

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus Bauru

Gustavo Henrique de Rosa

Análise de Aprendizado Adversarial baseado em Similaridade na Geração de Texto

Bauru
2022

Gustavo Henrique de Rosa

Análise de Aprendizado Adversarial baseado em Similaridade na Geração de Texto

Tese apresentada como parte dos requisitos para obtenção do título de Doutor em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho".

Financiadora: FAPESP - Proc. 2019/02205-5

Orientador: Prof. Dr. João Paulo Papa

Bauru
2022

de Rosa, Gustavo Henrique.
Análise de Aprendizado Adversarial baseado em Similaridade na Geração de Texto / Gustavo Henrique de Rosa. – Bauru, 2022
140 f. : il., tabs.

Orientador: Prof. Dr. João Paulo Papa
Tese (doutorado) - Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências

1. Aprendizado de Máquina. 2. Aprendizado Adversarial. 3. Processamento de Linguagem Natural. 4. Geração de Texto. 5. Redes Siamesas. 6. Otimização Meta-Heurística.
I. Papa, João Paulo. II. Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências. III. Título.

CDU – 518.72:76

Ficha catalográfica elaborada pela biblioteca da FC
UNESP - Câmpus de Bauru

ATA DA DEFESA PÚBLICA DA TESE DE DOUTORADO DE GUSTAVO HENRIQUE DE ROSA, DISCENTE DO PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO, DA FACULDADE DE CIÊNCIAS - CÂMPUS DE BAURU.

Aos 22 dias do mês de novembro do ano de 2022, às 00:00 horas, por meio de Videoconferência, realizou-se a defesa de TESE DE DOUTORADO de GUSTAVO HENRIQUE DE ROSA, intitulada **Geração de Linguagem Natural Utilizando Aprendizado Adversarial Baseado em Similaridade**. A Comissão Examinadora foi constituida pelos seguintes membros: Prof. Dr. JOAO PAULO PAPA (Orientador(a) - Participação Virtual) do(a) FC / UNESPBauru SP, Profa. Dra. HELENA DE MEDEIROS CASELI (Participação Virtual) do(a) Departamento de Computação / Universidade Federal de São Carlos, Prof. Dr. DIEGO FURTADO SILVA (Participação Virtual) do(a) Departamento de Computação / Universidade Federal de São Carlos, Prof. Dr. APARECIDO NILCEU MARANA (Participação Virtual) do(a) Departamento de Computação / UNESP/Câmpus de Bauru , Prof. Dr. KELTON AUGUSTO PONTARA DA COSTA (Participação Virtual) do(a) FC / UNESP/Bauru (SP). Após a exposição pelo doutorando e arguição pelos membros da Comissão Examinadora que participaram do ato, de forma presencial e/ou virtual, o discente recebeu o conceito final:_ _ _ _ _ _ _ Aprovado . Nada mais havendo, foi lavrada a presente ata, que após lida e aprovada, foi assinada pelo(a) Presidente(a) da Comissão Examinadora.

João Paulo Papa

Prof. Dr. JOAO PAULO PAPA

Gustavo Henrique de Rosa

Análise de Aprendizado Adversarial baseado em Similaridade na Geração de Texto

Tese apresentada como parte dos requisitos para obtenção do título de Doutor em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho".

Financiadora: FAPESP - Proc. 2019/02205-5

Comissão Examinadora

Prof. Dr. João Paulo Papa
Orientador
Universidade Estadual Paulista "Júlio de Mesquita Filho"

Profa. Dra. Helena de Medeiros Caseli
Universidade Federal de São Carlos

Prof. Dr. Diego Furtado Silva
Universidade de São Paulo

Prof. Dr. Aparecido Nilceu Marana
Universidade Estadual Paulista "Júlio de Mesquita Filho"

Prof. Dr. Kelton Augusto Pontara da Costa
Universidade Estadual Paulista "Júlio de Mesquita Filho"

Bauru
5 de dezembro de 2022

Em tempos de trevas, que o conhecimento seja fonte de luz.

Agradecimentos

Em primeiro lugar, gostaria de agradecer à fé, que independente da religião e doutrina, deve sempre existir e prevalecer.

Agradeço aos meus pais, Ana Valéria e Fernando, por sempre estarem presentes em minha vida, me auxiliando e fortalecendo.

Agradeço à minha parceira, Bruna, a qual eu sempre amarei mais que tudo. Sem dúvidas todo o tempo que passamos juntos e que iremos passar me faz ser uma pessoa melhor e me motivar mais para, juntos, melhorarmos o mundo e crescermos.

Agradeço à minha prima Bárbara, aos meus primos Bento e Mateo, aos meus tios Fabrício e Helena, Luciene e Marcos, Helen, Daiane e João, aos meus sogros Simone e Ricardo, ao meu canino Prince, por todas as nossas conversas e ensinamentos, contribuindo à minha formação. Agradeço em especial aos meus avós, José e Thereza, Lurdinha e principalmente ao meu falecido avô Mauro.

Agradeço ao meu orientador, João Paulo Papa, o qual sempre foi uma fonte de inspiração para mim e com certeza um dos meus maiores orgulhos.

Agradeço à Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) pelo concebimento de uma bolsa de Doutorado (Processo #2019/02205-5), a qual possibilitou o desenvolvimento desta tese.

Agradeço à todos os meus amigos de faculdade e de Brasília. Sempre que nos reencontramos acaba sendo o mesmo amor e amizade.

Agradeço à UNESP Bauru, a qual é o meu maior orgulho de poder fazer parte e ser considerado um Unespiano.

"Oh, it's a disgrace
To see the human race
In a rat race, yeah"
(Rat Race, Bob Marley)

Resumo

Algoritmos de Aprendizado de Máquina têm sido amplamente fomentados nos últimos anos, principalmente devido às suas capacidades discriminativas em problemas de Visão Computacional e Processamento de Linguagem Natural. Ademais, suas capacidades generativas permitiram aplicações em tarefas de natureza discreta (sequências de caracteres e palavras), isto é, geração de texto. Uma arquitetura em específico, denotada por Redes Adversariais Generativas, utiliza uma estrutura composta por discriminador e gerador, os quais procuram obter um equilíbrio entre gerar dados artificiais e classificá-los como dados verdadeiros. Recentemente, vários trabalhos propuseram soluções baseadas em Redes Adversariais Generativas para a geração de texto, porém apenas alguns deles conseguiram gerar textos sem palavras repetidas e com algum significado semântico. Um problema decorrente das Redes Adversariais Generativas consiste na dificuldade em estabelecer um equilíbrio no treinamento e, consequentemente, gerar textos artificiais que assemelham-se aos textos reais. Desta forma, a presente tese aprimora o desenvolvimento de modelos adversariais textuais através de funções de similaridade aprendidas por Redes Siamesas, as quais fornecem recompensas capazes de melhor distinguir entre textos artificiais e reais. Adicionalmente, tais modelos são aperfeiçoados com o uso da otimização meta-heurística, a qual fornece conjuntos de hiperparâmetros específicos para as tarefas em questão. Os resultados experimentais obtidos demonstram a capacidade da arquitetura proposta, denotada por Rede Adversarial Generativa por Similaridade Textual, do inglês *Text-Similarity Generative Adversarial Network* (TS-GAN), em quatro base de dados da literatura. As TS-GANs atingiram resultados superiores às arquiteturas adversariais estado-da-arte e, em suas versões pós-otimização, foram capazes de aprimorar as métricas de suas versões iniciais (sem otimização) em duas de quatro bases de dados.

Palavras-chave: Aprendizado de Máquina. Aprendizado Adversarial. Processamento de Linguagem Natural. Geração de Texto. Redes Siamesas. Otimização Meta-Heurística.

Abstract

Machine Learning algorithms have been paramount in the latest years, mainly due to their discriminative capacity in Computer Vision and Natural Language Processing tasks. Furthermore, their generative potentials allowed usage in discrete-based (sequences of characters and words) tasks, such as text generation. A specific architecture denoted as Generative Adversarial Networks uses an structure composed of discriminators and generators to establish an equilibrium between artificial data generation and their classification as real data. Several works proposed adversarial-based models to generate text; however, only a few could generate non-repeated text with little semantic significance. Furthermore, a recurring issue regarding Generative Adversarial Networks consists of the difficulty of establishing a training equilibrium and, consequently, generating artificial text that resembles the original ones. Therefore, this thesis enhances the development of text-based adversarial models through similarity functions learned from Siamese Networks, which provides rewards capable of better distinguishing between artificial and original texts. Such models are improved through meta-heuristic optimization, which furnishes specific hyperparameters to the accounted tasks. The experimental results indicate the capacity of the proposed architecture, denoted by Text-Similarity Generative Adversarial Network (TS-GAN), amongst four literature datasets. The TS-GANs obtained state-of-the-art results and, in their post-optimization versions, were able to improve their standard versions (without optimization) in two out of four datasets.

Keywords: Machine Learning. Adversarial Learning. Natural Language Processing. Text Generation. Siamese Networks. Meta-Heuristic Optimization.

Lista de ilustrações

Figura 1 – Processo de comunicação simplificado entre dois seres humanos. 25
Figura 2 – Processo de geração de texto a partir de um modelo linguístico. 28
Figura 3 – Arquitetura básica de uma Rede Neural Recorrente. 38
Figura 4 – Arquitetura de uma célula de uma Rede de Memória de Longo Prazo. . . . 40
Figura 5 – Arquitetura de uma Unidade Recorrente de Porta. 42
Figura 6 – Arquitetura de uma célula de uma Rede de Memória Relacional. 45
Figura 7 – Mecanismo de auto-atenção. 46
Figura 8 – Arquitetura padrão de uma Rede Adversarial Generativa. 48
Figura 9 – Estabilização do treinamento de uma Rede Adversarial Generativa: (a) distribuição de dados gerados é diferente da distribuição de dados reais e o discriminador ainda não foi treinado, (b) discriminador está sendo treinado para estabilizar o gerador, (c) gradiente do discriminador guiou o gerador à uma região mais propensa a mapear os dados reais e (d) ao fim do treinamento, se o gerador e o discriminador estabilizarem, chegarão à um ponto onde o discriminador não conseguirá diferenciar as distribuições. . . 49
Figura 10 – Exemplo de arquitetura de uma Rede Siamesa. 51
Figura 11 – Exemplo da função $f(x, y) = \text{sen}(x) \cdot \text{sqrt}(x) \cdot \text{sen}(y) \cdot \text{sqrt}(y)$ com diversos pontos de mínimo e máximo locais. 56
Figura 12 – Exemplo de uma solução do GP representando a expressão $1,7\log(x) - 3,9\text{cos}(y)$. 64
Figura 13 – Fluxograma da evolução de uma população da Programação Genética. . . . 65
Figura 14 – Arquitetura de uma Rede Adversarial Generativa Sequencial. 67
Figura 15 – Arquitetura de uma Rede Adversarial Generativa *Gumbel-Softmax*. 70
Figura 16 – Arquitetura de uma Rede Adversarial Generativa Ampliada Por Máxima-Verosimilhança. 72
Figura 17 – Arquitetura de uma Rede Adversarial Generativa Relacional. 74
Figura 18 – Arquitetura de uma Rede Adversarial Generativa por Similaridade Textual. . 76
Figura 19 – Fluxograma da metodologia proposta. 80
Figura 20 – Fluxograma proposto para a Frente I. 84
Figura 21 – Fluxograma proposto para a Frente II. 90
Figura 22 – Convergência da função de perda de treinamento na base de dados Avaliação dos Consumidores da Amazon: (a) recorrente, (b) discriminador adversarial e (c) gerador adversarial. 94
Figura 23 – Convergência da função de perda de treinamento na base de dados Avaliações do Yelp: (a) recorrente, (b) discriminador adversarial e (c) gerador adversarial. 95

Figura 24 – Convergência da função de perda de treinamento na base de dados Legendas de Imagens COCO: (a) recorrente, (b) discriminador adversarial e (c) gerador adversarial. 96
Figura 25 – Convergência da função de perda de treinamento na base de dados Notícias do WMT EMNLP2017: (a) recorrente, (b) discriminador adversarial e (c) gerador adversarial. 97
Figura 26 – Análise comparativa da efetividade do processo de busca e modelagem da Rede Adversarial Generativa por Similaridade Textual na base de dados Avaliação dos Consumidores da Amazon. 110
Figura 27 – Análise comparativa da efetividade do processo de busca e modelagem da Rede Adversarial Generativa por Similaridade Textual na base de dados Avaliações do Yelp. 110
Figura 28 – Análise comparativa da efetividade do processo de busca e modelagem da Rede Adversarial Generativa por Similaridade Textual na base de dados Legendas de Imagens COCO. 111
Figura 29 – Análise comparativa da efetividade do processo de busca e modelagem da Rede Adversarial Generativa por Similaridade Textual na base de dados Notícias do WMT EMNLP2017. 111
Figura 30 – Convergência da função objetivo de otimização na base de dados Avaliações de Consumidores da Amazon. 112
Figura 31 – Convergência da função objetivo de otimização na base de dados Avaliações do Yelp. 113
Figura 32 – Convergência da função objetivo de otimização na base de dados Legendas de Imagens COCO. 114
Figura 33 – Convergência da função objetivo de otimização na base de dados Notícias do WMT EMNLP2017. 114

Lista de tabelas

Tabela 1 – Sigla e nomenclaturas das meta-heurísticas utilizadas. 57
Tabela 2 – Exemplos de avaliações da base de dados Avaliação de Consumidores da Amazon. 81
Tabela 3 – Exemplos de avaliações da base de dados Avaliações do Yelp. 82
Tabela 4 – Exemplos de amostras da base de dados Legendas de Imagens COCO. . . . 83
Tabela 5 – Exemplos de sentenças da base de dados Notícias do WMT EMNLP2017. . 83
Tabela 6 – Dicionário (nome e símbolo) dos hiperparâmetros utilizados na Frente I. . . 86
Tabela 7 – Configuração experimental das arquiteturas e de seus hiperparâmetros para cada base de dados. 89
Tabela 8 – Configuração experimental das meta-heurísticas. 91
Tabela 9 – Configuração experimental do intervalo de busca dos hiperparâmetros para cada base de dados. 92
Tabela 10 – Métricas dos modelos avaliados no conjunto de teste da base de dados Avaliação de Consumidores da Amazon. 98
Tabela 11 – Métricas dos modelos avaliados no conjunto de teste da base de dados Avaliações do Yelp. 99
Tabela 12 – Métricas dos modelos avaliados no conjunto de teste da base de dados Legendas de Imagens COCO. 100
Tabela 13 – Métricas dos modelos avaliados no conjunto de teste da base de dados Notícias do WMT EMNLP2017. 101
Tabela 14 – Métricas da TS-GAN-C e TS-GAN-T com diferentes geradores e discriminadores avaliadas no conjunto de teste da base de dados Avaliação de Consumidores da Amazon. 102
Tabela 15 – Métricas da TS-GAN-C e TS-GAN-T com diferentes geradores e discriminadores avaliadas no conjunto de teste da base de dados Avaliações do Yelp. 103
Tabela 16 – Métricas da TS-GAN-C e TS-GAN-T com diferentes geradores e discriminadores avaliadas no conjunto de teste da base de dados Legendas de Imagens COCO. 104
Tabela 17 – Métricas da TS-GAN-C e TS-GAN-T com diferentes geradores e discriminadores avaliadas no conjunto de teste da base de dados Notícias do WMT EMNLP2017. 105
Tabela 18 – Métricas da TS-GAN (gerador LSTM e discriminador RNN) com diferentes funções de perda e distâncias avaliadas no conjunto de teste da base de dados Avaliação de Consumidores da Amazon. 106

Tabela 19 – Métricas da TS-GAN (gerador LSTM e discriminador RNN) com diferentes funções de perda e distâncias avaliadas no conjunto de teste da base de dados Avaliações do Yelp. 107
Tabela 20 – Métricas da TS-GAN (gerador LSTM e discriminador GRU) com diferentes funções de perda e distâncias avaliadas no conjunto de teste da base de dados Legendas de Imagens COCO. 108
Tabela 21 – Métricas da TS-GAN (gerador LSTM e discriminador GRU) com diferentes funções de perda e distâncias avaliadas no conjunto de teste da base de dados Notícias do WMT EMNLP2017. 109
Tabela 22 – Hiperparâmetros encontrados durante a otimização da TS-GAN (gerador LSTM, discriminador RNN, função de perda Contrastiva e função de distância Euclidiana) na base de dados Avaliações de Consumidores da Amazon. 116
Tabela 23 – Métricas pós-otimização da TS-GAN (gerador LSTM, discriminador RNN, função de perda Contrastiva e função de distância Euclidiana) avaliada no conjunto de teste da base de dados Avaliação de Consumidores da Amazon. 116
Tabela 24 – Hiperparâmetros encontrados durante a otimização da TS-GAN (gerador LSTM, discriminador RNN, função de perda Contrastiva e função de distância Euclidiana) na base de dados Avaliações do Yelp. 117
Tabela 25 – Métricas pós-otimização da TS-GAN (gerador LSTM, discriminador RNN, função de perda Contrastiva e função de distância Euclidiana) avaliada no conjunto de teste da base de dados Avaliações do Yelp. 117
Tabela 26 – Hiperparâmetros encontrados durante a otimização da TS-GAN (gerador LSTM, discriminador GRU, função de perda Contrastiva e função de distância Manhattan) na base de dados Legendas de Imagens COCO. 118
Tabela 27 – Métricas pós-otimização da TS-GAN (gerador LSTM, discriminador GRU, função de perda Contrastiva e função de distância Manhattan) avaliada no conjunto de teste da base de dados Legendas de Imagens COCO. 118
Tabela 28 – Hiperparâmetros encontrados durante a otimização da TS-GAN (gerador LSTM, discriminador GRU, função de perda Contrastiva e função de distância Euclidiana) na base de dados Notícias do WMT EMNLP2017. . . . 119
Tabela 29 – Métricas pós-otimização da TS-GAN (gerador LSTM, discriminador GRU, função de perda Contrastiva e função de distância Euclidiana) avaliada no conjunto de teste da base de dados Notícias do WMT EMNLP2017. . . . 119
Tabela 30 – Tempo médio de treinamento e otimização, em horas, dos modelos recorrentes e adversariais na base de dados Avaliação do Consumidores da Amazon. 120
Tabela 31 – Tempo médio de treinamento e otimização, em horas, dos modelos recorrentes e adversariais na base de dados Avaliações do Yelp. 120

Tabela 32 – Tempo médio de treinamento e otimização, em horas, dos modelos recorrentes e adversariais na base de dados Legendas de Imagens COCO. 121
Tabela 33 – Tempo médio de treinamento e otimização, em horas, dos modelos recorrentes e adversariais na base de dados Notícias do WMT EMNLP2017. . . 121

Lista de abreviaturas e siglas

ACM	*Association for Computing Machinery*
AI	*Artificial Intelligence*
AL	*Adversarial Learning*
BA	*Bat Algorithm*
BCE	*Binary Cross-Entropy*
BLEU	*Bilingual Evaluation Understudy*
CL	*Contrastive Loss*
CNN	*Convolutional Neural Network*
COCO	*Common Objects in Context*
CS	*Cuckoo Search*
CV	*Computer Vision*
DL	*Deep Learning*
FA	*Firefly Algorithm*
GA	*Genetic Algorithm*
GAN	*Generative Adversarial Network*
GP	*Genetic Programming*
GRU	*Gated Recurrent Unit*
GSGAN	*Gumbel-Softmax Generative Adversarial Network*
LeakGAN	*Leak Generative Adversarial Network*
LSTM	*Long Short-Term Memory*
MaliGAN	*Maximum-Likelihood Augmented Generative Adversarial Network*
MC	*Monte Carlo*
METEOR	*Metric for Evaluation of Translation with Explicit Ordering*

ML	*Machine Learning*
MLE	*Maximum Likelihood Estimation*
MSE	*Mean Squared Error*
NALP	*Natural Adversarial Language Processing*
NLL	*Negative Log-Likelihood*
NLP	*Natural Language Processing*
PSO	*Particle Swarm Optimization*
RelGAN	*Relational Generative Adversarial Network*
ReLU	*Rectified Linear Unit*
RL	*Reinforcement Learning*
RMC	*Relational Memory Core*
RNN	*Recurrent Neural Network*
ROUGE	*Recall-Oriented Understudy for Gisting Evaluation*
SeqGAN	*Sequence Generative Adversarial Network*
SNN	*Siamese Neural Network*
TL	*Triplet Loss*
TS-GAN	*Text-Similarity Generative Adversarial Network*
WMT	*Workshop in Statistical Machine Translation*

Sumário

1 INTRODUÇÃO 19
1.1 Problema 20
1.2 Proposta 22
1.3 Objetivos 22
1.3.1 Objetivo Geral 22
1.3.2 Objetivos Específicos 23
1.4 Contribuições 23
1.5 Estrutura da Tese 24

2 GERAÇÃO DE TEXTO 25
2.1 Estágios da Geração 26
2.2 Modelos Linguísticos 27
2.3 Amostragem de Distribuições Probabilísticas 30
2.4 Métricas de Avaliação 32

3 REDES NEURAIS 36
3.1 Redes Neurais Recorrentes 37
3.1.1 Rede Neural Recorrente de Elman 38
3.1.2 Rede de Memória de Longo Prazo 39
3.1.3 Unidade Recorrente de Porta 41
3.1.4 Rede de Memória Relacional 44
3.2 Redes Adversariais Generativas 47
3.2.1 Aprendizado por Reforço 50
3.2.2 Diferenciação por *Gumbel-Softmax* 50
3.3 Redes Siamesas 51
3.3.1 Funções de Perda por Similaridade 52

4 OTIMIZAÇÃO META-HEURÍSTICA 55
4.1 Algoritmo do Morcego 57
4.2 Algoritmo do Vagalume 59
4.3 Busca do Cuckoo 60
4.4 Otimização por Enxame de Partículas 62
4.5 Programação Genética 64

5 TRABALHOS RELACIONADOS 66
5.1 Rede Adversarial Generativa Sequencial 67

5.2 Rede Adversarial Generativa *Gumbel-Softmax* . . . 69
5.3 Rede Adversarial Generativa Ampliada Por Máxima-Verosimilhança 71
5.4 Rede Adversarial Generativa Relacional . . . 73

6 REDE ADVERSARIAL GENERATIVA POR SIMILARIDADE TEXTUAL . . . 76
6.1 Discriminador por Similaridade . . . 77
6.2 Estratégia de Geração . . . 77
6.3 Algoritmo de Treinamento . . . 78

7 METODOLOGIA . . . 80
7.1 Bases de Dados . . . 81
7.2 Frente I . . . 84
7.2.1 Experimentos Propostos . . . 85
7.2.2 Configuração Experimental . . . 85
7.3 Frente II . . . 90
7.3.1 Experimento Proposto . . . 90
7.3.2 Configuração Experimental . . . 91

8 EXPERIMENTOS E RESULTADOS . . . 93
8.1 Frente I . . . 93
8.1.1 Convergência do Treinamento . . . 93
8.1.2 Qualidade da Geração de Texto . . . 97
8.1.3 Influência da Arquitetura dos Geradores e Discriminadores . . . 101
8.1.4 Comparação entre Funções de Perda . . . 106
8.2 Frente II . . . 112
8.2.1 Convergência da Otimização . . . 112
8.2.2 Influência dos Hiperparâmetros Otimizados . . . 115
8.2.3 Análise do Custo Computacional . . . 119

9 CONCLUSÕES E TRABALHOS FUTUROS . . . 123

REFERÊNCIAS . . . 128

APÊNDICES 134

APÊNDICE A – EXEMPLOS DE TEXTOS GERADOS . . . 135
A.1 Avaliação de Consumidores da Amazon . . . 135
A.2 Avaliações do Yelp . . . 137
A.3 Legendas de Imagens COCO . . . 139

A.4 Notícias do WMT EMNLP2017 140

1 Introdução

Recentes avanços em técnicas inteligentes possibilitaram a resolução de problemas com um mínimo de esforço humano. A capacidade computacional, responsável por realizar bilhões de cálculos por segundo, facilitou a realização de tarefas antes executadas apenas por seres humanos. A área da Inteligência Artificial (IA) objetiva estudar tais operações dentro do contexto biológico, tentando simular o funcionamento do cérebro humano. Em outras palavras, a pesquisa nesta área é responsável por modelar intrinsecamente o cérebro humano, dado que o poder de compreensão do nosso cérebro auxilia e orienta a criação de algoritmos capazes de assimilar informações ao seu redor e aprender com suas próprias ações (ULLMAN, 2019).

