan, 2016) foram utilizados corpus bem maiores. Visto isso, talvez com corpus bem maiores podemos obter resultamos melhores com a rede e com isso trazer letras com uma qualidade mais aceitável ao ser humano, pois o ser humano é capaz de detectar certos nuances que a maquina ainda não compreende.

Com relação ao gênero musical, pode-se perceber uma diferença no número de épocas e na qualidade das sentenças geradas. As letras de RAP contém mais inconsistência em sua estrutura, por conter gírias, e menos ligações que façam sentido, do que nas letras de sertanejo e samba, que possuem um sentido na ligação das sentenças e não possuem gírias. Visto isso, é preciso saber se há uma relação entre o gênero musical e o tamanho do corpus, e se isso pode afetar no treinamento e na qualidade das sentenças que são geradas pela rede neural.

Um outro ponto a ser discutido, está relacionado a forma de avaliação da rede e das letras. Com relação a rede neural, talvez o loss não seja um bom indicador se a rede está boa ou ruim, pois visto os resultados dos experimentos foi possível observar que mesmo com loss alto, obteve-se resultados satisfatórios. Com relação as letras, como validar se elas tem uma boa qualidade ou não, talvez seja preciso um teste de Turing para se verificar se as letras tem ou não boa qualidade, visto que para a máquina isso não é uma tarefa fácil.

7 CONCLUSÕES E TRABALHOS FUTUROS

Neste capítulo são descritas as considerações finais deste trabalho, que teve como objetivo desenvolver um modelo que fosse capaz de gerar letras musicais de diferentes estilos, utilizando redes LSTM.

Com a execução deste trabalho foram possíveis construir 3 corpus de musicas de diferentes estilos musicais, um crawler que foi utilizado para criar a corpora, e 2 protótipos que foram utilizados para treinar as redes neurais e gerar as sentenças musicais. Obteve-se também, resultados bastante satisfatórios com as sentenças geradas nos dois protótipos.

As dificuldades enfrentadas neste trabalho estão relacionadas a criação da corpora e testes com as redes neurais. A corpora apesar de ter sido capturada por um crawler, precisou ser analisada linha por linha, pois devido a formatação da Linguagem de Marcação de Hipertexto (HTML) das páginas, algumas das letras viam de forma bem desorganizada, com palavras unidas e outras separadas no meio. Além disso, em algumas capturas foram encontrados códigos em HTML, por isso foi necessária uma limpeza de forma manual. Um outro fator que dificultou bastante, foi a questão do tempo de treinamento, pois se levava horas dependendo da configuração para se treinar a rede. Vale ressaltar que foram testadas varias combinações até que se chegasse a um resultado mais satisfatório.

Como trabalhos futuros, é possível que as redes sejam treinadas com um corpus bem maior, pegando vários artistas de um mesmo gênero musical. Uma sugestão é tentar ajustar a arquitetura e os parâmetros da rede de acordo com o crescimento do corpus.

REFERÊNCIAS

Bengio, Y.; Simard, P.; Frasconi, P. Learning long-term dependencies with gradient descent is difficult. **IEEE Transactions on Neural Networks**, v. 5, n. 2, p. 157–166, March 1994.

BODEN, M. A. Creativity and Artificial Intelligence. In: **The Philosophy of Creativity**. [S.l.]: Oxford University Press, 2014. p. 224–244.

BROWNLEE, J. Jason Brownlee book. 2017.

CHOI, K.; FAZEKAS, G.; SANDLER, M. Text-based LSTM networks for Automatic Music Composition. p. 1–8, 2016. Disponível em: <http://arxiv.org/abs/1604.05358>.

DOğAN, E.; KAYA, B.; MÜNGEN, A. Generation of Original Text with Text Mining and Deep Learning Methods for Turkish and Other Languages. **2018 International Conference on Artificial Intelligence and Data Processing, IDAP 2018**, 2019.

GURNEY, K. **An Introduction to Neural Networks**. 1st editio. ed. London: CRC Press, 1997. 234 p. ISBN 9781482286991. Disponível em: <https://www.taylorfrancis.com/books/9781482286991>.

HAJ-YAHIA, Z. **Sequence Modeling with Neural Networks – Part I**. 2018. Disponível em: <https://www.kdnuggets.com/2018/10/sequence-modeling-neural-networks-part-1.html>.

HAYKIN, S. **Redes Neurais - 2ed.** Bookman, 2001. ISBN 9788573077186. Disponível em: <https://books.google.com.br/books?id=lBp0X5qfyjUC>. Acesso em: 28 abr. 2019.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural computation**, v. 9, p. 1735–80, 12 1997.

LUO, Y.; HUANG, Y. Text Steganography with High Embedding Rate: Using Recurrent Neural Networks to Generate Chinese Classic Poetry. p. 99–104, 2017.

NGUYEN, M. **Illustrated Guide to LSTM's and GRU's: A step by step explanation**. 2018. Disponível em: <https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>. Acesso em: 28 abr. 2019.