Entretanto, o conceito de IA é muito abrangente devido às inúmeras possibilidades de tarefas a serem desempenhadas. À medida que o estudo nesta área evoluiu, técnicas de Reconhecimento de Padrões e da Teoria do Aprendizado Computacional combinaram-se em uma nova subárea responsável por criar algoritmos capazes de aprender com seus erros e realizar predições, intitulada Aprendizado de Máquina, do inglês *Machine Learning* (ML) (SARKER, 2021). Esta área fornece importantes ferramentas para a resolução de problemas de outras subáreas da Inteligência Artificial, tais como a Visão Computacional e o Processamento de Linguagem Natural.

A Visão Computacional, do inglês *Computer Vision* (CV), procura compreender como os seres humanos interpretam informações através de sua visão (LECUN et al., 2010). É de conhecimento biológico que o sistema visual humano enxerga através de feixes de luz captados pela retina, a qual é responsável por decodificar estas informações, transformá-las em impulsos nervosos, carregá-los pelos neurônios e interpretá-los no cérebro. Uma das características mais interessantes de nossa visão é o aprendizado hierárquico de informações, onde à medida que enxergamos determinado objeto, procuramos decompô-lo em pequenos blocos de características para facilitar o entendimento do mesmo. Tal analogia é empregada em um algoritmo de CV conhecido por Redes Neurais Convolucionais, do inglês *Convolutional Neural Networks* (CNN) (LECUN et al., 1998).

Por outro lado, a finalidade do Processamento de Linguagem Natural, do inglês *Natural Language Processing* (NLP), é entender e aprender como seres humanos comunicam-se entre si, sendo boa parte desta comunicação realizada através de texto ou voz (EISENSTEIN, 2019). A criação de algoritmos capazes de interpretar a linguagem humana possibilita uma melhor interação entre máquinas e seres humanos, tornando-as mais aptas a nos entender e realizar tarefas de acordo com a nossa necessidade.

Inicialmente, as Redes Neurais Recorrentes, do inglês *Recurrent Neural Networks* (RNN) (ELMAN, 1990), tornaram-se as primeiras redes neurais a serem empregadas em tarefas de NLP. Entretanto, foram rapidamente substituídas pelas Redes de Memória de Longo Prazo, do inglês *Long Short-Term Memory* (LSTM) (HOCHREITER; SCHMIDHUBER, 1997), devido à sua maior resiliência ao problema da dissipação e estouro do gradiente, do inglês *vanishing and exploding gradient problem*. Outra técnica que também demonstrou-se resiliente à este problema foi a Unidade Recorrente de Porta, do inglês *Gated Recurrent Unit* (GRU) (CHO et al., 2014), a qual utiliza um mecanismo de portões similar às LSTMs para decidir quais informações devem ser consideradas. Uma evolução da LSTM, chamada de LSTM Bi-direcional (GRAVES et al., 2013), foi proposta a fim de abordar contexto em ambas direções (esquerda e direita). Adicionalmente, as CNNs também começaram a serem utilizadas no contexto de NLP (SONG et al., 2019; GIMÉNEZ et al., 2020), dado que são facilmente paralelizáveis e capazes de extrair relações semânticas em sequências. Recentemente, é possível encontrar abordagens híbridas (ZHENG; ZHENG, 2019; JAIN et al., 2021), em que a combinação de CNNs e LSTMs incrementam a robustez do modelo.

Embora a mídia e a comunidade salientem que muitos problemas da AI estejam resolvidos, os modelos de ML, sejam eles utilizados em CV ou NLP, ainda são bastante vulneráveis à manipulação adversarial (MORRIS et al., 2020), ou seja, são vulneráveis às pequenas modificações em dados não utilizados no aprendizado, resultando em classificações e previsões incorretas. Esta vulnerabilidade representa potenciais ameaças de segurança e é capaz de comprometer aplicações reais de ML, sendo assim, um interessante campo de pesquisa a ser fomentado.

O propósito do Aprendizado Adversarial, do inglês *Adversarial Learning* (AL), é introduzir exemplos artificialmente criados, distintos dos dados originais e capazes de dificultar o processo de aprendizado da rede. À cada iteração do treinamento, novos exemplos adversariais são gerados e introduzidos na rede, alavancando a performance e a robustez dos modelos de ML, tornando-os mais eficazes quando utilizados em aplicações do mundo real. Adicionalmente, tal metodologia é uma opção viável para a geração de dados artificiais que assemelham-se aos dados reais (geração de texto).

1.1 Problema

A tarefa de geração de texto objetiva entender a estrutura do texto e criar um modelo linguístico a partir do aprendizado de regras gramaticais, sintáticas e semânticas. De um ponto de vista computacional, esta tarefa almeja predizer um estado $t + 1$ dado t estados anteriores, onde cada estado pode ser composto por caracteres, palavras e até mesmo sentenças. Portanto, um modelo linguístico robusto é capaz de reproduzir a estrutura do texto treinado a partir da predição de novos dados baseados em informações passadas, ou seja, gerar texto artificialmente.

Por exemplo, imagine uma linguagem fictícia composta por três *tokens* (informações), a qual segue a determinada estrutura: <sujeito> <verbo> <substantivo>. O objetivo do modelo linguístico é aprender tal estrutura, sendo capaz de predizer um *token* de <verbo> baseado na entrada de um *token* de <sujeito>, bem como predizer um *token* de <substantivo> baseado em *tokens* de <sujeito> e <verbo>.

A geração de texto difere de tarefas tradicionais de Aprendizado de Máquina pois utiliza uma nova dependência sequencial. Em outras palavras, *tokens* (caracteres, palavras ou sentenças) são interpretados como informações contínuas, onde cada peça representa um estado na sequência. Adicionalmente, sequências prévias precisam ser compreendidas no futuro, dado que um modelo linguístico aprende novas informações baseadas em informações passadas (LI et al., 2018). Tradicionalmente, a tarefa de geração de texto é solucionada através de modelos recorrentes, tais como Redes Neurais Recorrentes, as quais são capazes de re-utilizar informações aprendidas para aprender novas informações. Não obstante, tais arquiteturas são suscetíveis à manipulação adversarial (BIGGIO; ROLI, 2018), isto é, tais modelos não atingem performances adequadas quando confrontados com dados distintos do processo de aprendizado, ou seja, dados levemente modificados.

Recentemente, as Redes Adversariais Generativas, do inglês *Generative Adversarial Networks* (GAN) (GOODFELLOW et al., 2014), tornaram-se uma importante base para a manipulação adversarial. A ideia é treinar simultaneamente duas redes distintas, uma com poder generativo, capaz de aprender a distribuição dos dados e gerar dados artificiais, e outra com poder discriminativo, capaz de estimar a probabilidade de uma amostra pertencer aos dados reais ao invés dos dados artificiais. Contudo, o treinamento simultâneo de duas arquiteturas é instável e ocasionalmente não atinge um equilíbrio de convergência. Ademais, tais redes precisam de um ajuste matemático quando utilizadas em tarefas de geração de texto, isto é, uma função de saída diferenciável[1] para que possam propagar seus gradientes e garantir a convergência do treinamento.

Uma estratégia para reduzir a instabilidade do treinamento é aprimorar o processo de identificação entre dados reais e falsos, por exemplo, a partir de uma função de similaridade. Neste contexto, as Redes Siamesas (ROY et al., 2019) são modeladas com o intuito de aprender a discernir se um par de dados é similar ou não. Elas também são indiferentes às arquiteturas, ou seja, quaisquer modelos podem ser utilizados como bases, tais como RNNs e CNNs, e consequentemente, utilizados como parte de uma arquitetura adversarial. Entretanto, tais arquiteturas requerem um alto número de hiperparâmetros de entrada, tais como, taxa de aprendizado, número de neurônios de vetorização, número de neurônios escondidos, temperatura, dentre outros. Estes hiperparâmetros são de extrema importância e precisam ser bem definidos, pois atuam como diretrizes e regularizadores do processo de treinamento, e auxiliam o processo de convergência e equilíbrio da rede neural. Recentemente, técnicas de otimização inspiradas

[1] Comumente, redes recorrentes utilizam uma função de saída não-diferenciável denotada por *Softmax*.

na natureza, denotadas por meta-heurísticas (YANG, 2011), vêm sendo utilizadas para a otimização de hiperparâmetros, dado que são capazes de realizar buscas extensas e intensas, encontrando soluções plausíveis sem recorrer aos algoritmos tradicionais de otimização, por exemplo, algoritmos baseados em derivadas e gradientes.

1.2 Proposta

Atualmente, a literatura apresenta diversos modelos adversariais textuais capazes de gerar dados sintéticos, contudo, apenas alguns deles conseguiram gerar textos sem palavras repetidas e com algum significado semântico. A dificuldade em estabelecer um equilíbrio de convergência, bem como a grande quantidade de hiperparâmetros são fatores que dificultam o aprendizado destes modelos e, consequentemente, a geração de textos lexicalmente corretos, sintaticamente adequados e semanticamente plausíveis.

Esta tese pretende aperfeiçoar a geração de texto de modelos adversariais textuais através do aprimoramento da estabilidade do seu processo de aprendizado, onde um conjunto adicional de funções oriundas das Redes Siamesas, denotado por funções de similaridade, serão empregados para melhor discernir entre dados reais e falsos e, consequentemente, fortalecer a geração de textos artificiais. Adicionalmente, pretende-se utilizar técnicas de otimização meta-heurística para selecionar conjuntos mais aptos de hiperparâmetros dos modelos supracitados. Portanto, a presente tese pode ser resumida em duas frentes de pesquisa:

a) Frente I: geração de texto utilizando modelos adversariais textuais;
b) Frente II: seleção de hiperparâmetros via otimização meta-heurística para modelos adversariais textuais.

Ademais, tais frentes de pesquisa serão validadas nas mesmas bases de dados utilizadas pelos modelos adversariais textuais estado da arte. Tal proposição tem o intuito de estabelecer uma análise comparativa e verificar tanto a efetividade e eficácia da metodologia proposta.

1.3 Objetivos

1.3.1 Objetivo Geral

Propor a utilização de funções de similaridade aprendidas por Redes Siameses para estabilizar o treinamento de modelos adversariais textuais. Adicionalmente, aliar o uso da otimização meta-heurística ao problema de seleção de hiperparâmetros de modelos adversariais textuais, visando aperfeiçoar a robustez e a geração de texto dos mesmos.

1.3.2 Objetivos Específicos

Esta tese objetiva validar o problema da geração de texto baseado em modelos adversariais textuais utilizando-se duas frentes: utilização de funções de similaridade aprendidas por Redes Siameses para estabilização do treinamento de modelos adversariais textuais, e otimização meta-heurística para seleção de hiperparâmetros de modelos adversariais textuais. Os passos necessários à sua obtenção estão descritos a seguir:

a) Apontar os principais problemas e limitações dos modelos adversariais estado da arte aplicados à geração de texto;

b) Modelar a abordagem de geração de texto por meio de um modelo adversarial textual baseado em funções de similaridade;

c) Apontar a aplicabilidade de técnicas de otimização meta-heurística para a otimização do modelo adversarial textual proposto;

d) Modelar a otimização meta-heurística de hiperparâmetros do modelo adversarial proposto aplicado à geração de texto;

e) Validar as metodologias propostas nas mesmas bases de dados utilizadas pelos modelos estado da arte, visando estabelecer bases comparativas;

1.4 Contribuições

Além de atingir os objetivos propostos, esta tese pretende contribuir os seguintes pontos à comunidade científica:

a) Criação de modelos adversariais textuais baseados em funções de similaridade;

b) Otimização de hiperparâmetros dos modelos adversariais textuais propostos;

c) Aprimoramento da tarefa de geração de texto através das estratégias supracitadas;

d) Implementação de uma biblioteca de Aprendizado Adversarial aplicado ao Processamento de Linguagem Natural, denominada *Natural Adversarial Language Processing* (NALP);

e) Implementação de uma biblioteca de Redes Siamesas, denominada *Dual-based Neural Learning* (Dualing);

f) Implementação de uma biblioteca de otimização meta-heurística, denominada *Opytimizer*;

g) Publicação de resultados e inovações em conferências e periódicos correlatos.

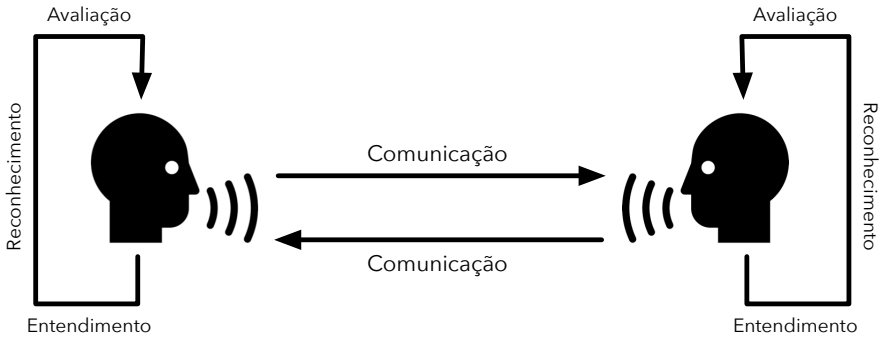
1.5 Estrutura da Tese

Esta tese de doutorado está organizada da seguinte forma: o Capítulo 2 apresenta conceitos e definições sobre a tarefa de geração de texto, enquanto o Capítulo 3 apresenta conceitos e definições sobre Redes Neurais Recorrentes, Redes Adversariais Generativas e Redes Siamesas. O Capítulo 4 introduz a fundamentação teórica referente à Otimização Meta-Heurística. O Capítulo 5 apresenta os trabalhos correlatos e que servem de base comparativa para a presente tese. O Capítulo 6 introduz as bases formais e definições teóricas sobre o modelo adversarial textual proposto, denotado por Rede Adversarial Generativa por Similaridade Textual. O Capítulo 7 apresenta a metodologia para a condução dos experimentos propostos, enquanto o Capítulo 8 apresenta os resultados experimentais, bem como uma análise crítica sobre os mesmos. O Capítulo 9 apresenta as conclusões e trabalhos futuros desta tese, bem como as publicações realizadas durante o período da mesma. Finalmente, o Apêndice A exemplifica textos gerados pelos modelos experimentados.

2 Geração de Texto

Quaisquer formas de comunicação, escritas ou verbais, carregam vastas quantias de informação, as quais podem ser utilizadas para prever o comportamento humano. Entretanto, tais informações não são triviais de serem analisadas, pois cada ser humano possui características inerentes à sua forma de comunicação, tais como, tom, seleção de palavras, vocabulário, sotaque, dentre outras. A Figura 1 ilustra um processo de comunicação simplificado entre dois seres humanos.

Figura 1 – Processo de comunicação simplificado entre dois seres humanos.



Fonte: Elaborado pelo autor.

A modelagem e compreensão da interação entre humanos e computadores é estruturada através de uma linguagem natural, e objetiva aprender estruturas inerentes à linguagem humana, tais como, morfologia, sintaxe e semântica. Desta forma, é possível definir a geração de texto como o processo generativo de construção de texto a partir de uma linguagem natural. Contudo, tal tarefa é complicada pois as linguagens humanas requerem um discernimento tanto de caracteres como de palavras e, ademais, como ambos estão conectados. Enquanto humanos são capazes de aprender e masterizar uma linguagem facilmente, as máquinas ainda esbarram em suas características ambíguas e imprecisas, tornando o processo extremamente moroso.

Inicialmente, os sistemas de geração de texto eram estruturados através de sistemas baseados em regras (MCDONALD, 2010), onde vastas quantidades de regras eram "traduzidas" em formatos pré-definidos de linguagem. Tal abordagem pode ser exemplificada pelos sistemas preenchedores de lacunas, os quais geralmente eram organizados em estruturas constantes e apenas produziam saídas definidas por planos de ações[2] (FRANK, 1980). Na década de 90, sistemas de geração de texto foram aplicados em produtos comerciais de caráter simples, principalmente para mapeamento de dados de entrada em textos pré-definidos. Por

[2] Os planos de ações eram gramáticas baseadas em teorias de linguagens formais e estruturadas através da linguagem de programação LISP.

exemplo, o primeiro sistema comercial focado em mapeamento de dados para texto foi o *Forecast Generator* (FoG) (Goldberg et al., 1994), o qual gerava textos de previsão do tempo em Francês e Inglês a partir dos dados de temperatura.

Nos anos 2000, a pesquisa em algoritmos de Aprendizado de Máquina possibilitou a aplicação de sistemas de geração de linguagem em tarefas de sumarização de informações, especialmente nas áreas de diagnóstico clínico (HARRIS, 2008; PORTET et al., 2009) e jornalismo "automático" (SAUPER; BARZILAY, 2009). Recentemente, o Aprendizado em Profundidade, do inglês *Deep Learning* (DL), tornou-se um marco para a geração de texto (GATT; KRAHMER, 2018) e introduziu técnicas capazes de melhor representar e descrever as características intrínsecas das linguagens naturais (DENG; LIU, 2018; OTTER et al., 2020; GALASSI et al., 2020).

No entanto, apesar de toda tarefa de geração de texto ser caracterizada pelo mapeamento de informações de entrada em textos de saída, Reiter e Dale (2002) constataram que a decomposição do processo de geração de texto facilita a sua compreensão e, consequentemente, dividiram-na nos seguintes estágios: determinação de conteúdo, estruturação de documentos, agregação, escolha léxica, geração de expressões de referência e realização.

2.1 Estágios da Geração

Para facilitar a compreensão de cada estágio, seja "Previsão do Tempo do Brasil" um exemplo de modelo de geração de texto. Este modelo recebe 5 números de entrada, os quais são predições de temperatura para cada região do Brasil. A partir destes números, o sistema é responsável por gerar um sumário textual contendo estas mesmas informações.

A determinação de conteúdo é o processo de decisão sobre qual informação deve ser utilizada no texto e geralmente está estruturada em entidades, conceitos e relações. No modelo de exemplo, é possível definir uma relação de previsão de temperatura por região, contendo as entidades temperatura e região. Logo, a determinação de conteúdo é responsável por explicitamente informar a temperatura em determinada região, por exemplo, 30 °C na região Norte. A estruturação de documentos é o processo de organização das informações, isto é, imposição da ordem e estrutura nas quais as informações definidas pelo estágio anterior devem ser apresentadas. No exemplo citado, tal estágio corresponde a organização de como as regiões serão citadas, ou seja, se serão citadas de acordo com a ordem crescente de temperatura ou por proximidade de região.

Posteriormente, o estágio de agregação combina diferentes relações (ou amostras de uma mesma relação) em sentenças únicas a fim de aprimorar a legibilidade e naturalidade do texto gerado. Por exemplo, as previsões de temperatura para cada região podem ser agregadas da seguinte forma: Ambas regiões Norte e Sudeste farão 30 °C ou mais no próximo final de semana. A escolha léxica é responsável por estruturar palavras e frases em conceitos, isto é,

decidir quais palavras e frases devem ser escolhidas para expressar corretamente o domínio das entidades e relações definidas. Por exemplo, as palavras “quente" e “frio" podem ser utilizadas para indicar uma referência de temperatura entre as regiões, onde regiões com níveis inferiores de temperaturas serão mais frias em relação àquelas com níveis superiores.

A geração de expressões de referência é o estágio de seleção de palavras ou frases para a identificação das entidades pertencentes ao domínio. Tal estágio está próximo do processo de escolha léxica, pois ambos são focados em produzir formas linguísticas capazes de identificar os elementos pertencentes ao domínio em questão. Por exemplo, a definição de lugares mais específicos dentro de uma determinada região (cidades ao leste da região Norte) ou localizações entre regiões (fronteira das regiões Norte e Nordeste) facilitam a compreensão da previsão gerada, dado que irão fornecer uma melhor descrição de onde realmente estão os níveis de temperatura indicados. Finalmente, o estágio de realização é a utilização correta de todas as regras sintáticas, morfológicas e gramaticais para a criação do texto. Este estágio é responsável por utilizar tempos verbais, conectivos, pronomes, plurais, pontuações, dentre outras regras que a linguagem possa exigir.

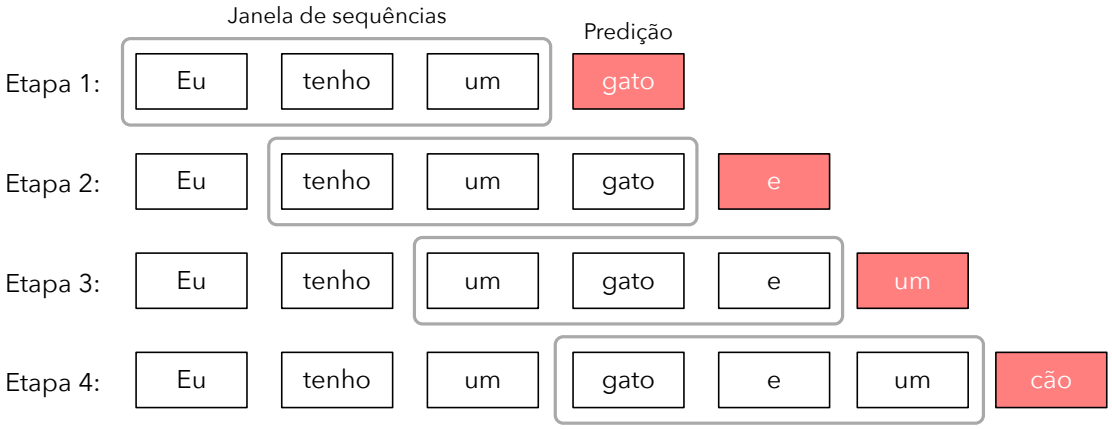
Alternativamente, abordagens recentes visam construir modelos linguísticos completos (DUŠEK et al., 2020), ou seja, modelos que não necessitam de estágios intermediários como os citados anteriormente. Estes modelos são treinados usando algoritmos de ML em grandes bases de dados e predizem informações com base nos exemplos aprendidos, aprendendo as regras gramaticais, sintáticas e semânticas oriundas dos exemplos utilizados no próprio processo de treinamento. Adicionalmente, também é possível verificar um novo paradigma sendo criado, onde modelos de geração neural vem substituindo tradicionais modelos de geração estatística.

2.2 Modelos Linguísticos

Os modelos linguísticos são utilizados como representações digitais de linguagens e de suas regras, isto é, eles compreendem estruturas gramaticais, sintáticas e semânticas pertencentes às linguagens. A maior parte dos modelos linguísticos são baseados em conceitos estatísticos e representados por distribuições probabilísticas, onde, por exemplo, dada uma sequência de tamanho m, o modelo linguístico assinala probabilidades $P(t_1, t_2, \ldots, t_m)$ para cada *token* t da sequência (MULDER et al., 2015). Em outras palavras, um modelo linguístico estima a probabilidade de ocorrência de *tokens* baseado em exemplos, onde modelos simples trabalham ao nível de caracteres ou palavras, enquanto modelos complexos utilizam sentenças ou parágrafos. Todavia, tal tarefa é complicada dado que os modelos precisam aprender características intrínsecas às linguagens naturais, tais como, gramática, sintaxe e semântica. Adicionalmente, o processo de treinamento é baseado em um conjunto de exemplos, não sendo prático para estimar dados não-existentes, por exemplo, *tokens* não encontrados no conjunto de dados. A Figura 2 ilustra um exemplo simplificado do processo de geração de texto utilizando

um modelo linguístico.

Figura 2 – Processo de geração de texto a partir de um modelo linguístico.



Fonte: Elaborado pelo autor.

Comumente, os modelos linguísticos são utilizados como arquiteturas-base em tarefas complexas de Processamento de Linguagem Natural, tais como, reconhecimento de voz, reconhecimento ótico de caracteres, tradução automática, correção gramatical, dentre outras. Por outro lado, um modelo linguístico pré-treinado pode ser utilizado para a geração de texto de acordo com as estruturas que foram aprendidas. Atualmente, os tipos mais comuns de modelos linguísticos são: unigrama, n-grama, exponencial e neural (NASEEM et al., 2021).

Modelo Linguístico Unigrama

Os modelos linguísticos baseados em unigramas utilizam apenas as probabilidades de ocorrência de cada palavra para definir suas probabilidades, ou seja, este modelo pode ser interpretado por diversos autômatos finitos de estados únicos. Por exemplo, seja t_1, t_2 e t_3 os *tokens* presentes em uma determinada sequência, enquanto $P(\cdot)$ é a probabilidade de ocorrência de cada *token*. A Equação 1 descreve a formulação matemática deste modelo:

$$P(t_1 t_2 t_3) = P(t_1)P(t_2 \mid t_1)P(t_3 \mid t_1 t_2) = P(t_1)P(t_2)P(t_3). \tag{1}$$

Modelo Linguístico n-Gramas

Os modelos baseados em n-gramas são estruturados através da Suposição de Markov (FOSLER-LUSSIER, 1998), ou seja, a probabilidade de um token i dado $i-1$ *tokens* anteriores pode ser estimada através da observação de um contexto histórico de $n-1$ *tokens*. Matematicamente falando, é possível reescrever a probabilidade de observação de uma sequência de m *tokens*, dada por $P(\cdot)$, através das Equações 2 e 3:

$$P(t_1, t_2, \ldots, t_m) \approx \prod_{i=1}^{m} P(t_i \mid t_{i-(n-1)}, \ldots, t_{i-1}) \tag{2}$$

e

$$P(t_i \mid t_{i-(n-1)}, \ldots, t_{i-1}) = \frac{c(t_{i-(n-1)}, \ldots, t_{i-1}, t_i)}{c(t_{i-(n-1)}, \ldots, t_{i-1})}, \tag{3}$$

onde $c(\cdot)$ é uma função de contagem de ocorrência de *tokens*.

Modelo Linguístico Exponencial

Os modelos linguísticos exponenciais utilizam a entropia máxima para codificar a relação entre uma determinada palavra e seu histórico de n-gramas, os quais são descritos pela Equação 4:

$$P(t_m \mid t_1, \ldots, t_{m-1}) = \frac{1}{Z(t_1, \ldots, t_{m-1})} e^{a^T f(t_1, \ldots, t_m)}, \tag{4}$$

onde $Z(t_1, \ldots, t_{m-1})$ é a função de partição, a é o vetor de parâmetros a serem aprendidos e $f(t_1, \ldots, t_m)$ é a função de características, usualmente representada pela própria ocorrência dos n-gramas.

Modelo Linguístico Neural

Os modelos linguísticos neurais são fomentados pelo desenvolvimento de redes neurais e são tidos como os atuais estado da arte devido à sua capacidade de generalização (BENGIO et al., 2003). Ademais, é comum encontrar em tais modelos a utilização de camadas de vetorização, isto é, camadas que codificam vetores altamente dimensionais de valores reais de *tokens* (caracteres ou palavras) em espaços vetoriais projetados[3].

Recentemente, Redes Neurais Recorrentes começaram a serem utilizadas para a modelagem de modelos linguísticos neurais, dado que elas são capazes de estruturar dependências temporais e re-utilizar seus estados ocultos. Um modelo linguístico baseado em RNNs provê alta generalização, pois seus neurônios são alimentados por conexões recorrentes, criando uma

[3] Tal abordagem é utilizada para atacar o problema da dimensionalidade, causado pela utilização de vocabulários grandes e responsável por produzir representações esparsas.

espécie de “memória". Por exemplo, uma rede neural tradicional não é o melhor modelo para modelar uma distribuição probabilística capaz de saber que <verbo> deve vir antes de um <substantivo>, pois ela utiliza apenas informações “futuras". Por outro lado, Redes Neurais Recorrentes são capazes de utilizar suas “memórias" e decidir as próximas saídas baseadas em saídas anteriores.

Esses modelos são estruturados como classificadores probabilísticos, os quais objetivam estimar a distribuição probabilística $P(t_i \mid \kappa)$, onde t_i representa os i *tokens* a serem estimados dado um contexto κ. Tipicamente, estes modelos são treinados em uma janela fixa de contexto através de algoritmos tradicionais de Aprendizado de Máquina, tais como, gradiente estocástico e retropropagação. A Equação 5 descreve a formulação matemática destes modelos:

$$P(t_i \mid \kappa) = P(t_i \mid t_{i-k}, \ldots t_{i-1}), \tag{5}$$

onde k representa a quantidade de *tokens* anteriores a serem considerados. A partir do cálculo da distribuição probabilística, é possível utilizar uma estratégia de amostragem para selecionar um *token*. Tal abordagem objetiva aumentar a generalização do modelo linguístico, bem como introduzir um fator de aleatoriedade capaz de promover a diversidade da geração artificial de texto.

2.3 Amostragem de Distribuições Probabilísticas

Os modelos linguísticos, independentes de serem unigrama, n-gramas, neurais, dentre outros, necessitam de uma estratégia de amostragem de distribuição probabilística para selecionar o *token* a ser gerado. Como mencionado anteriormente, tal metodologia possibilita a introdução de diversidade no processo de geração de texto e objetiva mitigar a geração de textos repetitivos.

Busca Gulosa

A técnica de amostragem mais comum é a chamada Busca Gulosa, a qual seleciona um *token* que possua a maior probabilidade dentro da distribuição probabilística, conforme definido pela Equação 6:

$$t_i = \max P(t_i \mid \kappa), \tag{6}$$

onde $P(\cdot)$ representa a distribuição probabilística de todos os possíveis *tokens* a serem gerados. Entretanto, tal técnica é responsável por gerar textos repetitivos dado que sempre seleciona a maior probabilidade, e também“ignora" palavras com menores probabilidades mas que sejam mais adequadas considerando-se o contexto.

Amostragem por Temperatura

Alternativamente, a Amostragem por Temperatura consiste em amostrar aleatoriamente um *token* de acordo com a sua distribuição probabilística. Tal fator de aleatoriedade é controlado por um parâmetro de temperatura, o qual possibilita ajustar a distribuição probabilística e controlar a geração tanto de palavras fortes (alta probabilidade) como de palavras fracas (baixa probabilidade). Esta técnica é descrita pela Equação 7, como segue:

$$t_i \sim \frac{P(t_i \mid \kappa)}{\tau}, \tag{7}$$

onde $P(\cdot)$ representa a distribuição probabilística de todos os possíveis *tokens* a serem gerados e normalizada por uma temperatura τ. À medida que $\tau \to 0$, a Amostragem por Temperatura assemelha-se à Busca Gulosa e passa a sofrer de seus mesmos problemas.

Amostragem de k-*Tokens*

Fan et al. (2018) introduziram um simples conceito, denotado por Amostragem de k-*Tokens*, onde a cada etapa de geração, os k *tokens* mais prováveis são filtrados e a sua distribuição probabilística é re-calculada. Tal processo é descrito pela Equação 8, como segue:

$$t_i \sim P(t_i \mid \kappa) \mid t \in \mathcal{V}_{top-k}, \tag{8}$$

onde $\mathcal{V}_{top-k}$ representa o conjunto dos k *tokens* mais prováveis e $P(\cdot)$ é a distribuição probabilística re-calculada para estes mesmos k *tokens*.

Amostragem Nuclear

A última técnica consiste na Amostragem Nuclear (HOLTZMAN et al., 2020), a qual amostra um *token* a partir do menor conjunto possível de *tokens* que possuam sua probabilidade cumulativa inferior à p. Por exemplo, seja uma distribuição probabilística $P(t)$ representada por $[0,7;\ 0,15;\ 0,1;\ 0,05]$ e t representado pelo seguinte conjunto de *tokens*: bonito, cachorro, carro, pessoa.

A Amostragem Nuclear calcula a probabilidade cumulativa C da distribuição probabilística $P(t)$, a qual é descrita pela Equação 9:

$$C = \sum P(t), \tag{9}$$

resultando em $[0,7;\ 0,85;\ 0,95;\ 1,0]$ para o exemplo supracitado. Posteriormente, dada uma probabilidade p, por exemplo, $p = 0,93$, o menor conjunto de *tokens* é selecionado, isto é, os dois primeiros *tokens* serão selecionados (bonito e cachorro) e representarão o conjunto $\mathcal{V}_{top-p}$.

Finalmente, é possível re-calcular a distribuição probabilística dos *tokens* pertencentes à $\mathcal{V}_{top-p}$ e realizar a amostragem final, descrita pela Equação 10:

$$t_i \sim P(t_i \mid \kappa) \mid t \in \mathcal{V}_{top-p}. \tag{10}$$

2.4 Métricas de Avaliação

As métricas de avaliação permitem uma análise quantitativa da geração de texto e compará-la perante às linguagens criadas pelos seres humanos. Desta forma, o surgimento de algoritmos focados em métricas de avaliação possibilitaram a estimação do quão "boas" as informações geradas são, bem como viabilizaram formas de mensurar a sua qualidade em relação aos outros algoritmos geradores de linguagem.

Bilingual Evaluation Understudy

A *Bilingual Evaluation Understudy* (BLEU) (PAPINENI et al., 2002) é uma métrica de qualidade de texto gerado por máquinas e é estruturada a partir da correspondência entre um texto fonte (escrito por um humano) e um texto artificial (gerado por uma máquina). Seu cálculo é realizado a partir da co-ocorrência de n-gramas entre a sentença de referência e as sentenças candidatas, computando assim, um tipo de precisão entre as sentenças. Seja $\Phi = \{\phi_1, \phi_2, \ldots, \phi_\kappa\}$ a sentença de referência com κ *tokens*, enquanto $\tilde{\Phi} = \{\tilde{\phi}_1, \tilde{\phi}_2, \ldots, \tilde{\phi}_\omega\}$ é a sequência candidata com ω *tokens*. A Equação 11 descreve a sua fórmula:

$$P_n = \frac{\sum_k \min(d_k(\tilde{\phi}), \max d_k(\phi))}{\sum_k d_k(\tilde{\phi})}, \tag{11}$$

onde $d_k(\cdot)$ é a quantidade de ocorrências de um n-grama e k é o número máximo de n-gramas possíveis. No entanto, tal formulação infla o valor da métrica para sentenças curtas e necessita de um fator de penalização de acordo com a Equação 12:

$$b = \begin{cases} 1 & \text{se } \omega > \kappa \\ e^{1-\frac{\kappa}{\omega}} & \text{se } \omega \leq \kappa \end{cases}. \tag{12}$$

Portanto, a pontuação final BLEU é calculada a partir de uma média geométrica ponderada entre a precisão dos n-gramas individuais. Ademais, o valor da métrica está sempre contido no intervalo $[0, 1]$, onde valores próximos a 1 indicam uma maior similaridade entre os textos. A Equação 13 descreve tal formulação:

$$B_N = b \cdot \exp((\sum_{n=1}^{N} w_n \log(P_n))), \tag{13}$$

onde $n = 1, 2, 3, \ldots, N$ e w_n é uma constante para todos os valores de n.

Comumente, não existe a necessidade de atingir valores iguais a 1 para a métrica BLEU, dado que tal valor representaria uma correspondência exata entre as sentenças avaliadas e nem sempre isto é atingindo por seres humanos. Gatt e Krahmer (2018) constataram que existe uma correlação significativa (por volta de $0,3$ a $0,87$) entre avaliações humanas e métricas automáticas, tais como BLEU e METEOR. Os mesmos autores constataram que tal correlação está mais próxima de um contexto gramatical do que semântico.

No entanto, mesmo sendo uma das métricas mais utilizadas para avaliação de qualidade de texto, a BLEU está limitada à uma análise de palavras, ou seja, a correspondência entre os n-gramas. Desta forma, ela é apenas adequada para comparações à nível de *corpus* (coleção de textos), não sendo tão efetiva quando avaliada em sentenças individuais. Ademais, ela é incapaz de considerar a semântica da sentença. Este problema refere-se à utilização em linguagens morfologicamente ricas, dado que morfemas podem alterar a semântica das sentenças e mesmo assim produzir altas correspondências de palavras, inflando o valor da métrica.

Metric for Evaluation of Translation with Explicit Ordering

A *Metric for Evaluation of Translation with Explicit Ordering* (METEOR) (BANERJEE; LAVIE, 2005) também avalia a qualidade de um algoritmo de tradução de máquina ou geração de texto, porém é baseada na média harmônica dos uni-gramas de precisão e sensitividade. Sua grande vantagem em relação à BLEU está na utilização do processo de *stemming* (redução de palavras flexionadas à raiz) e na correspondência de sinônimos, os quais permitem um melhor discernimento entre textos semanticamente coerentes e incoerentes.

A METEOR é calculada a partir do alinhamento entre uni-gramas presentes na sentença candidata e na sentença de referência. Este alinhamento é computado a partir da minimização de pedaços ch de *tokens* contíguos e igualmente ordenados no par de sentenças. Desta forma, o alinhamento é baseado na correspondência exata de *tokens* e é seguido de uma correspondência de sinônimos e *stemming* dos *tokens*. Dado um conjunto M de alinhamentos, é possível calcular a precisão P_M e a sensitividade R_M, formuladas pelas Equações 14 e 15, respectivamente:

$$P_M = \frac{|M|}{\sum_k d_k(\tilde{\phi})} \tag{14}$$

e

$$R_M = \frac{|M|}{\sum_k d_k(\phi)}, \tag{15}$$

onde $\tilde{\phi}$ é a sentença candidata, ϕ é a sentença de referência, $d_k(\cdot)$ é a quantidade de ocorrências de um n-grama e k é o número máximo de n-gramas possíveis.

Posteriormente, é calculada a média harmônica entre a precisão e sensitividade de acordo com a Equação 16:

$$F = \frac{P_M \cdot R_M}{\alpha P_M + (1-\alpha) R_M}, \tag{16}$$

onde $\alpha = 0,9$ é um parâmetro ajustável que define a proporção entre a precisão e a sensitividade.

Finalmente, é possível calcular o fator de penalização e a pontuação final de acordo com as Equações 17 e 18, respectivamente:

$$\Delta = \gamma(\frac{ch}{M})^{\theta} \tag{17}$$

e

$$M = (1-\Delta)F, \tag{18}$$

onde $\gamma = 0,5$ e $\theta = 3$ são parâmetros ajustáveis utilizados para emular o "julgamento" humano. Note que esta métrica também está contida no intervalo $[0, 1]$, onde valores próximos a 1 indicam uma maior similaridade entre as sentenças avaliadas.

Negative Log-Likelihood

A *Negative Log-Likelihood* (NLL) (ZHANG et al., 2020) é geralmente utilizada como função de perda no processo de aprendizado, objetivando maximizar a probabilidade de uma classificação correta. Em outras palavras, durante o processo de treinamento de um modelo linguístico, o NLL é utilizado para verificar se a rede está aprendendo a gerar *tokens* no estado $t+1$ dado os t estados anteriores.

Seja K o número de classes a serem classificadas, isto é, o número de possíveis *tokens* a serem gerados. A função de perda é definida através da Equação 19:

$$NLL = -\frac{1}{n} \sum_{\boldsymbol{x} \in \mathcal{X}} \sum_{k=1}^{K} y_k \cdot e^{a_k}, \tag{19}$$

onde n é o número de amostras no conjunto de treinamento $\mathcal{X}$, $\boldsymbol{x}$ é o dado de entrada, $\boldsymbol{y}$ é o rótulo verdadeiro no formato *one-hot encoding* e a é a saída da rede que está sendo treinada. A métrica NLL é utilizada principalmente para avaliar a qualidade do aprendizado da rede e está restrita a um intervalo positivo, isto é, $[0, +\infty]$. Adicionalmente, à medida que o NLL tende a 0, a rede está sendo capaz de aprender as informações do conjunto de treinamento, ou seja, a rede está inferindo corretamente os rótulos das amostras de treino.

Recall-Oriented Understudy for Gisting Evaluation

A métrica *Recall-Oriented Understudy for Gisting Evaluation* (ROUGE) (LIN, 2004) foi originalmente desenvolvida para a avaliação de algoritmos de sumarização de texto. Essencialmente, esta métrica é computada de uma forma similar ao BLEU, só que orientada à sensitividade, pois verifica a ocorrência de n-gramas entre uma sentença candidata e sentenças de referência. Adicionalmente, análoga à BLEU, a métrica ROUGE está contida no intervalo $[0, 1]$, onde valores próximos a 1 indicam uma maior similaridade entre as sentenças avaliadas. Deste modo, a Equação 20 descreve a sua formulação:

$$R_n = \frac{\sum_k \min(d_k(\tilde{\phi}), \max d_k(\phi))}{\sum_k d_k(\phi)}, \tag{20}$$

onde $\tilde{\phi}$ é a sentença candidata, ϕ é a sentença de referência, $d_k(\cdot)$ é a quantidade de ocorrências de um n-grama e k é o número máximo de n-gramas possíveis.

3 Redes Neurais

Um dos grandes avanços da área de Aprendizado de Máquina foi o desenvolvimento das Redes Neurais, as quais simulam o funcionamento intrínseco do cérebro humano e aprendem padrões a partir de conjuntos de dados. Usualmente, uma Rede Neural é constituída por neurônios divididos em camadas de entrada, ocultas e de saída. Os neurônios, de forma análoga ao cérebro, são conectados através de sinapses (pesos) que propagam ou inibem a quantidade de informação que possuem. Tradicionalmente, as Redes Neurais são estruturadas através de conexões uni-direcionais, onde neurônios de uma camada L comunicam-se apenas com neurônios de uma camada $L+1$, e são treinadas através de um algoritmo de Retropropagação, o qual estima gradientes de atualização de pesos através de derivadas de funções de perda[4].

A área de Processamento de Linguagem Natural utiliza dados sequenciais (temporais), necessitando de uma arquitetura que seja capaz de considerar tais informações. Desta forma, as Redes Neurais Recorrentes tornaram-se marcos para a resolução de problemas de NLP, pois são capazes de re-utilizar informações passadas para aprender novas informações. Entretanto, tais modelos são suscetíveis à adversarialidade, a qual insere dados artificiais no processo de treinamento e prejudica o desempenho do mesmo. Logo, o surgimento de uma gama de arquiteturas denotadas por Redes Adversariais Generativas permitiram um maior controle da manipulação adversarial. Estas redes são modeladas a partir de duas arquiteturas distintas, isto é, um gerador capaz de aprender a distribuição dos dados e gerar dados artificiais, e um discriminador, capaz de estimar a probabilidade de um dado pertencer ao conjunto real ao invés do conjunto artificial.

Não obstante, o treinamento simultâneo de duas arquiteturas sofre com problemas de instabilidade e convergência, principalmente por conta da geração de gradientes "fortes" (*vanishing gradients*) quando o discriminador não é bem treinado, ocasionando em uma dissipação do aprendizado do gerador. Portanto, o objetivo da presente tese é reduzir tal instabilidade através de uma função de similaridade aprendida por Redes Siamesas. Estas redes são modeladas com o intuito de aprender a discernir se um par de dados é similar ou não, aprimorando o processo de identificação entre dados reais e artificiais. As próximas subseções introduzem a fundamentação teórica sobre os três tipos de Redes Neurais utilizadas na presente tese, isto é, Redes Neurais Recorrentes, Redes Adversariais Generativas e Redes Siamesas.

[4] A função de perda é responsável por definir o objetivo do aprendizado, por exemplo, uma classificação de informações.

3.1 Redes Neurais Recorrentes

Os modelos linguísticos neurais são estruturados através de dependências temporais, isto é, são baseados em predições de estados $t+1$ dado t estados anteriores. Tal dependência configura um comportamento sequencial, o qual é modelado através de estruturas que consideram o fator tempo. As Redes Neurais Recorrentes conseguem solucionar esta dependência através da forma de conexão de seus neurônios, os quais formam grafos direcionados ao longo de sequências. Por exemplo, Sutskever et al. (2011) propuseram a utilização de RNNs para a criação de um modelo linguístico baseado em caracteres aplicado a tarefa de compressão de informação. Os autores, além de atingirem resultados estado da arte, observaram que tais modelos também foram capazes de gerar linguagem natural, entender o vocabulário da língua inglesa e aprender a balancear o uso de parênteses e aspas em sequências com por volta 30 caracteres.

Entretanto, a utilização do fator tempo durante o treinamento pode acarretar no problema da dissipação do gradiente. Tal problema ocorre quando o gradiente torna-se extremamente pequeno ao longo das iterações, prevenindo as atualizações de pesos e, em seu pior caso, inibindo a rede de ser treinada. Ademais, existe o problema do estouro do gradiente, o qual ocorre quando o gradiente torna-se demasiado grande ao longo das iterações, explodindo as atualizações de pesos durante o treinamento.

A criação de mecanismos de portões, utilizados nas arquiteturas das Redes de Memória de Longo Prazo e das Unidades Recorrentes de Porta, aliviaram o problema de dissipação do gradiente, pois eles possibilitam controlar a quantidade de informação aprendida a ser utilizada no futuro. Tais mecanismos funcionam como reguladores de informação e permitem com que a rede estruture suas camadas ocultas em formato de células de memória. Recentemente, Wen e Young (2020) apresentaram um gerador de linguagem natural baseado em LSTMs e demonstraram a sua capacidade em gerar diálogos com poucas repetições e erros semânticos, os quais eram problemas recorrentes no processo de geração de RNNs tradicionais. Adicionalmente, os autores demonstraram a capacidade do modelo proposto em adaptar-se a diferentes domínios de informação, predizendo informações não vistas em seu treinamento.