OLAH, C. **Understanding LSTM Networks**. 2015. Disponível em: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>.

OLSTON, C.; NAJORK, M. Web crawling. **Found. Trends Inf. Retr.**, Now Publishers Inc., Hanover, MA, USA, v. 4, n. 3, p. 175–246, mar. 2010. ISSN 1554-0669. Disponível em: <http://dx.doi.org/10.1561/1500000017>.

SON, S.-h.; LEE, H.-y. Korean Song-lyrics Generation by Deep Learning. In: **4th International Conference on Intelligent Information Technology**. Da, Nang, Viet Nam: [s.n.], 2019. p. 96–100.

Yizhe Zhang, Zhe Gan, L. C. Generating Text via Adversarial Training. 2016.

APÊNDICE A – CRAWLER

```
1  var request = require( request );
2  var cheerio = require( cheerio );
3
4  request( https://www.letras.com.br/racionais-mcs , function (err, res
       , body) {
5      if (err) console.log( Erro:   + err);
6
7      var $ = cheerio.load(body);
8
9      $( .lista li  ).each(function () {
10         var link = $(this).find( .c-claro a ).attr( href );
11         console.log(link);
12     });
13 });
```

Código-fonte 1 – Capturar links

```
1
2  var Crawler = require("crawler");
3
4  var lineReader = require( line-reader );
5
6  var c = new Crawler({
7      maxConnections: 10,
8      callback: function (error, res, done) {
9          if (error) {
10             console.log(error);
11         } else {
12             var $ = res.$;
13             console.log($("#letra").text());
14
15         }
16         done();
17     }
```

```
18 });
19
20
21 lineReader.eachLine( HJ-114-links.txt , function (line, last) {
22     c.queue(line);
23 });
```

Código-fonte 2 – Capturar letras

APÊNDICE B – PROTÓTITPO 1

Protótipo 1 utilizado para o treinamento e gerageração das sentenças.

Protótipo 1

November 15, 2019

1 Importações

[0]:
```
from __future__ import print_function
from keras.callbacks import LambdaCallback, ModelCheckpoint
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
from keras.optimizers import RMSprop
from keras.utils.data_utils import get_file
from keras.models import load_model
from keras.utils import multi_gpu_model
import matplotlib.pyplot as plt
import numpy as np
import random
import sys
import io
```

[0]:
```
with io.open('C:/Users/jefer/Desktop/TCC/Codigos/Dataset/raca-negra-240.txt')␣
 ↪as f:
    text = f.read().lower()
print('corpus length:', len(text))
```

2 Limpeza do Corpus

[0]:
```
text = text.replace('\n', ' ')
text = text.replace('.', ' ')
text = text.replace(',', ' ')
text = text.replace('?', ' ')
text = text.replace('!', ' ')
text = text.replace(':', ' ')
text = text.replace('  ', ' ')
text = text.replace('   ', ' ')
```

[0]:
```
text
```

[0]:
```
chars = sorted(list(set(text)))
print('total chars:', len(chars))
char_indices = dict((c, i) for i, c in enumerate(chars))
indices_char = dict((i, c) for i, c in enumerate(chars))
```

3 Criação dos Tokens

[0]:
```
maxlen = 40
step = 3
sentences = []
next_chars = []
for i in range(0, len(text) - maxlen, step):
    sentences.append(text[i: i + maxlen])
    next_chars.append(text[i + maxlen])
print('nb sequences:', len(sentences))
```

[0]:
```
print('Vectorization...')
x = np.zeros((len(sentences), maxlen, len(chars)), dtype=np.bool)
y = np.zeros((len(sentences), len(chars)), dtype=np.bool)
for i, sentence in enumerate(sentences):
    for t, char in enumerate(sentence):
        x[i, t, char_indices[char]] = 1
    y[i, char_indices[next_chars[i]]] = 1
```

4 Criação do modelo

[0]:
```
model = Sequential()
model.add(LSTM(128, input_shape=(maxlen, len(chars))))

model.add(Dense(len(chars), activation='softmax'))
optimizer = RMSprop(lr=0.01)
model.compile(loss='categorical_crossentropy', optimizer=optimizer)
model.summary()
```

[0]:
```
filepath = "weights-rn-.hdf5"
checkpoint = ModelCheckpoint(filepath,
                             monitor='loss',
                             verbose=1,
                             save_best_only=True,
                             mode='min')
```

[0]:
```
history = model.fit(x, y,
          batch_size=60,
          epochs=3,
          callbacks=[checkpoint])
```

[0]:
```
N=3
plt.plot(np.arange(0,N),history.history['loss'],label='Training_loss')
plt.title('Loss do treinamento - Raça Negra')
plt.xlabel('Épocas')
```

[0]:
```
def sample(preds, temperature=1.0):
    # helper function to sample an index from a probability array
    preds = np.asarray(preds).astype('float64')
    preds = np.log(preds) / temperature
    exp_preds = np.exp(preds)
    preds = exp_preds / np.sum(exp_preds)
    probas = np.random.multinomial(1, preds, 1)
    return np.argmax(probas)
```