Alternativamente, um mecanismo moderno, denotado por atenção, vem obtendo resultados estado da arte em tarefas de geração de texto, sobretudo em arquiteturas baseadas em Transformadores (VASWANI et al., 2017). Este mecanismo é estruturado por um conjunto de projeções lineares ativadas por funções não-lineares, capazes de processar quais são os estados intermediários mais importantes da rede, ou seja, tal mecanismo permite com que a rede “atente-se" aos estados mais relevantes e produza vetores de contexto mais informativos. Desta forma, Santoro et al. (2018) combinaram conceitos de memória relacional, LSTMs e mecanismos de atenção para propor uma arquitetura denotada por Rede de Memória Relacional, do inglês *Relational Memory Core* (RMC). Esta arquitetura atingiu resultados estado da arte em tarefas de geração de texto, aprimorando os resultados obtidos pelas tradicionais RNNs e

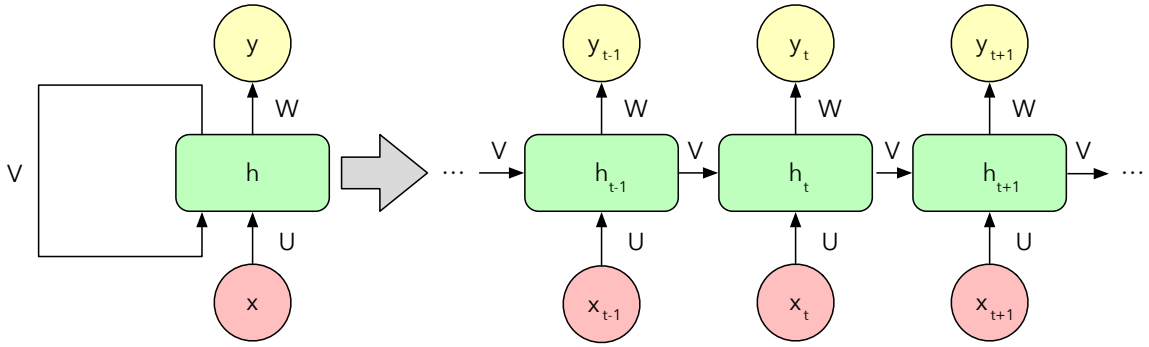
LSTMs.

As quatro arquiteturas recorrentes supracitadas, isto é, RNNs, LSTMs, GRUs e RMCs, são tidas como arquiteturas tradicionais no Processamento de Linguagem Natural, e mais especificamente, na tarefa de geração de texto. Portanto, as próximas subseções encarregam-se de descrever com maiores detalhes a formulação matemática e algoritmos de treinamento destas arquiteturas.

3.1.1 Rede Neural Recorrente de Elman

O exemplo mais simples de arquitetura recorrente é a Rede Neural Recorrente de Elman (ELMAN, 1990), a qual é ilustrada pela Figura 3. Esta rede possui nós de uma determinada camada L conectados diretamente aos nós da camada $L+1$ (camada posterior). Adicionalmente, cada nó possui uma função de ativação condicionada a um tempo (estado) t e conexões com pesos de valores reais.

Figura 3 – Arquitetura básica de uma Rede Neural Recorrente.



Fonte: Elaborado pelo autor.

A ativação da camada oculta deste modelo de rede neural é descrito pela Equação 21:

$$\boldsymbol{h}_t = \rho(\boldsymbol{U}\boldsymbol{x}_t + \boldsymbol{W}\boldsymbol{h}_{t-1} + \boldsymbol{b}), \tag{21}$$

em que $\boldsymbol{x}_t \in \mathbb{R}^n$ é a entrada da RNN, $\boldsymbol{h}_t \in \mathbb{R}^k$ é o estado oculto da RNN que funciona como a memória do tempo atual t, $\boldsymbol{U} \in \mathbb{R}^{n \times k}$ e $\boldsymbol{W} \in \mathbb{R}^{k \times k}$ correspondem aos pesos a serem aprendidos e $\boldsymbol{b} \in \mathbb{R}^k$ corresponde ao vetor de viés da rede. A função logística $\rho(x)$ é a mais comum para a ativação da camada oculta[5].

Adicionalmente, a camada de saída é descrita pela Equação 22:

$$\boldsymbol{y}_t = \boldsymbol{V}\boldsymbol{h}_t, \tag{22}$$

[5] Outras funções de ativação também podem ser utilizadas, como a Unidade Linear Retificada, do inglês *Rectified Linear Unit* (ReLU) (GLOROT et al., 2011), por exemplo.

onde $\boldsymbol{V} \in \mathbb{R}^{p \times k}$ é um parâmetro a ser aprendido pela rede e $\boldsymbol{y}_t \in \mathbb{R}^p$ é o vetor de saída da RNN. Note que n é número de neurônios de entrada, k o número de neurônios ocultos e p o número de neurônios de saída. O Algoritmo 1 descreve o pseudo-código de treinamento da Rede Neural Recorrente de Elman.

ALGORITMO 1: Pseudo-código de treinamento da Rede Neural Recorrente de Elman.

Entrada: Conjunto de treinamento $\mathcal{X} = \{(\boldsymbol{x}_1, \hat{y}_1), (\boldsymbol{x}_2, \hat{y}_2), \ldots, (\boldsymbol{x}_m, \hat{y}_m)\}$, função de perda loss$(\cdot)$, número de épocas E, número de estados T e taxa de treinamento η.

Salida: Parâmetros otimizados da RNN (θ^*).

1 $e \leftarrow 1$;
2 $\theta \leftarrow (\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}, \boldsymbol{b})$;
3 **mientras** $(e <= E)$ **hacer**
4 **para** t *de* 1 *até* t **hacer**
5 $\boldsymbol{h}_t \leftarrow \rho(\boldsymbol{U}\boldsymbol{x}_t + \boldsymbol{W}\boldsymbol{h}_{t-1} + \boldsymbol{b})$;
6 $\boldsymbol{y}_t \leftarrow \boldsymbol{V}\boldsymbol{h}_t$;
7 $R \leftarrow \nabla_\theta$ loss$(\boldsymbol{y}, \hat{\boldsymbol{y}})$;
8 $\theta \leftarrow \theta - \eta \cdot$ Adam(θ, R);
9 $e \leftarrow e + 1$;
10 $\theta^* \leftarrow \theta$

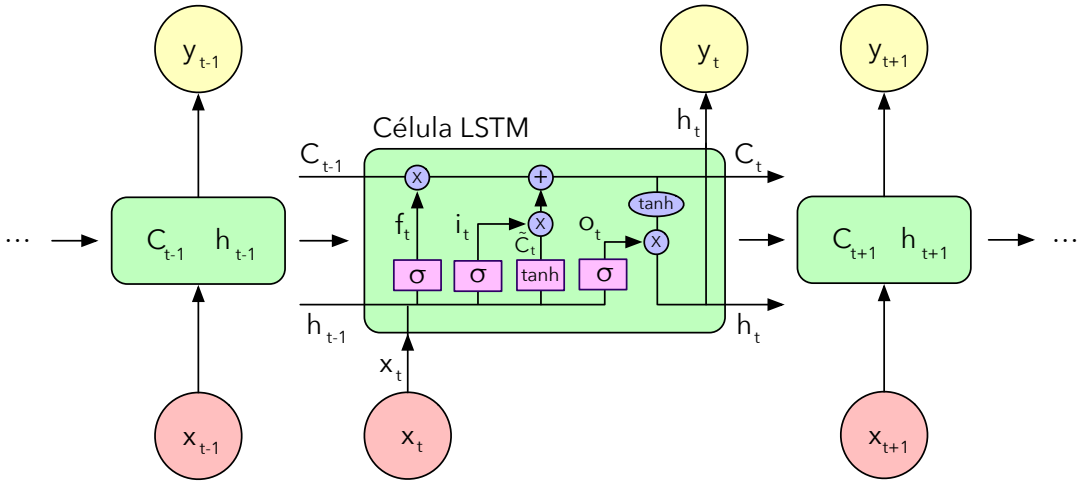
A linha 1 define o início da época, enquanto a linha 2 define os parâmetros a serem otimizados. As linhas 3-9 definem uma época de treinamento, onde cada estado oculto e saída da rede são atualizados pelas linhas 5 e 6, respectivamente, os gradientes são calculados pela linha 7, os parâmetros são atualizados pela linha 8, e a linha 9 incrementa o contador de épocas. Por fim, a linha 10 atribui os parâmetros pós-treinamento como os parâmetros otimizados da rede.

3.1.2 Rede de Memória de Longo Prazo

As Redes de Memória de Longo Prazo propostas por Hochreiter e Schmidhuber (1997) são um tipo especial de RNNs, explicitamente projetadas para aprenderem informações por longos períodos de tempo. Assim como as tradicionais RNNs, elas possuem uma arquitetura que se repete ao longo de um período de tempo t. Contudo, sua principal diferença está na camada oculta, a qual possui uma célula (unidade) com quatro entidades que interagem entre si. Esta célula é ilustrada pela Figura 4.

Uma célula de LSTM é representada pelas seguintes variáveis: estado da célula $\boldsymbol{C}_t \in \mathbb{R}^k$, neurônio de entrada $\boldsymbol{x}_t \in \mathbb{R}^n$, camada escondida $\boldsymbol{h}_t \in \mathbb{R}^k$ e neurônio de saída $\boldsymbol{y}_t \in \mathbb{R}^p$. O estado da célula $\boldsymbol{C}_t$ funciona como uma esteira, percorrendo toda a correia e sofrendo apenas alterações lineares. A célula de LSTM tem a capacidade de adicionar ou remover informações contidas no estado da célula, cuidadosamente reguladas por estruturas chamadas de portões.

Figura 4 – Arquitetura de uma célula de uma Rede de Memória de Longo Prazo.



Fonte: Elaborado pelo autor.

Os portões são descritos como meios que permitem a passagem ou não de informações. Eles são compostos por camadas neurais com função de ativação sigmoide $\sigma(\cdot)$ e uma operação de multiplicação pontual. A saída da função sigmoide é um número real entre 0 e 1, descrevendo o quanto de cada componente deve ser levado em consideração. O valor 0 corresponde à nenhuma informação enquanto o valor 1 corresponde à informação transmitida de maneira integral. Uma célula tradicional de LSTM possui três portões, denominados de entrada, saída e esquecimento.

O primeiro passo de uma LSTM é determinar o quanto de informação será inutilizada no estado da célula. Esta decisão é efetuada pelo portão de esquecimento, formulado pela Equação 23:

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_f \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_f). \tag{23}$$

O próximo passo é decidir a quantidade de novas informações que devem ser armazenadas no estado da célula, a qual é efetuada em duas partes. Primeiramente, o portão de entrada decide quais valores serão atualizados utilizando a Equação 24. Posteriormente, a camada com função de ativação tangente hiperbólica $tanh(\cdot)$ cria um vetor com os novos valores candidatos $\tilde{\boldsymbol{C}}_t$, de acordo com a Equação 25:

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_i \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_i) \tag{24}$$

e

$$\tilde{\boldsymbol{C}}_t = tanh(\boldsymbol{W}_C \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_C). \tag{25}$$

Após o cálculo de $\tilde{\boldsymbol{C}}_t$, é necessário atualizar o estado antigo da célula $\boldsymbol{C}_{t-1}$, o qual gera um novo estado denotado por $\boldsymbol{C}_t$. Tal operação é descrita pela Equação 26:

$$\boldsymbol{C}_t = \boldsymbol{f}_t \odot \boldsymbol{C}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{C}}_t, \tag{26}$$

onde $\odot$ representa o produto de Hadamard.

O penúltimo passo é a decisão sobre a saída da célula. O portão de saída é responsável por filtrar o estado da célula utilizando a Equação 27 e a camada oculta é atualizada de acordo com a Equação 28:

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_o \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_o) \tag{27}$$

e

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot tanh(\boldsymbol{C}_t). \tag{28}$$

Note que as variáveis $\boldsymbol{W}_f \in \mathbb{R}^{k \times k}$, $\boldsymbol{W}_i \in \mathbb{R}^{k \times k}$, $\boldsymbol{W}_C \in \mathbb{R}^{k \times k}$ e $\boldsymbol{W}_o \in \mathbb{R}^{k \times k}$ denotam as matrizes de pesos, e as variáveis $\boldsymbol{b}_f \in \mathbb{R}^k$, $\boldsymbol{b}_i \in \mathbb{R}^k$, $\boldsymbol{b}_C \in \mathbb{R}^k$ e $\boldsymbol{b}_o \in \mathbb{R}^k$ são os vetores de viés para cada um dos portões.

Finalmente, a camada de saída pode ser calculada através da Equação 29:

$$\boldsymbol{y}_t = \boldsymbol{V}\boldsymbol{h}_t, \tag{29}$$

onde $\boldsymbol{V} \in \mathbb{R}^{p \times k}$ é a matriz de pesos da saída e $\boldsymbol{y}_t \in \mathbb{R}^p$ é o vetor de saída da LSTM. Note que n é número de neurônios de entrada, k o número de neurônios ocultos e p o número de neurônios de saída. O Algoritmo 2 descreve o pseudo-código de treinamento da Rede de Memória de Longo Prazo.

A linha 1 define o início da época, enquanto a linha 2 define os parâmetros a serem otimizados. As linhas 3-14 definem uma época de treinamento, onde os portões da rede são atualizados pelas linhas 5-9, seu estado oculto e saída são atualizados pelas linhas 10 e 11, respectivamente, os gradientes são calculados pela linha 12, os parâmetros são atualizados pela linha 13, e a linha 14 incrementa o contador de épocas. Por fim, a linha 15 atribui os parâmetros pós-treinamento como os parâmetros otimizados da rede.

3.1.3 Unidade Recorrente de Porta

As Unidades Recorrentes de Porta introduzidas por Cho et al. (2014) também procuram solucionar o problema da dissipação ou estouro do gradiente e são vistas como variações das LSTMs, dado que suas arquiteturas são extremamente similares e, em alguns casos, produzem resultados igualmente robustos.

A fim de solucionar o problema de dissipação ou estouro do gradiente, a GRU[6] utiliza uma estrutura composta por dois portões, denotados por atualização e reinicialização. Basicamente,

[6] Novamente, $\boldsymbol{x}_t \in \mathbb{R}^n$ é o neurônio de entrada, $\boldsymbol{h}_t \in \mathbb{R}^k$ é a camada oculta, $\boldsymbol{y}_t \in \mathbb{R}^p$ é o neurônio de saída, $\boldsymbol{W} \in \mathbb{R}^{k \times k}$ é a matriz de pesos e $\boldsymbol{b} \in \mathbb{R}^k$ é o vetor de viés.

ALGORITMO 2: Pseudo-código de treinamento da Rede de Memória de Longo Prazo.

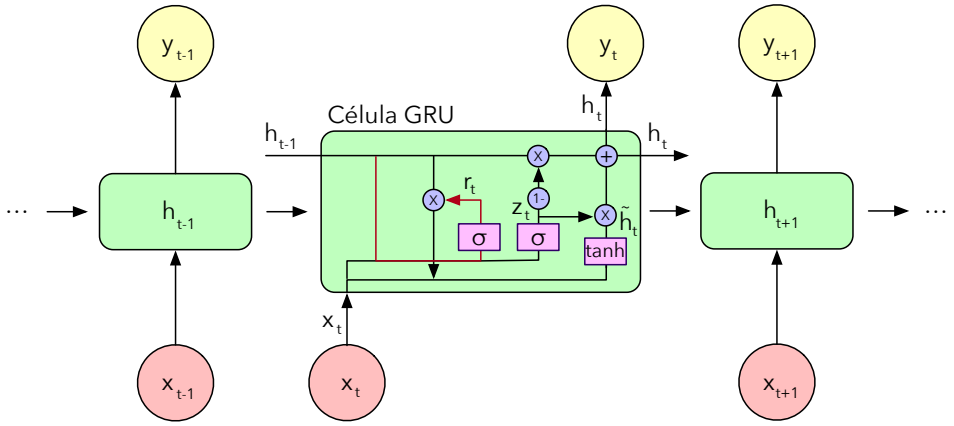
Entrada: Conjunto de treinamento $\mathcal{X} = \{(\boldsymbol{x}_1, \hat{y}_1), (\boldsymbol{x}_2, \hat{y}_2), \ldots, (\boldsymbol{x}_m, \hat{y}_m)\}$, função de perda loss$(\cdot)$, número de épocas E, número de estados T e taxa de treinamento η.

Salida: Parâmetros otimizados da LSTM (θ^*).

$e \leftarrow 1$;
$\theta \leftarrow \boldsymbol{W}_f, \boldsymbol{W}_i, \boldsymbol{W}_C, \boldsymbol{W}_o, \boldsymbol{b}_f, \boldsymbol{b}_i, \boldsymbol{b}_C, \boldsymbol{b}_o, \boldsymbol{V}$;
mientras $(e <= E)$ **hacer**
  para t *de* 1 *até* T **hacer**
    $\boldsymbol{f}_t \leftarrow \sigma(\boldsymbol{W}_f \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_f)$;
    $\boldsymbol{i}_t \leftarrow \sigma(\boldsymbol{W}_i \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_i)$;
    $\tilde{\boldsymbol{C}}_t \leftarrow tanh(\boldsymbol{W}_C \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_C)$;
    $\boldsymbol{C}_t \leftarrow \boldsymbol{f}_t \odot \boldsymbol{C}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{C}}_t$;
    $\boldsymbol{o}_t \leftarrow \sigma(\boldsymbol{W}_o \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_o)$;
    $\boldsymbol{h}_t \leftarrow \boldsymbol{o}_t \odot tanh(\boldsymbol{C}_t)$;
    $\boldsymbol{y}_t \leftarrow \boldsymbol{V}\boldsymbol{h}_t$;
  $L \leftarrow \nabla_\theta$ loss$(\boldsymbol{y}, \hat{\boldsymbol{y}})$;
  $\theta \leftarrow \theta - \eta \cdot$ Adam(θ, L);
  $e \leftarrow e + 1$;
$\theta^* \leftarrow \boldsymbol{W}_f, \boldsymbol{W}_i, \boldsymbol{W}_C, \boldsymbol{W}_o, \boldsymbol{b}_f, \boldsymbol{b}_i, \boldsymbol{b}_C, \boldsymbol{b}_o, \boldsymbol{V}$;

são dois vetores que decidem quais informações devem ser passadas para a saída da rede. Sua principal função, análoga às LSTMs, é reter informação de longos períodos ou remover informações irrelevantes sem que ocorram perdas significativas através do avanço do tempo. A Figura 5 ilustra uma Unidade Recorrente de Porta.

Figura 5 – Arquitetura de uma Unidade Recorrente de Porta.



Fonte: Elaborado pelo autor.

O primeiro passo nesta estrutura é calcular o portão de atualização $\boldsymbol{z}_t$ para o estado t, a partir da Equação 30:

$$\boldsymbol{z}_t = \sigma(\boldsymbol{W}_z \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_z). \tag{30}$$

Essencialmente, o portão de atualização determina a quantidade de informações passadas (t estados anteriores) que serão utilizadas nos estados seguintes, através de uma função de ativação sigmoide $\sigma(\cdot)$. Este portão é extremamente útil ao modelo, pois pode copiar ou não toda a informação "armazenada". Posteriormente, o portão de reinicialização é utilizado para decidir o quanto de informação do passado deve ser esquecido, sendo calculado pela Equação 31:

$$\boldsymbol{r}_t = \sigma(\boldsymbol{W}_r \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_r). \tag{31}$$

Com os dois portões atualizados, é possível calcular uma nova memória para o tempo atual, representada por $\tilde{\boldsymbol{h}}_t$ e calculada de acordo com a Equação 32:

$$\tilde{\boldsymbol{h}}_t = tanh(\boldsymbol{W}_h \cdot [\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_h). \tag{32}$$

O valor atual da célula $\boldsymbol{h}_t$, o qual é o vetor das informações atuais da unidade, é calculado pela Equação 33:

$$\boldsymbol{h}_t = (1 - \boldsymbol{z}_t) \odot \boldsymbol{h}_{t-1} + \boldsymbol{z}_t \odot \tilde{\boldsymbol{h}}_t. \tag{33}$$

Note que as variáveis $\boldsymbol{W}_z \in \mathbb{R}^{k \times k}$, $\boldsymbol{W}_r \in \mathbb{R}^{k \times k}$ e $\boldsymbol{W}_h \in \mathbb{R}^{k \times k}$ denotam as matrizes de pesos, e as variáveis $\boldsymbol{b}_z \in \mathbb{R}^k$, $\boldsymbol{b}_r \in \mathbb{R}^k$ e $\boldsymbol{b}_h \in \mathbb{R}^k$ são os vetores de viés para cada um dos portões.

Finalmente, a camada de saída pode ser calculada através da Equação 44:

$$\boldsymbol{y}_t = \boldsymbol{V}\boldsymbol{h}_t, \tag{34}$$

onde $\boldsymbol{V} \in \mathbb{R}^{p \times k}$ é a matriz de pesos da saída e $\boldsymbol{y}_t \in \mathbb{R}^p$ é o vetor de saída da GRU. Note que n é número de neurônios de entrada, k o número de neurônios ocultos e p o número de neurônios de saída. O Algoritmo 3 descreve o pseudo-código de treinamento da Unidade Recorrente de Porta.

A linha 1 define o início da época, enquanto a linha 2 define os parâmetros a serem otimizados. As linhas 3-12 definem uma época de treinamento, onde os portões da rede são atualizados pelas linhas 5 e 6, seu estado oculto e saída são atualizados pelas linhas 7-8 e 9, respectivamente, os gradientes são calculados pela linha 10, os parâmetros são atualizados pela linha 11, e a linha 12 incrementa o contador de épocas. Por fim, a linha 15 atribui os parâmetros pós-treinamento como os parâmetros otimizados da rede.

ALGORITMO 3: Pseudo-código de treinamento da Unidade Recorrente de Porta.

Entrada: Conjunto de treinamento $\mathcal{X} = \{(\boldsymbol{x}_1, \hat{y}_1), (\boldsymbol{x}_2, \hat{y}_2), \ldots, (\boldsymbol{x}_m, \hat{y}_m)\}$, função de perda loss$(\cdot)$, número de épocas E, número de estados T e taxa de treinamento η.

Salida: Parâmetros otimizados da GRU (θ^*).

$e \leftarrow 1$;
$\theta \leftarrow \boldsymbol{W}_z, \boldsymbol{W}_r, \boldsymbol{W}_h, \boldsymbol{b}_z, \boldsymbol{b}_r, \boldsymbol{b}_h, \boldsymbol{V}$;
mientras $(e <= E)$ **hacer**
  para t *de* 1 *até* T **hacer**
    $\boldsymbol{z}_t = \sigma(\boldsymbol{W}_z \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_z)$;
    $\boldsymbol{r}_t = \sigma(\boldsymbol{W}_r \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_r)$;
    $\tilde{\boldsymbol{h}}_t = tanh(\boldsymbol{W}_h \cdot [\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_h)$;
    $\boldsymbol{h}_t = (1 - \boldsymbol{z}_t) \odot \boldsymbol{h}_{t-1} + \boldsymbol{z}_t \odot \tilde{\boldsymbol{h}}_t$;
    $\boldsymbol{y}_t \leftarrow \boldsymbol{V}\boldsymbol{h}_t$;
  $G \leftarrow \nabla_\theta$ loss$(\boldsymbol{y}, \hat{\boldsymbol{y}})$;
  $\theta \leftarrow \theta - \eta \cdot$ Adam(θ, G);
  $e \leftarrow e + 1$;
$\theta^* \leftarrow \boldsymbol{W}_z, \boldsymbol{W}_r, \boldsymbol{W}_h, \boldsymbol{b}_z, \boldsymbol{b}_r, \boldsymbol{b}_h, \boldsymbol{V}$;

3.1.4 Rede de Memória Relacional

A Rede de Memória Relacional introduzida por Santoro et al. (2018) baseia-se em conceitos de raciocínio relacional[7] adaptados ao espaço temporal para realizar o aprendizado de informações. Por exemplo, a capacidade de comparação e contraposição de informações vistas em diferentes estados é um raciocínio relacional temporal, podendo ser modelado através de mecanismos de atenção (BAHDANAU et al., 2014). A célula proposta foi construída a partir da combinação de diferentes arquiteturas, tais como, LSTMs, Redes Neurais de Memória Ampliada (GRAVES et al., 2014) e Transformadores (VASWANI et al., 2017). Adicionalmente, ela utiliza um número fixo de compartimento de memórias e permite a interação entre as mesmas durante cada estado t. Esta célula é ilustrada pela Figura 6.

Seja $\boldsymbol{M}_t \in \mathbb{R}^k$ o estado da célula (memórias) e $\boldsymbol{x}_t \in \mathbb{R}^k$ o neurônio de entrada no tempo t. Inicialmente, o mecanismo de auto-atenção calcula três projeções lineares com tamanho de saída igual a l, descritas pelas Equações 35, 36 e 37:

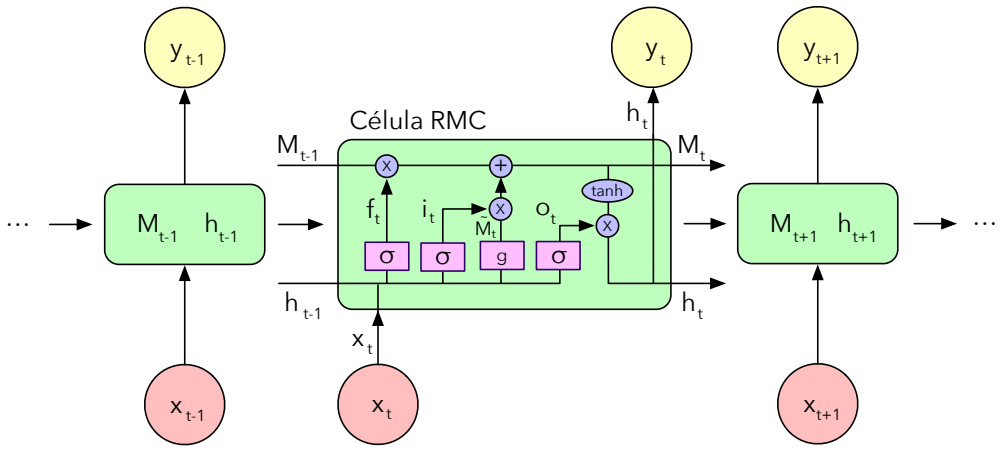
$$\boldsymbol{q} = \boldsymbol{W}_q \boldsymbol{M}_t, \tag{35}$$

$$\boldsymbol{k} = \boldsymbol{W}_k \cdot [\boldsymbol{M}_t, \boldsymbol{x}_t] \tag{36}$$

e

[7] Raciocínio relacional é o processo de compreensão do relacionamento entre entidades.

Figura 6 – Arquitetura de uma célula de uma Rede de Memória Relacional.



Fonte: Elaborado pelo autor.

$$\boldsymbol{v} = \boldsymbol{W}_v \cdot [\boldsymbol{M}_t, \boldsymbol{x}_t], \tag{37}$$

onde $\boldsymbol{q} \in \mathbb{R}^l$ é o vetor de consultas e $\boldsymbol{W}_q \in \mathbb{R}^{k \times l}$ seus pesos, $\boldsymbol{k} \in \mathbb{R}^l$ é o vetor de chaves e $\boldsymbol{W}_k \in \mathbb{R}^{k \times l}$ seus pesos, e $\boldsymbol{v} \in \mathbb{R}^n$ é o vetor de valores e $\boldsymbol{W}_v \in \mathbb{R}^{k \times l}$ seus pesos.

A atualização do estado da célula, denotada por $\tilde{\boldsymbol{M}}_t$, é calculada pelo mecanismo de auto-atenção, ilustrado pela Figura 7, através da Equação 38:

$$\tilde{\boldsymbol{M}}_t = \text{softmax}(\frac{\boldsymbol{q}\boldsymbol{k}^T}{\sqrt{l}})\boldsymbol{v}. \tag{38}$$

Após o cálculo de $\tilde{\boldsymbol{M}}_t$, é possível atualizar os portões de esquecimento, entrada e saída de acordo com as Equações 39, 40 e 41, respectivamente:

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_f \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_f), \tag{39}$$

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_i \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_i) \tag{40}$$

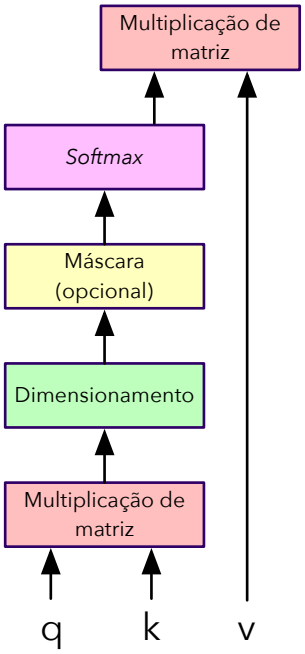
e

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_o \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_o), \tag{41}$$

onde $\boldsymbol{W}_f \in \mathbb{R}^{k \times k}$, $\boldsymbol{W}_i \in \mathbb{R}^{k \times k}$ e $\boldsymbol{W}_o \in \mathbb{R}^{k \times k}$ denotam as matrizes de pesos, e as variáveis $\boldsymbol{b}_f \in \mathbb{R}^k$, $\boldsymbol{b}_i \in \mathbb{R}^k$ e $\boldsymbol{b}_o \in \mathbb{R}^k$ são os vetores de viés para cada um dos portões.

Posteriormente, o estado atual da célula $\boldsymbol{M}_t$ e a camada oculta $\boldsymbol{h}_t \in \mathbb{R}^k$ são atualizados de acordo com as Equações 42 e 43, respectivamente:

Figura 7 – Mecanismo de auto-atenção.



Fonte: Elaborado pelo autor.

$$\boldsymbol{M}_t = \boldsymbol{f}_t \odot \boldsymbol{M}_{t-1} + \boldsymbol{i}_t \odot g(\tilde{\boldsymbol{M}}_t) \tag{42}$$

e

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot tanh(\boldsymbol{M}_t), \tag{43}$$

onde $\odot$ representa o produto de Hadamard e $g(\cdot)$ é uma função oriunda de uma rede Perceptron Multicamadas com normalização por camada.

Finalmente, a camada de saída pode ser calculada através da Equação 44:

$$\boldsymbol{y}_t = \boldsymbol{V}\boldsymbol{h}_t, \tag{44}$$

onde $\boldsymbol{V} \in \mathbb{R}^{p \times k}$ é a matriz de pesos da saída e $\boldsymbol{y}_t \in \mathbb{R}^p$ é o vetor de saída da RMC. Note que n é número de neurônios de entrada, k o número de neurônios ocultos e p o número de neurônios de saída. O Algoritmo 4 descreve o pseudo-código de treinamento da Rede de Memória Relacional.

A linha 1 define o início da época, enquanto a linha 2 define os parâmetros a serem otimizados. As linhas 3-17 definem uma época de treinamento, onde o mecanismo de atenção é calculado pelas linhas 5-7, a memória do estado é calculada pela linha 8, os portões da rede são atualizados pelas linhas 9-11, a memória é atualizada pela linha 12, o estado oculto e a saída são atualizados pelas linhas 13 e 14, respectivamente, os gradientes são calculados

ALGORITMO 4: Pseudo-código de treinamento da Rede de Memória Relacional.

Entrada: Conjunto de treinamento $\mathcal{X} = \{(\boldsymbol{x}_1, \hat{y}_1), (\boldsymbol{x}_2, \hat{y}_2), \ldots, (\boldsymbol{x}_m, \hat{y}_m)\}$, função de perda loss$(\cdot)$, número de épocas E, número de estados T e taxa de treinamento η.

Salida: Parâmetros otimizados da RMC (θ^*).

1 $e \leftarrow 1$;
2 $\theta \leftarrow \boldsymbol{W}_q, \boldsymbol{W}_k, \boldsymbol{W}_v, \boldsymbol{W}_f, \boldsymbol{W}_i, \boldsymbol{W}_o, \boldsymbol{b}_f, \boldsymbol{b}_i, \boldsymbol{b}_o, \boldsymbol{V}$;
3 **mientras** $(e <= E)$ **hacer**
4 **para** t *de* 1 *até* T **hacer**
5 $\boldsymbol{q} \leftarrow \boldsymbol{W}_q \boldsymbol{M}_t$;
6 $\boldsymbol{k} \leftarrow \boldsymbol{W}_k \cdot [\boldsymbol{M}_t, \boldsymbol{x}_t]$;
7 $\boldsymbol{v} \leftarrow \boldsymbol{W}_v \cdot [\boldsymbol{M}_t, \boldsymbol{x}_t]$;
8 $\tilde{\boldsymbol{M}}_t \leftarrow \text{softmax}(\frac{\boldsymbol{q}\boldsymbol{k}^T}{\sqrt{l}})\boldsymbol{v}$;
9 $\boldsymbol{f}_t \leftarrow \sigma(\boldsymbol{W}_f \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_f)$;
10 $\boldsymbol{i}_t \leftarrow \sigma(\boldsymbol{W}_i \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_i)$;
11 $\boldsymbol{o}_t \leftarrow \sigma(\boldsymbol{W}_o \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_o)$;
12 $\boldsymbol{M}_t \leftarrow \boldsymbol{f}_t \odot \boldsymbol{M}_{t-1} + \boldsymbol{i}_t \odot g(\tilde{\boldsymbol{M}}_t)$;
13 $\boldsymbol{h}_t \leftarrow \boldsymbol{o}_t \odot tanh(\boldsymbol{M}_t)$;
14 $\boldsymbol{y}_t \leftarrow \boldsymbol{V}\boldsymbol{h}_t$;
15 $R \leftarrow \nabla_\theta$ loss$(\boldsymbol{y}, \hat{\boldsymbol{y}})$;
16 $\theta \leftarrow \theta - \eta \cdot \text{Adam}(\theta, R)$;
17 $e \leftarrow e + 1$;
18 $\theta^* \leftarrow \boldsymbol{W}_q, \boldsymbol{W}_k, \boldsymbol{W}_v, \boldsymbol{W}_f, \boldsymbol{W}_i, \boldsymbol{W}_o, \boldsymbol{b}_f, \boldsymbol{b}_i, \boldsymbol{b}_o, \boldsymbol{V}$;

pela linha 15, os parâmetros são atualizados pela linha 16, e a linha 17 incrementa o contador de épocas. Por fim, a linha 18 atribui os parâmetros pós-treinamento como os parâmetros otimizados da rede.

3.2 Redes Adversariais Generativas

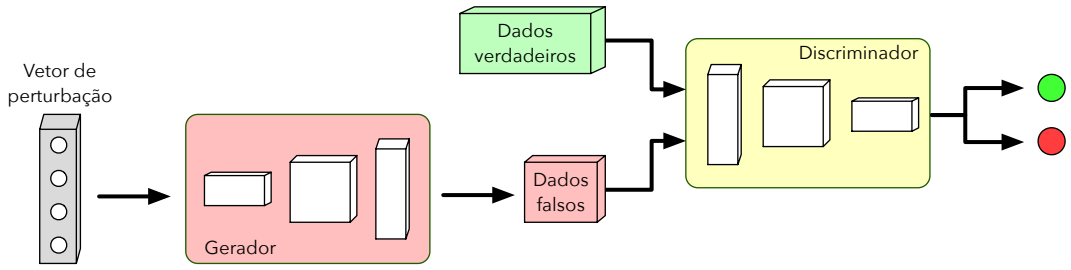
Recentemente, Goodfellow et al. (2014) propuseram as Redes Adversariais Generativas, as quais são implementadas por um sistema de duas redes neurais contestando-se em uma abordagem de soma zero[8]. A ideia por trás desta arquitetura é utilizar uma rede discriminativa em conjunto com uma rede generativa, onde a parte generativa é responsável por gerar dados artificiais a partir de ruídos e a rede discriminativa estimar a probabilidade das amostras (reais e artificiais) pertencerem aos dados reais.

Conforme mencionado, uma Rede Adversarial Generativa é composta por duas arquiteturas: gerador e discriminador. Inicialmente, seja a função diferenciável da arquitetura do gerador definida por G e seus parâmetros definidos por θ_G. Para aprender a distribuição probabilística do gerador sobre um conjunto de dados $\boldsymbol{x} \in \mathcal{X}$, é preciso definir um vetor

[8] Representação matemática (oriunda da área de Teoria de Jogos) de uma situação em que ganhos ou perdas de um candidato são exatamente balanceados pelos ganhos ou perdas de outros candidatos.

de ruído (perturbação) $\boldsymbol{z}$ e representar um mapeamento do espaço de características para o espaço de dados, tal como $G(\boldsymbol{z};\theta_G)$. Adicionalmente, seja D a arquitetura representada pelo discriminador e seus parâmetros definidos por θ_D, onde $D(\boldsymbol{x};\theta_D)$ é a função que representa a saída desta rede e que fornece um valor escalar. Note que $D(\boldsymbol{x})$ representa a probabilidade de um dado $\boldsymbol{x}$ pertencer ao espaço de dados $\mathcal{X}$ ao invés de pertencer ao espaço de dados artificiais. Uma sucinta ilustração desta arquitetura é representada pela Figura 8.

Figura 8 – Arquitetura padrão de uma Rede Adversarial Generativa.



Fonte: Elaborado pelo autor.

O discriminador da Rede Adversarial Generativa, D, é treinado para maximizar a probabilidade de classificação de dados pertencentes ao conjunto real, isto é, inferir corretamente os rótulos dos dados de treinamento e dos artificiais. Simultaneamente, G é treinado para minimizar $log(1 - D(G(\boldsymbol{z}))$. Em outras palavras, o sistema de duas redes neurais contestam-se utilizando uma abordagem de zero-soma, representada pela Equação 45:

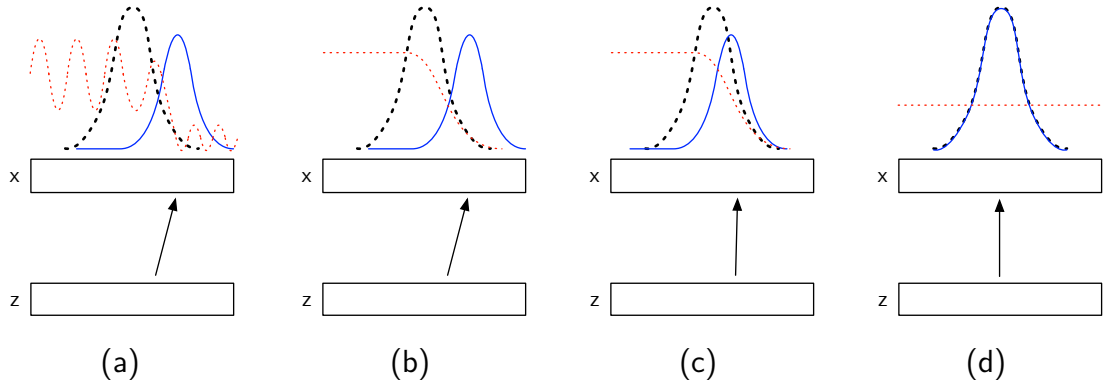
$$\min_G \max_D C(D,G) = \mathbb{E}_{\boldsymbol{x}}[logD(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}}[log(1 - D(G(\boldsymbol{z}))], \tag{45}$$

onde $C(D,G)$ é a função de custo a ser minimizada, $D(\boldsymbol{x})$ é a probabilidade estimada pelo discriminador de uma amostra real $\boldsymbol{x}$ ser real, $\mathbb{E}_{\boldsymbol{x}}$ é a esperança matemática sobre todas as instâncias do conjunto de dados real $\mathcal{X}$, $G(\boldsymbol{z})$ é a saída do gerador dado o vetor de perturbação $\boldsymbol{z}$, $D(G(\boldsymbol{z}))$ é a probabilidade estimada pelo discriminador de uma amostra falsa ser real, e $\mathbb{E}_{\boldsymbol{z}}$ é a esperança matemática sobre todas as entradas aleatórias no gerador, isto é, o valor esperado sobre todas as amostras falsas geradas por $G(\boldsymbol{z})$. A Figura 9 ilustra a estabilização do treinamento de uma Rede Adversarial Generativa, onde o discriminador (linha pontilhada vermelha) classifica a distribuição dos dados reais (linha pontilhada preta) e dos dados gerados (linha azul). Adicionalmente, $\boldsymbol{z}$ é o vetor de ruído, $\boldsymbol{x}$ é o vetor de dado real e as setas indicam o mapeamento de $G(\boldsymbol{z}) = \boldsymbol{x}$.

Entretanto, um problema oriundo da Equação 45 é que a GAN pode ficar presa aos ótimos locais quando a tarefa do discriminador é muito fácil. Em outras palavras, no início do treinamento, quando G ainda não obteve uma parametrização adequada, D pode rejeitar as amostras geradas com alta confiabilidade, dado que elas são extremamente diferentes do conjunto de dados reais[9] Portanto, uma alternativa à este problema é treinar G para maximizar

[9] Este problema satura a função $log(1 - D(G(\boldsymbol{z}))$.

Figura 9 – Estabilização do treinamento de uma Rede Adversarial Generativa: (a) distribuição de dados gerados é diferente da distribuição de dados reais e o discriminador ainda não foi treinado, (b) discriminador está sendo treinado para estabilizar o gerador, (c) gradiente do discriminador guiou o gerador à uma região mais propensa a mapear os dados reais e (d) ao fim do treinamento, se o gerador e o discriminador estabilizarem, chegarão à um ponto onde o discriminador não conseguirá diferenciar as distribuições.



Fonte: Elaborado pelo autor.

a função $log(D(G(\boldsymbol{z}))$, a qual provê gradientes mais robustos durante o início do treinamento. O Algoritmo 5 descreve o pseudo-código de treinamento da Rede Adversarial Generativa.

ALGORITMO 5: Pseudo-código de treinamento da Rede Adversarial Generativa.

Entrada: Conjunto de treinamento $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$, discriminador $D(\boldsymbol{x}; \theta_D)$, gerador $G(\boldsymbol{z}; \theta_G)$, número de épocas E_A, número de passos E_{A_D}, tamanho do lote l e taxa de treinamento η.

Salida: Parâmetros otimizados do discriminador (θ_D^*) e gerador (θ_G^*).

1 $e \leftarrow 1$;
2 **mientras** $(e <= E_A)$ **hacer**
3 **para** i *de* 1 *até* E_{A_D} **hacer**
4 Amostrar $\{G(\boldsymbol{z}_1), G(\boldsymbol{z}_2), \ldots, G(\boldsymbol{z}_l\})$ da distribuição artificial;
5 Amostrar $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_l\}$ da distribuição real;
6 $D \leftarrow \nabla_{\theta_D} \frac{1}{l} \sum_{j=1}^{l} [\log D(\boldsymbol{x}_j) + \log(1 - D(G(\boldsymbol{z}_j)))]$;
7 $\theta_D \leftarrow \theta_D - \eta \cdot \text{Adam}(\theta_D, D)$;
8 Amostrar $\{G(\boldsymbol{z}_1), G(\boldsymbol{z}_2), \ldots, G(\boldsymbol{z}_l\})$ da distribuição artificial;
9 $G \leftarrow \nabla_{\theta_G} \frac{1}{l} \sum_{j=1}^{l} [\log(1 - D(G(\boldsymbol{z}_j)))]$;
10 $\theta_G \leftarrow \theta_G - \eta \cdot \text{Adam}(\theta_G, G)$;
11 $e \leftarrow e + 1$;
12 $\theta_D^* \leftarrow \theta_D$;
13 $\theta_G^* \leftarrow \theta_G$;

A linha 1 define o início da época, enquanto as linhas 2-11 definem uma época de treinamento, onde os dados falsos e reais são amostrados pelas linhas 4 e 5, respectivamente, e os gradientes do discriminador são calculados e usados para atualizar os parâmetros do mesmo

pelas linhas 6 e 7. A linha 8 amostra novamente dados falsos, os quais serão utilizados para calcular os gradientes do gerador (linha 9) e atualizar os seus parâmetros (linha 10). Por fim, a linha 11 incrementa o contador de épocas, enquanto as linhas 12 e 13 atribuem os parâmetros pós-treinamento como os parâmetros otimizados do discriminador e gerador, respectivamente.

3.2.1 Aprendizado por Reforço

Para realizar o treinamento de uma Rede Adversarial Generativa, o discriminador e o gerador necessitam de funções de perda diferenciáveis. Entretanto, quando RNNs são utilizadas como geradoras das GANs, é impossível diferenciar suas saídas, dado que são usualmente obtidas através da amostragem de valores categóricos de uma distribuição *Softmax*.

Uma solução para tal problema consiste na criação de um agente de Aprendizado por Reforço, do inglês *Reinforcement Learning* (RL), para o gerador. Essencialmente, um algoritmo de RL é definido por uma função diretriz $\pi(\boldsymbol{a} \mid s, \theta_p)$ parametrizada por θ_p, a qual define uma distribuição probabilística sobre todas as ações $\boldsymbol{a}$ de um agente, dado seu estado atual s. Note que s é o estado do agente (texto gerado) e $\boldsymbol{a}$ são suas ações (textos a serem gerados). Em outras palavras, um agente gera linguagem natural baseado em uma diretriz π, onde suas ações são amostradas de acordo com a distribuição definida pela mesma diretriz. Portanto, é possível otimizar θ_p através de um método de gradiente de diretriz, por exemplo, o algoritmo REINFORCE (WILLIAMS, 1992).