[0]:
```
def generate_text(length, diversity):
    # Get random starting text
    start_index = random.randint(0, len(text) - maxlen - 1)
    generated = ''
    sentence = text[start_index: start_index + maxlen]
    generated += sentence
    for i in range(length):
            x_pred = np.zeros((1, maxlen, len(chars)))
            for t, char in enumerate(sentence):
                x_pred[0, t, char_indices[char]] = 1.

            preds = model.predict(x_pred, verbose=0)[0]
            next_index = sample(preds, diversity)
            next_char = indices_char[next_index]

            generated += next_char
            sentence = sentence[1:] + next_char
    return generated
```

[0]:
```
for i in range(30):
    print(generate_text(60, 0.3))
    print("----------")
```

[0]:
```
```

APÊNDICE C – PROTÓTITPO 2

Protótipo 2 utilizado para o treinamento e gerageração das sentenças.

Protótipo 2

November 15, 2019

1 Importações

[0]:
```
from keras.preprocessing.sequence import pad_sequences
from keras.layers import Embedding, LSTM, Dense, Dropout
from keras.preprocessing.text import Tokenizer
from keras.callbacks import EarlyStopping, ModelCheckpoint
from keras.models import Sequential
import keras.utils as ku
from tensorflow import set_random_seed
from numpy.random import seed
set_random_seed(2)
seed(1)
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import string, os
import sys
import io
import warnings
warnings.filterwarnings("ignore")
warnings.simplefilter(action='ignore', category=FutureWarning)
```

[0]:
```
texto  = []
with open("C:/Users/jefer/Desktop/TCC/Codigos/Dataset/HJ-206.txt") as file:
    for line in file:
        texto.append(line)
```

[0]:
```
len(texto)
```

2 Limpeza do Corpus

[0]:
```
def clearTxt(txt):
    txt = "".join(v for v in txt if v not in string.punctuation).lower()
    txt = txt.encode("utf8").decode("ascii",'ignore')
    txt = txt.strip('\n')
```

```
    return txt

corpus = [clearTxt(x) for x in texto]

while("" in corpus) :
    corpus.remove("")

corpus[:10]
```

3 Criação dos Tokens

[0]:
```
tokenizer = Tokenizer()

def creatTokens(corpus):

    tokenizer.fit_on_texts(corpus)
    total_words = len(tokenizer.word_index) + 1

    input_sequences = []
    for line in corpus:
        token_list = tokenizer.texts_to_sequences([line])[0]
        for i in range(1, len(token_list)):
            n_gram_sequence = token_list[:i+1]
            input_sequences.append(n_gram_sequence)
    return input_sequences, total_words

inp_sequences, total_words = creatTokens(corpus)
inp_sequences[:10]
```

[0]:
```
def creatPaddedSequences(input_sequences):
    max_sequence_len = max([len(x) for x in input_sequences])
    input_sequences = np.array(pad_sequences(input_sequences,␣
 ↪maxlen=max_sequence_len, padding='pre'))

    predictors, label = input_sequences[:,:-1],input_sequences[:,-1]
    label = ku.to_categorical(label, num_classes=total_words)
    return predictors, label, max_sequence_len

predictors, label, max_sequence_len = creatPaddedSequences(inp_sequences)
```

4 Criação do modelo

[0]:
```
def createModel(max_sequence_len, total_words):
    input_len = max_sequence_len - 1
    model = Sequential()
    # Camada de entrada
    model.add(Embedding(total_words, 10, input_length=input_len))

    # Camada oculta LSTM
    model.add(LSTM(256))
    model.add(Dropout(0.1))

    # Camada de saída
    model.add(Dense(total_words, activation='softmax'))

    model.compile(loss='categorical_crossentropy', optimizer='adam')

    return model

model = createModel(max_sequence_len, total_words)
model.summary()
```

5 Treinamento

[0]:
```
history = model.fit(predictors, label, batch_size=60, epochs=3, verbose=5)
```

[0]:
```
N=60
plt.plot(np.arange(0,N),history.history['loss'],label='Training_loss')
plt.title('Loss do treinamento - Henrique e Juliano')
plt.xlabel('Épocas')
```

[0]:
```
print(history.history['loss'])
```

6 Geração das Letras

[0]:
```
def generateText(seedText, nextWords, model, max_sequence_len):
    for _ in range(nextWords):
        token_list = tokenizer.texts_to_sequences([seedText])[0]
        token_list = pad_sequences([token_list], maxlen=max_sequence_len-1,␣
 ↪padding='pre')
        predicted = model.predict_classes(token_list, verbose=0)

        output_word = ""
```

```
        for word,index in tokenizer.word_index.items():
            if index == predicted:
                output_word = word
                break
        seedText += " "+output_word
    return seedText.title()
```

[0]:
```
str1 = generateText("vamos viver", 5, model, max_sequence_len)
str2 = generateText(str1, 5, model, max_sequence_len)
str3 = generateText(str2, 5, model, max_sequence_len)
print(str3)
```

[0]:
```

```