3.2.2 Diferenciação por *Gumbel-Softmax*

Como mencionado anteriormente, um problema das Redes Adversariais Generativas é a necessidade de uma função de perda diferenciável para realizar o seu treinamento. Uma alternativa ao Aprendizado por Reforço consiste na solução proposta por Jang et al. (2016), a qual utiliza valores de saída amostrados de um função *Gumbel-Softmax*. Esta função utiliza conjuntamente as distribuições *Softmax* e *Gumbel*, bem como um parâmetro de temperatura para criar uma distribuição contínua e que assemelha-se a uma distribuição categórica. A Equação 46 descreve esta operação:

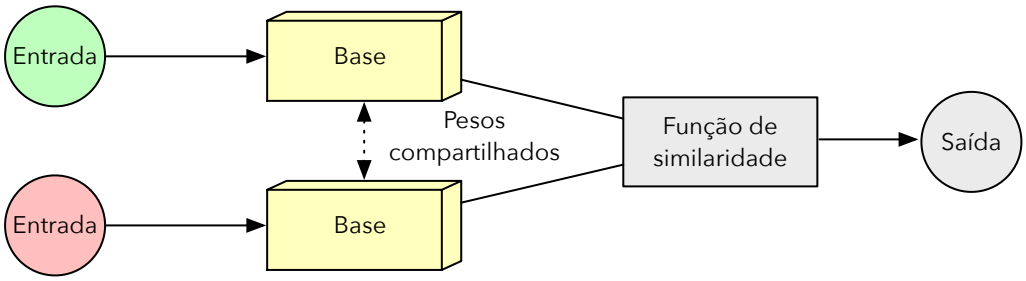
$$\tilde{\boldsymbol{y}} = \frac{e^{\frac{\boldsymbol{w}+\boldsymbol{g}}{\Delta}}}{\sum e^{\frac{\boldsymbol{w}+\boldsymbol{g}}{\Delta}}}, \tag{46}$$

onde $\tilde{\boldsymbol{y}}$ são as saídas diferenciáveis, $\boldsymbol{w}$ é um vetor de probabilidades logarítmicas não-normalizadas, $\boldsymbol{g}$ é um vetor amostrado da distribuição *Gumbel* e Δ é o parâmetro de temperatura. À medida que $\Delta \to 0$, $\tilde{\boldsymbol{y}}$ aproxima-se de uma distribuição categórica, enquanto $\tilde{\boldsymbol{y}}$ aproxima-se de uma distribuição uniforme à medida que $\Delta \to \infty$.

3.3 Redes Siamesas

As primeiras Redes Siamesas, do inglês *Siamese Neural Networks* (SNN), foram introduzidas por Bromley et al. (1993) para detectar conjuntos de assinaturas forjadas, embora também tenham sido apresentadas por Baldi e Chauvin (1993). A principal característica de uma Rede Siamesa é a utilização de duas arquiteturas "base" idênticas[10], as quais dividem seus parâmetros (pesos e vieses) e são treinadas conjuntamente de acordo com um objetivo de similaridade. A diferença entre as arquiteturas base reside em suas entradas de dados, onde cada sub-rede irá processar um dado específico e calcular a projeção deste dado em um espaço latente[11]. Por fim, a arquitetura siamesa utiliza uma medida de similaridade para definir se ambas projeções são diferentes ou similares. A Figura 10 ilustra um exemplo de arquitetura siamesa.

Figura 10 – Exemplo de arquitetura de uma Rede Siamesa.



Fonte: Elaborado pelo autor.

Seja S o conjunto de duas arquiteturas neurais base de uma Rede Siamesa, enquanto $f(\cdot)$ corresponda à saída desta respectiva rede. Adicionalmente, seja $\mathcal{X}$ o conjunto de dados a ser utilizado no treinamento de tal arquitetura. A Rede Siamesa calcula uma função de similaridade a partir de um par de dados $(\boldsymbol{x}, \hat{\boldsymbol{x}})$ e de acordo com a Equação 47:

$$\gamma = \frac{f(\boldsymbol{x}) \bullet f(\hat{\boldsymbol{x}})}{\sqrt{f(\boldsymbol{x}) \bullet f(\boldsymbol{x})}\sqrt{f(\hat{\boldsymbol{x}}) \bullet f(\hat{\boldsymbol{x}})}}, \tag{47}$$

dado que $\boldsymbol{x} \in \mathcal{X}$, $\hat{\boldsymbol{x}} \in \mathcal{X}$ e $\boldsymbol{x} \neq \hat{\boldsymbol{x}}$. Adicionalmente, γ é a similaridade de cosseno e $\bullet$ é o operador de produto escalar.

O modelo original utiliza a similaridade de cosseno para mapear a similaridade entre o par de dados $(\boldsymbol{x}, \hat{\boldsymbol{x}})$, a qual fornece valores no intervalo $[-1, 1]$. Portanto, parte do intervalo corresponde à um rótulo negativo, isto é, pares são dissimilares quando γ está entre -1 e 0, e a outra parte corresponde à um rótulo positivo, isto é, pares são similares quando γ está entre 0 e 1. O Algoritmo 6 descreve o pseudo-código de treinamento da Rede Siamesa.

[10] Quaisquer arquiteturas neurais podem ser utilizadas como bases da Rede Siamesa, tais como, RNNs e CNNs.

[11] Tal processo assemelha-se à uma extração de características.

ALGORITMO 6: Pseudo-código de treinamento da Rede Siamesa.

Entrada: Conjunto de treinamento $\mathcal{X} = \{(\boldsymbol{x}_1, \hat{\boldsymbol{x}}_1, y_1), (\boldsymbol{x}_2, \hat{\boldsymbol{x}}_2, y_2), \ldots, (\boldsymbol{x}_m, \hat{\boldsymbol{x}}_m, y_m)\}$, conjunto de arquiteturas siamesas S, número de épocas E e taxa de treinamento η.

Salida: Parâmetros otimizados da Rede Siamesa (θ_S^*).

1 $e \leftarrow 1$;
2 **mientras** $(e <= E)$ **hacer**
3 $\quad \boldsymbol{x} \leftarrow \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_m\}$;
4 $\quad \hat{\boldsymbol{x}} \leftarrow \{\hat{\boldsymbol{x}}_1, \hat{\boldsymbol{x}}_2, \ldots \hat{\boldsymbol{x}}_m\}$;
5 $\quad \boldsymbol{y} \leftarrow \{y_1, y_2, \ldots, y_m\}$;
6 $\quad f_1 \leftarrow S(\boldsymbol{x})$;
7 $\quad f_2 \leftarrow S(\hat{\boldsymbol{x}})$;
8 $\quad \gamma \leftarrow \frac{f_1 \bullet f_2}{\sqrt{f_1 \bullet f_1}\sqrt{f_2 \bullet f_2}}$;
9 $\quad S \leftarrow \nabla_{\theta_S}(\gamma - \boldsymbol{y})^2$;
10 $\quad \theta_S \leftarrow \theta_S - \eta \cdot \mathsf{Adam}(\theta_S, S)$;
11 $\quad e \leftarrow e + 1$;
12 $\theta_S^* \leftarrow \theta_S$;

A linha 1 define o início da época, enquanto as linhas 2-11 definem uma época de treinamento. O par de dados é selecionado pelas linhas 3 e 4, enquanto seu rótulo é seleciado pela linha 5. As saídas das redes são calculadas pelas linhas 6 e 7, a função de perda é calculada pela linha 8, os gradientes da rede são calculadoos pela linha 9 e os parâmetros são atualizados pela linha 10. Por fim, a linha 11 incrementa o contador de épocas, enquanto a linha 12 atribui os parâmetros pós-treinamento como os parâmetros otimizados da rede.

3.3.1 Funções de Perda por Similaridade

A tradicional Rede Siamesa utiliza a similaridade de cosseno para mapear a similaridade entre o par de dados e a função de perda de Erro Médio Quadrático, do inglês *Mean Squared Error* (MSE), para treinar a rede. Entretanto, Hadsell et al. (2006) constataram que a utilização de métricas de similaridade baseadas em distância, tais como distâncias de *Manhattan* e Euclidiana, consideram o peso e magnitude dos vetores de entrada e, consequentemente, são capazes de melhor representar a distância verdadeira entre os vetores. Por outro lado, a similaridade de cosseno calcula o ângulo entre os vetores de entrada e é usualmente utilizada quando a magnitude dos vetores não importa. Desta forma, três funções de perda baseadas em distância foram propostas com o intuito de aprimorar o treinamento destas redes.

Entropia Cruzada Binária

A Entropia Cruzada Binária, do inglês *Binary Cross-Entropy* (BCE), é tradicionalmente utilizada para o treinamento de arquiteturas neurais na tarefa de classificação binária. Dado que as Redes Siamesas realizam uma classificação binária, ou seja, classificam pares de dados

em similares e dissimilares, tal função de perda é uma escolha padrão. Usualmente, as funções de similaridade utilizadas em conjunto da Entropia Cruzada Binária são a concatenação dos vetores de saída (f_1 e f_2), bem como a diferença absoluta entre tais vetores. A Equação 48 descreve a sua formulação, como segue:

$$L_B = y \cdot log(\hat{y}) + (1 - y) \cdot (log(1 - \hat{y})), \tag{48}$$

onde y é o rótulo verdadeiro do par de dados e $\hat{y} = \sigma(|f_1 - f_2|)$, ou seja, a saída da função de ativação da Rede Siamesa[12].

Perda Contrastiva

Hadsell et al. (2006) propuseram a função de Perda Contrastiva, do inglês *Contrastive Loss* (CL), a qual almeja construir um espaço Euclidiano onde pares positivos (amostras similares) estão próximos entre si e distantes de pares negativos (amostras dissimilares). A vantagem desta função é utilizar um espaço de distância que melhor penaliza dados não similares entre si, bem como considerar dois distintos objetivos em uma mesma função, isto é, minimizar valores de representações similares e maximizar valores de representações dissimilares. Tal função é descrita pela Equação 49:

$$L_C = (1 - y) \cdot d^2 + y \cdot \max(0, m_g - d)^2, \tag{49}$$

onde y é o rótulo verdadeiro do par de dados, d é a distância Euclidiana entre f_1 e f_2, e m_g é a margem que define o raio de similaridade.

Perda Tripla

A função de Perda Tripla, do inglês *Triplet Loss* (TL) e proposta por Schroff et al. (2015), aprimora a Perda Contrastiva através da utilização de trios de dados, os quais são compostos por dados âncoras, dados positivos (similares aos dados âncoras) e dados negativos (dissimilares aos dados âncoras). O objetivo desta função é minimizar a distância entre dados positivos e dados âncoras e, simultaneamente, maximizar a distância entre dados negativos e dados âncoras. A Equação 50 descreve a função de Perda Tripla, como segue:

$$L_T = \max(0, m_g + d_p - d_n), \tag{50}$$

onde d_p é a distância Euclidiana entre a projeção da imagem âncora[13] e f_1, d_n é a distância Euclidiana entre a projeção da imagem âncora e f_2, e m_g é a margem que define o raio de similaridade.

[12] Tradicionalmente, tarefas de classificação binária utilizam a função sigmoide $\sigma(\cdot)$ como função de ativação.
[13] A projeção de uma imagem âncora $\tilde{\boldsymbol{x}}$ é dada por $S(\tilde{\boldsymbol{x}})$.

Adicionalmente, cada trio de dados recebe uma categorização de acordo com o seu posicionamento no espaço de projeções. Os trios fáceis contribuem com uma função de perda igual a zero e não são muito úteis para o aprendizado, pois estão posicionados de acordo com a seguinte fórmula: $d_p + m_g < d_n$. Por outro lado, os trios difíceis possuem os dados negativos mais próximos dos dados âncora, ou seja, $d_n < d_p$, enquanto os trios semi-difíceis possuem a parte negativa mais distante da âncora do que a parte positiva, isto é, $d_p < d_n < d_p + m_g$.

4 Otimização Meta-Heurística

A palavra matemática, do latim *mathematica*, refere-se ao conhecimento e é considerada uma base formal de representação para os fenômenos existentes. Diversas áreas do conhecimento utilizam-na para a resolução de seus problemas, tais como, Economia (DIXIT, 1990), Geofísica (WILKINSON et al., 2012), Engenharia Elétrica (CHENG et al., 2010), Engenharia Mecânica (RAO et al., 2011), Engenharia Civil (GOLDBERG; KUO, 1987), dentre outras. Entretanto, alguns problemas não são triviais e necessitam de um conjunto adicional de ferramentas para serem solucionados. Este conjunto de ferramentas, conhecido por otimização ou programação matemática, visa encontrar as informações (parâmetros) mais adequadas para solucioná-los.

Um exemplo clássico de otimização é o problema do Caixeiro Viajante (PAPADIMITRIOU, 1977), que consiste em visitar m localidades, passando uma única vez por cada uma delas e retornando à localidade inicial. O objetivo deste problema é encontrar a rota que possua o custo mínimo, ou seja, o melhor trajeto possível que percorra as m localidades. Analogamente ao problema mencionado, é possível deparar-se com outros problemas de otimização em nosso cotidiano; por exemplo, uma empresa de logística a qual necessita encontrar uma rota de custo mínimo para transportar sua carga. Ademais, diversas áreas do conhecimento beneficiam-se da otimização, tais como, modelagem de componentes industriais (RAO, 2009), pesquisa operacional (RARDIN, 1998), modelos econômicos de mercado (KONNO; YAMAZAKI, 1991), modelagem molecular (BARONE et al., 1998), dentre outros.

Um processo de otimização consiste em minimizar uma função matemática, denominada de função objetivo, a partir da escolha sistemática de um conjunto de valores permitidos. A Equação 51 descreve o modelo genérico de minimização[14] sem restrições.

$$\min_{\forall \boldsymbol{x} \in \Re^n} f(\boldsymbol{x}), \tag{51}$$

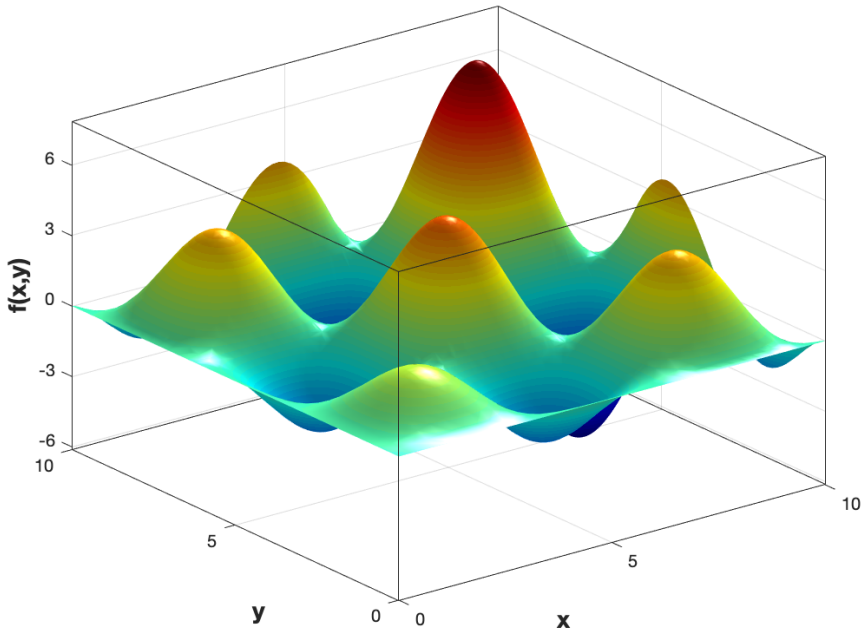
onde $f(\boldsymbol{x})$ é a função objetivo, dado que $\boldsymbol{x} \in \Re^n$. A otimização desta função objetivo visa encontrar o conjunto mais adequado de valores para $\boldsymbol{x}$, denominado por $\boldsymbol{x}^*$, onde $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$.

Contudo, à medida que as funções objetivo tornam-se mais complexas, é possível encontrar diversos pontos de mínimo, chamados de mínimos locais[15]. Portanto, a solução ótima do problema é dada pelo ponto de mínimo global, isto é, o menor ponto de mínimo local. A Figura 11 mostra a superfície de uma função objetivo que possui diversos pontos de mínimo e máximo locais.

[14] Lembre-se que $\max(f(x)) = \min(-(f(x))$.

[15] Note que também é possível encontrar pontos de máximo, denominados por máximos locais.

Figura 11 – Exemplo da função $f(x, y) = \text{sen}(x) \cdot \text{sqrt}(x) \cdot \text{sen}(y) \cdot \text{sqrt}(y)$ com diversos pontos de mínimo e máximo locais.



Fonte: Elaborado pelo autor.

Tradicionalmente, métodos iterativos de otimização são utilizados para a resolução de problemas de otimização, tais como, método de Newton, método Quasi-Newton, gradiente descendente, métodos de interpolação, dentre outros (BERTSEKAS, 1999). Estes métodos utilizam a avaliação de Hessianos, gradientes ou o próprio valor da função objetivo para encontrar as soluções ótimas, facilitando o processo de convergência embora aumentando a complexidade computacional do problema.

Recentemente, uma abordagem não-dependente da avaliação de derivadas, denominada por meta-heurística (YANG, 2011), vem sendo utilizada para a solução de problemas de otimização. A meta-heurística consiste em um procedimento de alto nível projetado para buscar, gerar ou selecionar uma heurística, provendo uma solução suficientemente boa para o problema de otimização. Em outras palavras, a meta-heurística é um processo generativo que combina conceitos de diversificação (*exploration*), o qual realiza buscas em grandes porções do espaço de busca, e intensificação (*exploitation*), o qual refina uma solução promissora a partir de uma busca em sua vizinhança.

As meta-heurísticas possuem uma complexidade menor quando comparadas aos métodos iterativos, pois são técnicas constituídas por simples buscas locais e procedimentos de aprendizado inspirados em comportamentos biológicos. Ademais, as meta-heurísticas não possuem domínios específicos, podendo ser aplicadas em quaisquer problemas, e possuem mecanismos que evitam o aprisionamento em regiões de mínimo ou máximo locais.

A literatura fornece um número vasto de algoritmos de otimização meta-heurística, os quais podem ser classificados em diferentes taxonomias (JR. et al., 2013), tais como, algoritmos de enxame, algoritmos bioinspirados, algoritmos evolucionistas e algoritmos baseados em conceitos físicos e químicos. Para a realização da presente tese, optamos por escolher quatro algoritmos estados da arte e baseados em enxame. O primeiro algoritmo, introduzido por Kennedy et al. (2001) é a tradicional Otimização por Enxame de Partículas, amplamente utilizada e tida como referência na área da computação bioinspirada. Os três algoritmos restantes, isto é, Algoritmo do Morcego (YANG, 2010b), Algoritmo do Vagalume (YANG, 2010a) e Busca do Cuckoo (YANG; DEB, 2010), foram introduzidos por Yang (2011) e demonstraram-se tão eficazes quanto à Otimização por Enxame de Partículas, atingindo resultados equiparáveis e até melhores em determinadas tarefas. Adicionalmente, também optamos por incluir um algoritmo evolucionista conhecido por Programação Genética, introduzido por Koza (1994), dado que o mesmo é capaz de melhor explorar o espaço de busca a partir da sintetização de expressões matemáticas, as quais são combinadas a partir de operadores básicos (soma, subtração, divisão, multiplicação, dentre outros). A Tabela 1 descreve as siglas e nomenclaturas de todos os algoritmos meta-heurísticos utilizados.

Sigla	**Nomenclatura (Português)**	**Nomenclatura (Inglês)**
BA	Algoritmo do Morcego	*Bat Algorithm*
CS	Busca do Cuckoo	*Cuckoo Search*
FA	Algoritmo do Vagalume	*Firefly Algorithm*
GP	Programação Genética	*Genetic Programming*
PSO	Otimização por Enxame de Partículas	*Particle Swarm Optimization*

Tabela 1 – Sigla e nomenclaturas das meta-heurísticas utilizadas.

4.1 Algoritmo do Morcego

Uma das características mais perceptíveis dos morcegos é a ecolocalização, um tipo de sonar utilizado a fim de detectar presas, localizar pontos de descanso e evitar obstáculos. Os morcegos emitem um pulso sonoro de alta frequência e "escutam" o eco que ressalta dos objetos ao seu redor. A frequência de cada pulso é constante[16], porém, sua duração é de apenas milissegundos (8 à 10ms). Adicionalmente, tal mecanismo de ecolocalização é capaz de detectar a distância, orientação e velocidade do alvo.

[16] O intervalo de variação da frequência é de 25kHz à 150kHz.

Matematicamente falando, a ecolocalização pode ser associada à uma função objetivo, possibilitando a formulação de um algoritmo de otimização. O Algoritmo do Morcego, do inglês *Bat Algorithm* (BA), utiliza três princípios básicos:

a) Todos os morcegos utilizam a ecolocalização como uma percepção de distância, diferenciando entre predadores, presas e obstáculos;

b) Morcegos voam randomicamente com velocidade $\boldsymbol{z}$ pelo espaço de busca, ocupando uma posição $\boldsymbol{w}$, possuindo uma frequência inicial fixa $f_{\min}$ e variando o ruído A em busca das melhores presas. Adicionalmente, os morcegos podem ajustar suas frequências de pulso f, bem como sua taxa de emissão de pulsos $r \in [0, 1]$, dependendo da proximidade do alvo;

c) O ruído é assumido como um valor fixo ou uma redução linear de um valor A para um valor mínimo $A_{\min}$.

Seja $\mathcal{B} = (b_1, b_2, \ldots, b_M)$ um bando de morcegos, tal que $b_i = (\boldsymbol{w}_i, \boldsymbol{z}_i)$, onde $\boldsymbol{w}_i \in \Re^N$ e $\boldsymbol{z}_i \in \Re^N$ são as posições e velocidades do morcego i, respectivamente. Para cada morcego i, onde $i = (1, 2, \ldots, M)$, são associados valores de frequência $f_i \in [f_{\min}, f_{\max}]$, ruído A_i e taxa de emissão de pulsos r_i. À cada iteração, ajusta-se o novo valor de frequência para cada morcego, bem como sua velocidade e posição, dadas pelas Equações 52, 53 e 54, respectivamente:

$$f_i^{t+1} = f_{\min} + (f_{\max} - f_{\min})\theta, \tag{52}$$

$$\boldsymbol{z}_i^{t+1} = \boldsymbol{z}_i^t + (\boldsymbol{w}_i^t - \hat{\boldsymbol{w}})f_i, \tag{53}$$

$$\boldsymbol{w}_i^{t+1} = \boldsymbol{w}_i^t + \boldsymbol{z}_i^{t+1} \tag{54}$$

onde θ é um vetor randômico obtido através de uma distribuição uniforme[17] e $\hat{\boldsymbol{w}}$ é a melhor posição global (solução), encontrada após a comparação entre todos os morcegos do bando.

Posteriormente, se a taxa de emissão de pulsos de um morcego i for menor que uma probabilidade p amostrada ($r_i < p$), uma nova solução é gerada em torno da melhor solução atual. A Equação 55 formula tal procedimento:

$$\boldsymbol{w}_i^{t+1} = \hat{\boldsymbol{w}} + \epsilon\bar{A}, \tag{55}$$

onde ϵ é um número randômico do intervalo $[-1, 1]$, e $\bar{A}$ é o ruído médio de todos os morcegos no bando. Note que o Algoritmo 7 é responsável por descrever o pseudo-código do funcionamento do Algoritmo do Morcego.

[17] Neste trabalho, $\mathcal{U}$ foi adotado como o símbolo da distribuição uniforme.

ALGORITMO 7: Pseudo-código do Algoritmo do Morcego.

Entrada: Número de morcegos M, número de variáveis de decisão N, número máximo de iterações T, frequência mínima $f_{\min}$, frequência máxima $f_{\max}$, ruído A e taxa de emissão de pulsos r.

Salida: Vetor $\boldsymbol{g}$ de variáveis de decisão contendo os valores mais plausíveis para a função objetivo.

```
Função objetivo f(w), w = (w_1, ..., w_N);
Inicializar bando de morcegos b_i | ∀i = (1, 2, ..., M), bem como suas posições w_i e
  velocidades z_i;
Definir a frequência f_i para cada b_i;
t ← 1;
mientras (t <= T) hacer
    para i de 1 até M hacer
        Gerar nova solução w_i a partir do ajuste de frequência e atualização de
          velocidade e posição, de acordo com as Equações 52, 53 e 54;
        p ← U(0, 1);
        si (r_i < p) entonces
            Selecionar a solução global ŵ;
            Gerar uma solução local w_i ao redor da solução global utilizando a
              Equação 55;
        si (p < A & f(w_i) < f(ŵ)) entonces
            Aceitar nova solução w_i;
        Ordenar os morcegos e encontrar a solução global atual ŵ;
    t ← t + 1;
g ← ŵ;
```

4.2 Algoritmo do Vagalume

Os vagalumes produzem luzes cintilantes através de um processo bioluminescente, o qual é frequentemente único para cada espécie. Tais sinais luminosos são responsáveis por atrair parceiros (comunicação), atrair possíveis presas e servir como mecanismo de alerta (YANG, 2010a).

Matematicamente falando, podemos associar os sinais luminosos à uma função objetivo, formulando um algoritmo de otimização. A intensidade de luz I decresce à medida que uma distância d aumenta, sendo $I \propto 1/d^2$. Aliado à absorção de luz pelo ar, os vagalumes podem ser vistos em distâncias limitadas, permitindo sua comunicação. Adicionalmente, o Algoritmo do Vagalume, do inglês *Firefly Algorithm* (FA), utiliza três princípios idealizados:

a) Todos os vagalumes são unissexo, sendo atraídos a qualquer outro vagalume independente do sexo;

b) Atratividade é proporcional à luminosidade, decrescendo à medida que a distância aumenta. Se não existe nenhum vagalume mais brilhante que outro, ele mover-se-á aleatoriamente;

c) A luminosidade de cada vagalume é afetada ou determinada pela superfície da função objetivo. Para um problema de minimização, a luminosidade é inversamente proporcional ao valor da função objetivo.

Seja $\mathcal{V} = (v_1, v_2, \ldots, v_M)$ um enxame de vagalumes, tal que $v_i = \boldsymbol{x}_i$, onde $\boldsymbol{x}_i \in \Re^N$ é a posição do vagalume i, onde $i = (1, 2, \ldots, M)$. Adicionalmente, a luminosidade de cada vagalume i é determinada pelo valor da função objetivo em sua posição, isto é, $f(\boldsymbol{x}_i)$. A atratividade de um vagalume $\boldsymbol{x}_i$ em relação à outro vagalume $\boldsymbol{x}_j$, $\beta(\boldsymbol{x}_i, \boldsymbol{x}_j)$, varia conforme uma distância d_{ij} (distância Euclidiana) entre o vagalume i e o vagalume j, sendo formulada pela Equação 56:

$$\beta(\boldsymbol{x}_i, \boldsymbol{x}_j) = \beta_0 e^{-\gamma d_{ij}^2}, \tag{56}$$

onde β_0 é a atratividade inicial e γ é o coeficiente de absorção de luz.

Cada vagalume na posição $\boldsymbol{x}_i$ e iteração t é atraído em direção à outro vagalume mais brilhante na posição $\boldsymbol{x}_j$ pela Equação 57 se, e somente se $f(\boldsymbol{x}_j) > f(\boldsymbol{x}_i)$:

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \beta(\boldsymbol{x}_i^t, \boldsymbol{x}_j^t)(\boldsymbol{x}_j^t - \boldsymbol{x}_i^t) + \alpha \left(\sigma \frac{1}{2}\right), \tag{57}$$

onde α é um fator de aleatoriedade que controla a magnitude da perturbação estocástica de $\boldsymbol{x}_i^t$ e $\sigma \in \mathcal{U}(0, 1)$. Após todos os vagalumes na iteração t terem sido atualizados, o melhor vagalume $\hat{\boldsymbol{x}}$ caminhará aleatoriamente sobre o espaço de busca, descrito pela Equação 58:

$$\hat{\boldsymbol{x}} = \hat{\boldsymbol{x}} + \alpha \left(\sigma \frac{1}{2}\right), \tag{58}$$

onde $\sigma \in \mathcal{U}(0, 1)$. O Algoritmo 8 é responsável por descrever o pseudo-código do funcionamento do Algoritmo do Vagalume.

4.3 Busca do Cuckoo

Os cuckoos são pássaros estonteantes devido aos sons que produzem e, principalmente, devido à sua estratégia agressiva de reprodução (YANG; DEB, 2010), a qual é realizada através do depósito de ovos em ninhos de outras aves (PAYNE, 2005). Ao utilizar uma reprodução parasita, é possível gerar um extenso número de filhotes sem preocupações próprias, dado que os filhotes de cuckoo nascem mais rápidos que os de outras espécies e eliminam seus pseudo-irmãos, tomando conta do ninho hospedeiro.

Estudos recentes demonstraram que a busca de comida pelos cuckoos ocorre de uma maneira quase randômica, assemelhando-se às características dos vôos de Lévy, do inglês *Lévy Flights* (BROWN et al., 2007; PAVLYUKEVICH, 2007). A Busca do Cuckoo, do inglês *Cuckoo Search* (CS), obedece à três regras idealizadas:

ALGORITMO 8: Pseudo-código do Algoritmo do Vagalume.

Entrada: Número de vagalumes M, número de variáveis de decisão N, número máximo de iterações T, coeficiente de absorção de luz γ, atratividade inicial β_0 e fator de aleatoriedade α.

Salida: Vetor $\boldsymbol{g}$ de variáveis de decisão contendo os valores mais plausíveis para a função objetivo.

```
Função objetivo f(x), x = (x_1, ..., x_N);
Inicializar a população de vagalumes v_i | ∀i = (1, 2, ..., M);
mientras (t <= T) hacer
    para i de 1 até M hacer
        para j de 1 até M hacer
            d_ij ← distância euclidiana entre vagalumes i e j;
            si (f(x_i) > f(x_j)) entonces
                Gerar atratividade β de acordo com a Equação 56;
                Mover o vagalume i em direção ao vagalume j de acordo com a
                 Equação 57;
    Avaliar enxame em busca do melhor vagalume;
    x̂ ← Melhor vagalume;
    Melhor vagalume x̂ anda aleatoriamente sobre o espaço a partir da Equação 58;
    t ← t + 1;
g ← x̂;
```

a) Cada cuckoo deposita um ovo por vez em um ninho aleatório;

b) Os melhores ninhos, os quais possuem os melhores ovos (soluções), são carregados para as próximas gerações;

c) O número de ninhos hospedeiros é fixo, sendo que o hospedeiro pode descobrir o ovo intruso com probabilidade $p_a \in [0, 1]$. Neste caso, o pássaro hospedeiro pode jogar fora o ovo ou abandonar o ninho, construindo um novo ninho em uma nova localização.

Seja $\mathcal{C} = (c_1, c_2, \ldots, c_M)$ um conjunto de ninhos hospedeiros, tal que $c_i = \boldsymbol{s}_i$, onde $\boldsymbol{s}_i \in \Re^N$ é a posição do cuckoo i e $i = (1, 2, \ldots, M)$. À cada iteração, uma nova solução $\boldsymbol{s}_i$ é gerada para um cuckoo i por meio de um vôo de Lévy, dada pela Equação 59:

$$\boldsymbol{s}_i^{t+1} = \boldsymbol{s}_i^t + \alpha * \text{Levy}(\lambda), \tag{59}$$

onde $\alpha > 0$ é o tamanho do passo e $*$ é o operador de produto interno. Os vôos de Lévy efetuam uma busca randômica com valores obtidos através de uma distribuição de Lévy, dada pela Equação 60:

$$\text{Levy} \backsim u = t^{-\lambda}, (1 < \lambda \leq 3), \tag{60}$$

a qual possui média e variância infinitas.

Caso esta nova solução seja melhor que a anterior, ela é substituída. Note que podemos trocar a probabilidade p_a por uma fração dos M ninhos a serem substituídos, abandonando-os e gerando novos ninhos em novas localizações por meio dos vôos de Lévy. O Algoritmo 9 é responsável por descrever o pseudo-código da Busca do Cuckoo.

ALGORITMO 9: Pseudo-código da Busca do Cuckoo.

Entrada: Número de ninhos hospedeiros M, número de variáveis de decisão N, número máximo de iterações T, probabilidade de substituição dos piores ninhos p_a, tamanho do passo α e variável β para computar a distribuição de Lévy.

Salida: Vetor $\boldsymbol{g}$ de variáveis de decisão contendo os valores mais plausíveis para a função objetivo.

```
Função objetivo f(s), s = (s_1, ..., s_N);
Inicializar a população de ninhos hospedeiros s_i | ∀i = (1, 2, ..., M);
t ← 1;
mientras (t <= T) hacer
    Escolher um ninho i aleatório;
    Gerar uma nova solução s_i a partir de um vôo randômico de Lévy usando a
     Equação 59;
    Escolher um ninho j aleatório;
    si (f(s_i) < f(s_j)) entonces
        Substituir o ninho j pela nova solução obtida anteriormente;
    Ordenar os ninhos e encontrar o atual melhor ŝ;
    Abandonar uma fração p_a dos piores ninhos e substituir suas soluções por novas
     posições via vôos de Lévy, utilizando a Equação 60;
    Ordenar novamente os ninhos e encontrar o atual melhor ŝ;
    t ← t + 1;
g ← ŝ;
```

4.4 Otimização por Enxame de Partículas

A Otimização por Enxame de Partículas, do inglês *Particle Swarm Optimization* (PSO), é um algoritmo de inteligência coletiva inspirado pela dinâmica do comportamento social (KENNEDY et al., 2001). A ideia é modelar a aprendizagem do comportamento social em um algoritmo de otimização, o qual permite que cada partícula (solução) voe dentro de uma espaço de busca (enxame) à procura de outras partículas com melhores características. Este processo simula a interação social entre um bando de aves à procura de comida ou até mesmo entre seres humanos tentando alcançar um objetivo em comum. Este mecanismo sócio-recognitivo pode ser resumido em três grandes princípios (KENNEDY et al., 2001): (i) avaliação, (ii) comparação e (iii) imitação, onde cada partícula avalia outras dentro de sua vizinhança através da função objetivo, comparando-as com seu próprio valor e, finalmente, decidindo se é uma boa escolha imitá-las ou não.

Seja $\mathcal{D} = (d_1, d_2, \ldots, d_M)$ um conjunto de partículas que compõem o enxame, $d_i = (\boldsymbol{\psi}_i, \boldsymbol{\rho}_i)$, onde $\boldsymbol{\psi}_i \in \Re^N$ e $\boldsymbol{\rho}_i \in \Re^N$ são as posições e velocidades da partícula i, respectivamente. Além disso, para cada partícula i, onde $i = (1, 2, \ldots, M)$, também é conhecida sua melhor solução (posição no enxame) local $\hat{\boldsymbol{\psi}}_i$, bem como a melhor solução (global) $\boldsymbol{g}$ de todo o enxame. Cada partícula é inicializada com valores aleatórios de velocidade e posição. Posteriormente, cada indivíduo é avaliado de acordo com uma função objetivo, atualizando o seu mínimo local. No final de cada iteração, o valor mínimo global é atualizado com a partícula que obteve a melhor posição no enxame. Este processo é repetido até que algum critério de convergência seja satisfeito. As Equações 61 e 62 são responsáveis por modelar a atualização da velocidade e posição da partícula i na iteração t, respectivamente:

$$\boldsymbol{\rho}_i^{t+1} = \mu \boldsymbol{\rho}_i^t + c_1 r_1 (\hat{\boldsymbol{\psi}}_i - \boldsymbol{\psi}_i^t) + c_2 r_2 (\boldsymbol{g} - \boldsymbol{\psi}_i^t) \tag{61}$$

e

$$\boldsymbol{\psi}_i^{t+1} = \boldsymbol{\psi}_i^t + \boldsymbol{\rho}_i^{t+1}, \tag{62}$$

onde μ é a força de inércia que controla o poder de interação entre as partículas, e $r_1, r_2 \in [0, 1]$ são variáveis aleatórias que dão traços de estocasticidade ao PSO. As constantes social c_1 e cognitiva c_2 são utilizadas para guiar as partículas dentro do espaço de busca. O Algoritmo 10 descreve o pseudo-código do PSO.

ALGORITMO 10: Pseudo-código da Otimização por Enxame de Partículas.

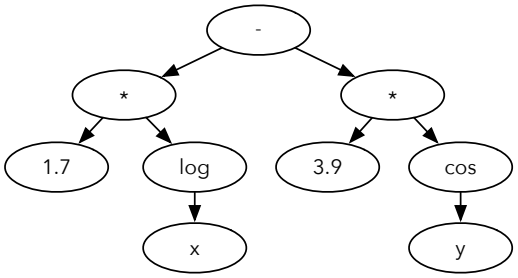
Entrada: Número de partículas M, número de variáveis de decisão N, número máximo de iterações T, constantes c_1 e c_2, e peso de inércia μ.
Saída: Vetor $\boldsymbol{g}$ de variáveis de decisão contendo os valores mais plausíveis para a função objetivo.

```
Função objetivo f(ψ), ψ = (ψ_1, ..., ψ_N);
Inicializar a população de partículas d_i | ∀i = (1, 2, ..., M);
t ← 1;
mientras (t <= T) hacer
    para i de 1 até M hacer
        Atualizar a velocidade da partícula i de acordo com a Equação 61;
        Atualizar a posição da partícula i de acordo com a Equação 62;
    Avaliar enxame em busca da melhor partícula;
    ψ̂ ← Melhor partícula;
    t ← t + 1;
g ← ψ̂;
```

4.5 Programação Genética

A Programação Genética, do inglês *Genetic Programming* (GP) (KOZA, 1994), é um algoritmo evolucionista de otimização, o qual utiliza princípios da Teoria da Evolução de *Darwin* e operadores bioinspirados para criar soluções promissoras e atingir um objetivo. Ao passo que o GP assemelha-se ao tradicional Algoritmo Genético, do inglês *Genetic Algorithm* (GA), existem diferenças fundamentais. Uma solução típica do GP utiliza uma estrutura de árvore composta por nós terminais e nós de função, a qual é ilustrada pela Figura 12. Necessariamente, os nós terminais representam constantes ou variáveis de decisão, enquanto os nós de função são operadores matemáticos aplicados aos nós terminais para realizar a avaliação das árvores. Durante o processo de evolução, diversos procedimentos são realizados sobre a população com o intuito de produzir indivíduos mais aptos, tais como, seleção, reprodução, mutação e *crossover*.

Figura 12 – Exemplo de uma solução do GP representando a expressão $1,7\log(x) - 3,9\cos(y)$.



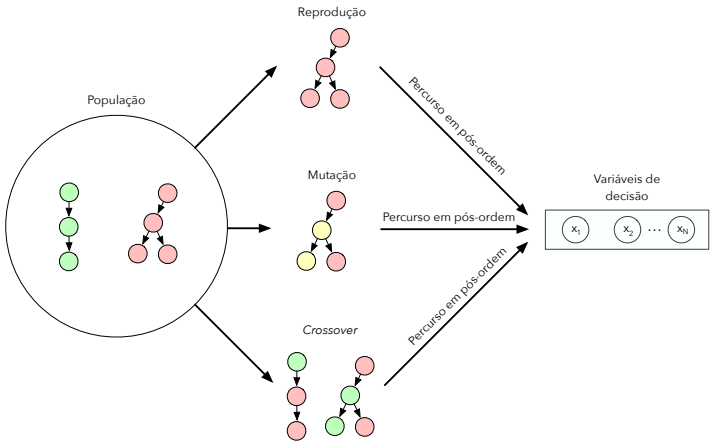
Fonte: Elaborado pelo autor.

Durante o processo de convergência (iterações ou gerações), as operações mencionadas acima são empregues na população. Inicialmente, os melhores indivíduos da geração atual são selecionados e reproduzidos, gerando um indivíduo possivelmente mais apto e garantindo melhores indivíduos nas futuras gerações. Posteriormente, operações de mutação e *crossover* são aplicadas para fornecer um fator de variabilidade na população. Em outras palavras, a mutação modifica aleatoriamente um alelo (nó ou galho) de uma árvore em específico, enquanto o *crossover* troca nós ou galhos entre duas árvores. Finalmente, após satisfazer o critério de parada, o melhor indivíduo (solução) é "colhido" da árvore e utilizado como solução final.

Seja $\mathcal{T} = (t_1, t_2, \ldots, d_M)$ um conjunto de árvores que compõem a população, $t_i = h(e_i)$, onde $h : E \rightarrow \Re^N$ é a função que avalia a expressão matemática e_i no espaço das expressões E e retorna um vetor de N posições contendo o valor das variáveis de decisão do indivíduo i, onde $i = (1, 2, \ldots, M)$. Adicionalmente, e_i é uma expressão matemática obtida após percorrer a árvore i em um percurso de pós-ordem. Durante o processo de evolução, até que um critério de parada seja satisfeito, a população é avaliada e ordenada de forma ascendente, ou seja, os melhores indivíduos são aqueles que possuem o menor valor de aptidão, isto é, avaliação da função objetivo. O primeiro operador seleciona indivíduos da população atual com probabilidade

p_r e reproduz-os (mantém-os) para a próxima geração. Posteriormente, com probabilidade p_m, as árvores são mutacionadas e, com probabilidade p_c, as árvores sofrem uma operação de *crossover*. Ao término do uso dos operadores genéticos, a população é novamente avaliada e ordenada. A Figura 13 ilustra o fluxograma da evolução de uma população da Programação Genética, enquanto o Algoritmo 11 descreve o pseudo-código da Programação Genética.

Figura 13 – Fluxograma da evolução de uma população da Programação Genética.



Fonte: Elaborado pelo autor.

ALGORITMO 11: Pseudo-código da Programação Genética.

Entrada: Número de árvores M, número de variáveis de decisão N, número máximo de iterações T, probabilidade de reprodução p_r, probabilidade de mutação p_m e probabilidade de *crossover* p_c.

Salida: Vetor $\boldsymbol{g}$ de variáveis de decisão contendo os valores mais plausíveis para a função objetivo.

Função avaliação de expressões $h : E \rightarrow \Re^N$;
Função objetivo $f(h(e))$, $h(e) = (x_1, \ldots, x_N)$;
Inicializar população de árvores $t_i \mid \forall i = (1, 2, \ldots, M)$;
$t \leftarrow 1$;
$M_r = Mp_r$;
$M_m = Mp_m$;
$M_c = \frac{Mp_c}{2}$;
mientras $(t <= T)$ **hacer**
    Selecionar M_r árvores dentre a população e armazená-las para a próxima geração;
    Selecionar M_m árvores e mutacionar seus galhos;
    Selecionar M_c pares de árvores e trocar os seus galhos;
    para i *de* 1 *até* M **hacer**
        Avaliar a expressão matemática e_i gerada pela árvore i;
    Ordenar as árvores de forma ascendente em busca da expressão da melhor árvore;
    $\hat{e} \leftarrow$ Expressão da melhor árvore;
    $t \leftarrow t + 1$;
$\boldsymbol{g} \leftarrow h(\hat{e})$;

5 Trabalhos Relacionados

Inicialmente, uma busca extensiva com duas palavras-chaves, e.g., *generative adversarial network* e *text generation*, foi conduzida em bases de dados renomadas, tais como *Science Direct*, *IEEEXplore*, *Springer*, *Association for Computing Machinery* (ACM) e *arXiv*, para encontrar os trabalhos mais relevantes dentro do contexto proposto por esta tese. O resultado da busca (mais de 100 ocorrências) necessitou ser afunilado para trabalhos que tenham sido citados ao menos uma vez e que tenham de fato empregue GANs para geração de texto. Com a restrição da busca, constatou-se que existem 35 trabalhos datados entre os anos 2016 e 2020, os quais estão melhores descritos pela coletânea proposta por Rosa e Papa (2021).

Fora do período analisado pela coletânea (a partir de 2021), Yuan et al. (2021) propuseram as Redes Adversariais Generativas Esparsas, as quais codificam representações esparsas a partir da combinação linear de alguns *tokens* da matriz de vetorização, aproximando o espaço de características entre o gerador e o discriminador. Wu et al. (2021) propuseram a utilização do aprendizado por imitação para as Redes Adversariais Generativas, onde uma política de otimização baseada em proximidade em conjunto de um discriminador contrastivo provem recompensas mais plausíveis e estabilizam a variância dos gradientes. Por fim, Yan et al. (2021) propuseram as Redes Adversariais Generativas com discriminadores condicionais capazes de fornecer recompensas baseadas em sequências ao invés de *tokens* individuais. Adicionalmente, os autores também introduzem geradores amplificados por discriminadores, projetados para prevenir o pré-treino.

A grande quantidade de modelos adversariais textuais propostos torna inviável a aplicação e avaliação comparativa de todos os mesmos. Portanto, a partir de uma nova seleção, foi possível constatar que existem 4 modelos principais[18] chamados de “estados da arte", isto é, modelos que propuseram inovações significativas e atingiram resultados que servem como referências de comparação. Yu et al. (2016) introduziram o primeiro modelo generativo para geração de sequências utilizando dados discretos, denotado por Rede Adversarial Generativa Sequencial. Esta arquitetura modela a parte generativa como uma diretriz estocástica baseada em Aprendizado por Reforço, contornando o problema da diferenciação da função objetivo com uma atualização automática de pesos. Paralelamente, Kusner e Hernández-Lobato (2016) propuseram a Rede Adversarial Generativa *Gumbel-Softmax*, a qual utiliza uma distribuição de *Gumbel-Softmax* para tornar possível a diferenciação da função objetivo.

Posteriormente, Che et al. (2017) introduziram a Rede Adversarial Generativa Ampliada por Máxima-Verosimilhança, a qual utiliza uma nova função objetivo de treinamento baseada em máxima-verosimilhança para auxiliar a estabilidade do mecanismo de recompensa do

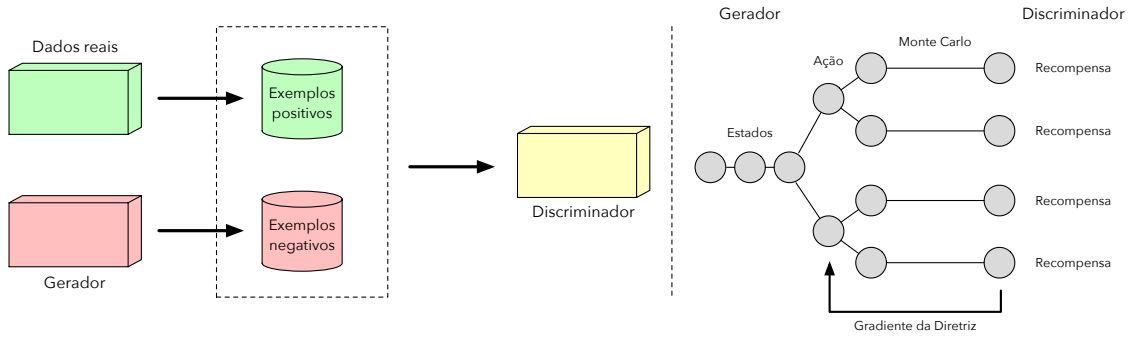
[18] Estes modelos correspondem aos trabalhos com maiores números de citações e comparações em trabalhos correlatos.

discriminador. Finalmente, Nie et al. (2019) introduziram um modelo generativo adversarial denominado Rede Adversarial Generativa Relacional, o qual é composto por três principais componentes: parte generativa baseada em memória relacional, função objetivo *Gumbel-Softmax* e múltiplas características provenientes de modelos de vetorização, os quais auxiliam na atualização dos gradientes da parte generativa.

5.1 Rede Adversarial Generativa Sequencial

Yu et al. (2016) introduziram as Redes Adversariais Generativas Sequenciais, do inglês *Sequence Generative Adversarial Networks* (SeqGAN), para a geração de sequências (geração de dados discretos). Os autores consideraram a geração de sequências como um processo de tomada de decisões sequenciais, onde o modelo generativo é estruturado como um agente de Aprendizado por Reforço e atualizado através de um método de gradiente de diretriz. O algoritmo escolhido para tal propósito é o REINFORCE em conjunto das árvores de *Monte Carlo* (MC) (SILVER et al., 2016), pois a utilização de ambos provê um sinal de recompensa à cada etapa de geração (tempo t). Ademais, o sinal de recompensa é fornecido pelo discriminador, o qual auxilia a GAN a gerar informações que assemelham-se ao conjunto real de dados. Esta arquitetura é ilustrada pela Figura 14.

Figura 14 – Arquitetura de uma Rede Adversarial Generativa Sequencial.



Fonte: Elaborado pelo autor.

O problema de geração de sequências é representado pela geração de uma sequência $\boldsymbol{y} = (y_1, y_2, \ldots, y_T) \mid \boldsymbol{y} \in \mathcal{Y}$, onde $\mathcal{Y}$ é o vocabulário de *tokens* candidatos e T é o número de tokens a serem gerados. Este problema é modelado utilizando o Aprendizado por Reforço, onde o modelo generativo G parametrizado por θ_G gera um estado s à cada tempo t, o qual corresponde à um *token* y_t gerado. Ademais, sua próxima ação, denotada por a, é o próximo *token* y_{t+1} a ser escolhido. Portanto, a diretriz $G(y_t \mid \boldsymbol{y}_{1:t-1}; \theta_G)$ é estocástica, tal que o estado de transição é determinístico depois da escolha de uma ação, isto é, $\delta^a_{s,\tilde{s}} = 1$ para o próximo estado $\tilde{s} = \boldsymbol{y}_{1:t}$, se o estado atual $s = \boldsymbol{y}_{1:t-1}$ e a ação $a = y_t$.

Adicionalmente, o modelo discriminativo D parametrizado por θ_D guia o gerador através de uma probabilidade $D(\boldsymbol{y})$ que indica se a sequência $\boldsymbol{y}$ pertence aos dados reais ou não. Este modelo discriminativo é treinado a partir da utilização de exemplos positivos (dados reais) e exemplos negativos (dados artificiais), fornecendo um sinal de recompensa para a atualização do modelo generativo. A Equação 63 descreve o processo de obtenção do sinal de recompensa:

$$Q_D^G(a;s) = \begin{cases} \frac{1}{N}\sum_{n=1}^{N} D(\boldsymbol{y}^n) \mid \boldsymbol{y}^n \in MC(\boldsymbol{y}_{1:t};N) & \text{para } t < T \\ D(\boldsymbol{y}) & \text{para } t = T \end{cases}, \tag{63}$$

onde $Q_D^G(a;s)$ é a função de ação-valor de uma sequência, isto é, o sinal de recompensa acumulado do estado s dado uma ação a e N o número de amostragens via *Monte Carlo*. O Algoritmo 12 descreve o pseudo-código do treinamento da Rede Adversarial Generativa Sequencial.

ALGORITMO 12: Pseudo-código de treinamento da Rede Adversarial Generativa Sequencial.

Entrada: Conjunto de treinamento $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$, discriminador D, gerador G, número de épocas E_A, tamanho máximo da sequência T, número de épocas do gerador E_{A_G}, número de épocas do discriminador E_{A_D}, tamanho do lote l, taxa de treinamento do gerador η_G e taxa de treinamento do discriminador η_D.

Salida: Parâmetros otimizados do discriminador (θ_D^*) e gerador (θ_G^*).

1 Pré-treinar gerador G por Estimação da Verosimilhança Máxima em exemplos de $\mathcal{X}$;
2 $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\} \in \tilde{\mathcal{X}} \leftarrow G(\cdot) \cup \mathcal{X}$;
3 Pré-treinar discriminador D por Entropia Cruzada Binária em exemplos de $\tilde{\mathcal{X}}$;
4 $e \leftarrow 1$;
5 **mientras** $(e <= E_A)$ **hacer**
6 **para** i *de* 1 *até* E_{A_G} **hacer**
7 Gerar sequência $\boldsymbol{y} = (y_1, y_2, \ldots, y_T)$ a partir de $G(\cdot)$;
8 **para** t *de* 1 *até* T **hacer**
9 $R \leftarrow Q(y_t; \boldsymbol{y}_{1:T-1})$;
10 $\theta_G \leftarrow \theta_G + \eta_G \cdot \nabla_{\theta_R} R$;
11 **para** i *de* 1 *até* E_{A_D} **hacer**
12 $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\} \in \tilde{\mathcal{X}} \leftarrow G(\cdot) \cup \mathcal{X}$;
13 $D \leftarrow \frac{1}{l}\sum_{j=1}^{l}[\log D(\boldsymbol{x}_j) + \log(1 - D(G(\cdot)))]$;
14 $\theta_D \leftarrow \theta_D - \eta_D \cdot \nabla_{\theta_D} D$;
15 $e \leftarrow e + 1$;
16 $\theta_D^* \leftarrow \theta_D$;
17 $\theta_G^* \leftarrow \theta_G$;

A linha 2 corresponde à união do conjunto de amostras artificias e reais, o qual é utilizado para o pré-treinamento do discriminador. A linha 4 inicializa o contador de épocas, enquanto as linhas 5-15 definem uma época de treinamento. A linha 7 gera uma amostra

falsa, a qual é utilizada no cálculo dos gradientes (linha 9) e atualização dos parâmetros do gerador (linha 10). A linha 12 une novamente amostras reais e artificiais, as quais são utilizadas para gerar os gradientes do discriminador (linha 13) e atualizar os seus parâmetros (linha 14). Por fim, a linha 15 incrementa o contador de épocas, enquanto as linhas 16 e 17 atribuem os parâmetros pós-treinamento como os parâmetros otimizados do discriminador e gerador, respectivamente.

Yu et al. avaliaram o modelo proposto em duas partes experimentais: sintética e real. Considerando a parte sintética, os autores propuseram uma base de dados sintética denotada por Oráculo, a qual é composta por 10.000 sequências de tamanho máximo de 20 caracteres e amostradas por uma LSTM aleatoriamente inicializada e pré-treinada via Estimação da Verosimilhança Máxima, do inglês *Maximum Likelihood Estimation* (MLE). Posteriormente, os autores utilizaram estes mesmos dados e compararam um conjunto de arquiteturas[19] através da métrica NLL. Os resultados experimentais comprovaram a eficácia da SeqGAN, a qual obteve um desempenho de $2,4\%$ melhor que a Política de Gradiente com BLEU e $15,3\%$ melhor que a LSTM com inicialização aleatória.

O segundo experimento foi avaliado em três bases de dados públicas: Poemas Chineses, Discursos Políticos do Obama e Músicas de Nottingham. Adicionalmente, os autores preferiram utilizar a métrica BLEU ao invés do NLL para avaliar as arquiteturas nestas bases, dado que os dados são separados entre conjuntos de treinamento e teste. Nesta parte experimental, a SeqGAN foi comparada apenas com uma LSTM pré-treinada via MLE, embora obteve um aumento de desempenho considerando as três bases de dados: $10,7\%$ nos Poemas Chineses, $2,64\%$ nos Discursos Políticos do Obama e $2,13\%$ nas Músicas de Nottingham.

Finalmente, os autores atestaram que a SeqGAN estabeleceu um marco para os modelos adversariais textuais, dado que foi o primeiro modelo capaz de modelar e solucionar o problema da diferenciabilidade em GANs via Aprendizado por Reforço. Embora tenha atingido resultados satisfatórios e melhores do que uma LSTM pré-treinada, os autores não puderam comparar com modelos adversariais textuais mais recentes e nem com um padrão de bases de dados estabelecidos pela comunidade. Tal fato dificulta a real comparação do quão robusta a SeqGAN é perante às outras arquiteturas correlatas.

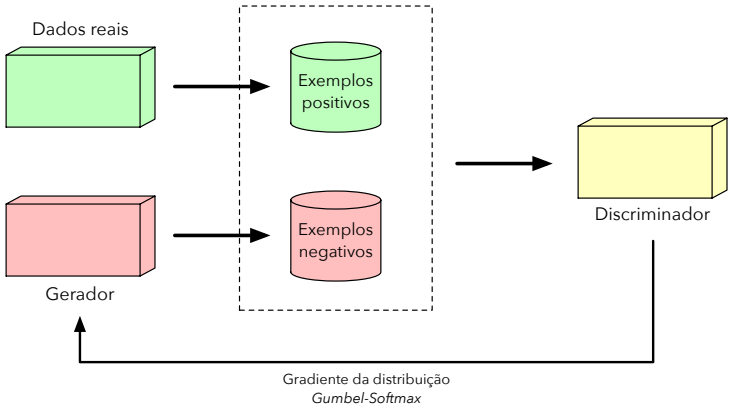
5.2 Rede Adversarial Generativa *Gumbel-Softmax*

As Redes Adversariais Generativas *Gumbel-Softmax*, do inglês *Gumbel-Softmax Generative Adversarial Network* (GSGAN), foram propostas por Kusner e Hernández-Lobato (2016) e solucionaram o problema da diferenciabilidade do gerador através da utilização de uma distribuição *Gumbel-Softmax*. Essencialmente, a distribuição *Gumbel-Softmax* é contínua

[19] As arquiteturas selecionadas foram SeqGAN, LSTM com inicialização aleatória, LSTM pré-treinada via MLE, Amostragem Calculada e Política de Gradiente com BLEU.

e diferenciável, permitindo que os gradientes do gerador sejam propagados e a mesma seja treinada. Ademais, a arquitetura deste modelo remete à tradicional Rede Adversarial Generativa, apenas com uma modificação na função de perda, a qual é obtida através da divergência de *Kullback-Leibler*. Esta arquitetura é ilustrada pela Figura 15 e o Algoritmo 13 descreve o pseudo-código do treinamento da Rede Adversarial Generativa *Gumbel-Softmax*.

Figura 15 – Arquitetura de uma Rede Adversarial Generativa *Gumbel-Softmax*.



Fonte: Elaborado pelo autor.

ALGORITMO 13: Pseudo-código de treinamento da Rede Adversarial Generativa *Gumbel-Softmax*.

Entrada: Conjunto de treinamento $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$, discriminador D, gerador G, número de épocas E_A, tamanho do lote l, taxa de treinamento do discriminador η_D e taxa de treinamento do gerador η_G.

Salida: Parâmetros otimizados do discriminador (θ_D^*) e gerador (θ_G^*).

1 Pré-treinar gerador G por Estimação da Verosimilhança Máxima em exemplos de $\mathcal{X}$;
2 $e \leftarrow 1$;
3 **mientras** $(e <= E_A)$ **hacer**
4 Amostrar $\{G(\boldsymbol{z}_1), G(\boldsymbol{z}_2), \ldots, G(\boldsymbol{z}_l\})$ da distribuição artificial;
5 Amostrar $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_l\}$ da distribuição real;
6 $D \leftarrow \nabla_{\theta_D} \frac{1}{l} \sum_{j=1}^{l} [-\log D(\boldsymbol{x}_j) - \log(1 - D(G(\boldsymbol{z}_j)))]$;
7 $\theta_D \leftarrow \theta_D - \eta_D \cdot \text{Adam}(\theta_D, D)$;
8 $G \leftarrow \nabla_{\theta_G} \frac{1}{l} \sum_{j=1}^{l} [\log \frac{D(G(\boldsymbol{z}_j))}{1 - D(G(\boldsymbol{z}_j))}]$;
9 $\theta_G \leftarrow \theta_G - \eta_G \cdot \text{Adam}(\theta_G, G)$;
10 $e \leftarrow e + 1$;
11 $\theta_D^* \leftarrow \theta_D$;
12 $\theta_G^* \leftarrow \theta_G$;

A linha 2 inicializa o contador de épocas, enquanto as linhas 3-10 definem uma época de treinamento. A linha 6 utiliza uma amostra artificial (linha 4) e uma amostra real (linha 5) para calcular os gradientes (linha 6) e atualizar os parâmetros do discriminador (linha 7). A linha 8 calcula os gradientes do gerador e linha 9 atualiza os seus parâmetros. Por fim, a

linha 10 incrementa o contador de épocas, enquanto as linhas 11 e 12 atribuem os parâmetros pós-treinamento como os parâmetros otimizados do discriminador e gerador, respectivamente.

Kusner e Hernández-Lobato utilizaram uma gramática livre de contexto (GLC) com 5.000 amostras de no máximo 12 caracteres como base de dados para a parte experimental. Adicionalmente, os autores forneceram informações específicas para o treinamento, isto é, ambos discriminador e gerador foram treinados por 20.000 épocas com um otimizador *Adam* (KINGMA; BA, 2017), taxa de treinamento igual a $0,001$ e lotes de 200 amostras. Os autores também optaram por um decaimento do parâmetro de temperatura da distribuição *Gumbel-Softmax*, o qual decai linearmente de 5 a 1. Finalmente, os resultados experimentais foram avaliados através de uma comparação visual entre os dados gerados pela GSGAN e por uma LSTM pré-treinada via MLE.

Os autores constataram que a GSGAN foi a primeira arquitetura a utilizar a distribuição *Gumbel-Softmax* para solucionar o problema da diferenciação de uma distribuição discreta, porém, a partir de uma análise crítica dos experimentos conduzidos, é possível perceber que ela não foi devidamente avaliada em bases de dados da tarefa de geração de texto e nem comparada às outras arquiteturas adversariais textuais. Adicionalmente, a presente tese acredita que um único artifício matemático (distribuição *Gumbel-Softmax*) não é capaz de solucionar o problema da estabilidade do treinamento e, se devidamente comparado, deverá atingir resultados inferiores ao Aprendizado por Reforço. Por outro lado, a tese acredita que o caminho mais efetivo é realizar mudanças na arquitetura do discriminador, o qual é ativamente responsável por influenciar os valores de gradientes utilizados no gerador.

5.3 Rede Adversarial Generativa Ampliada Por Máxima-Verosimilhança

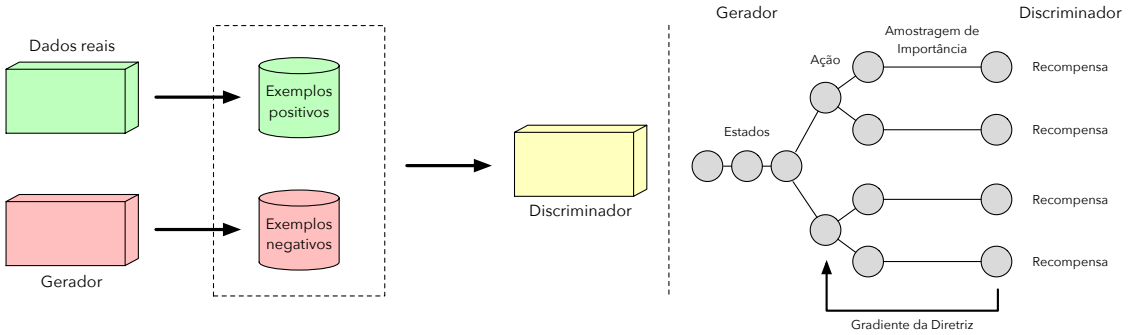
As Redes Adversariais Generativas Ampliadas por Máxima-Verosimilhança, do inglês *Maximum-Likelihood Augmented Generative Adversarial Network* (MaliGAN), foram propostas por Che et al. (2017) e introduzem uma nova função objetivo de treinamento baseada em máxima-verosimilhança, a qual auxilia na estabilidade do mecanismo de recompensa do discriminador. Tradicionalmente, modelos baseados em Aprendizado por Reforço sofrem com estabilidade nas épocas iniciais, dado que o gerador ainda não é capaz de gerar sequências boas e, consequentemente, não consegue receber sinais de recompensa adequados. Embora um pré-treinamento dedicado e mecanismos de redução de variância tenham sido utilizados por Yu et al. (2016), o algoritmo de RL ainda é bastante instável e não escalável para bases de dados grandes.

A Rede Adversarial Generativa Ampliada Por Máxima-Verosimilhança, ilustrada pela Figura 16 e descrita pelo Algoritmo 14, utiliza um objetivo baseado em máximo-verosimilhança e informações do discriminador como fontes adicionais de sinais de recompensa. Este objetivo é otimizado através de técnicas de amostragem de importância e redução de variância, permitindo

ao modelo não somente otimizar a máxima-verosimilhança de comportamentos bons, mas também minimizar a verosimilhança de comportamentos ruins. Em outras palavras, o modelo é capaz de focar no aprendizado de informações importantes e ignorar comportamentos não relevantes, auxiliando na estabilidade do treinamento. A Equação 64 descreve o sinal de recompensa proposto:

$$R = \frac{D(\boldsymbol{x})}{1 - D(\boldsymbol{x})}. \tag{64}$$

Figura 16 – Arquitetura de uma Rede Adversarial Generativa Ampliada Por Máxima-Verosimilhança.



Fonte: Elaborado pelo autor.

A linha 2 corresponde à união do conjunto de amostras artificias e reais, o qual é utilizado para o pré-treinamento do discriminador (linha 3). A linha 4 inicializa o contador de épocas, enquanto as linhas 5-12 definem uma época de treinamento. A linha 7 gera uma amostra falsa, a qual é utilizada no cálculo dos gradientes (linha 8) e atualização dos parâmetros do discriminador (linha 9). A linha 10 calcula os gradientes do gerador e a linha 11 atualiza os seus parâmetros. Por fim, a linha 12 incrementa o contador de épocas, enquanto as linhas 13 e 14 atribuem os parâmetros pós-treinamento como os parâmetros otimizados do discriminador e gerador, respectivamente.

A parte experimental da MaliGAN foi realizada em duas tarefas distintas: geração de imagens e geração de texto. Considerando a tarefa de geração de texto, os autores utilizaram as bases de dados de Poemas Chineses e Penn Treebank. A primeira base de dados foi particionada em dois subconjuntos, contendo um número fixo de caracteres (5 e 7), e avaliado através das métricas BLEU e PPL. A MaliGAN obteve taxas melhores de BLEU e PPL perante às arquiteturas comparadas, isto é, atingiu desempenho superior a 10% quando comparada à LSTM pré-treinada e $3,2\%$ em relação à SeqGAN. A base de dados Penn Treebank é focada em geração de texto baseada em sentenças, onde apenas sentenças com menos de 35 palavras foram consideradas pelo experimento em questão. Neste experimento, a MaliGAN foi comparada apenas com uma LSTM pré-treinada via MLE e obteve um desempenho de $10,4\%$ a mais na métrica PPL.

ALGORITMO 14: Pseudo-código de treinamento da Rede Adversarial Generativa Ampliada Por Máxima-Verosimilhança.

Entrada: Conjunto de treinamento $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$, discriminador D, gerador G, número de épocas E_A, número de épocas do discriminador E_{A_D}, tamanho do lote l, taxa de treinamento do discriminador η_D e taxa de treinamento do gerador η_G.

Salida: Parâmetros otimizados do discriminador (θ_D^*) e gerador (θ_G^*).

Pré-treinar gerador G por Estimação da Verosimilhança Máxima em exemplos de $\mathcal{X}$;
$\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\} \in \tilde{\mathcal{X}} \leftarrow G(\cdot) \cup \mathcal{X}$;
Pré-treinar discriminador D por Entropia Cruzada Binária em exemplos de $\tilde{\mathcal{X}}$;
$e \leftarrow 1$;
mientras $(e <= E_A)$ **hacer**
    para i *de* 1 *até* E_{A_D} **hacer**
        $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\} \in \tilde{\mathcal{X}} \leftarrow G(\cdot) \cup \mathcal{X}$;
        $D \leftarrow \frac{1}{l} \sum_{j=1}^{l} [\log D(\boldsymbol{x}_j) + \log(1 - D(G(\cdot)))]$;
        $\theta_D \leftarrow \theta_D - \eta_D \cdot \nabla_{\theta_D} D$;
    $R \leftarrow \frac{1}{l} \sum_{j=1}^{l} [\frac{D(\boldsymbol{x}_j)}{1 - D(\boldsymbol{x}_j)}]$;
    $\theta_G \leftarrow \theta_G + \eta_G \cdot \nabla_{\theta_R} R$;
    $e \leftarrow e + 1$;
$\theta_D^* \leftarrow \theta_D$;
$\theta_G^* \leftarrow \theta_G$;

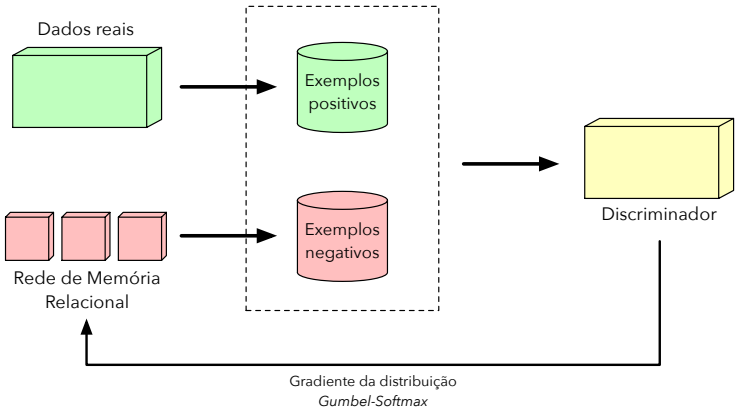
O ponto positivo da arquitetura da MaliGAN é a proposição de uma nova função objetiva baseada na divergência de *Kullback-Leibler*, a qual não necessita da distribuição *Gumbel-Softmax* nem do Aprendizado por Reforço. A ideia é tentar mitigar os problemas de instabilidade causado por estas abordagens citadas, bem como aprimorar as atualizações de gradiente com uma função objetivo que visa calcular a diferença entre duas distribuições de probabilidade. No entanto, a função objetiva proposta não é totalmente independente do Aprendizado por Reforço e utiliza uma estratégia baseada em *Monte Carlo* para calcular os sinais de recompensa.

5.4 Rede Adversarial Generativa Relacional

Nie et al. (2019) propuseram as Redes Adversariais Generativas Relacionais, do inglês *Relational Generative Adversarial Network* (RelGAN), as quais utilizam um gerador composto por uma Rede de Memória Relacional, bem como uma diferenciação por *Gumbel-Softmax* e múltiplas representações vetoriais. Tais técnicas são capazes de melhor discriminar os dados e prover sinais mais informativos para a atualização de parâmetros. O processo de aprendizado é o mesmo proposto pela Rede Adversarial Generativa *Gumbel-Softmax*, exceto por duas pequenas diferenças: (i) função de perda relativística (JOLICOEUR-MARTINEAU, 2018) e (ii) decréscimo exponencial do parâmetro de temperatura da distribuição *Gumbel-Softmax*.

Esta arquitetura é ilustrada pela Figura 17 e o Algoritmo 15 descreve o pseudo-código do treinamento da Rede Adversarial Generativa Relacional.

Figura 17 – Arquitetura de uma Rede Adversarial Generativa Relacional.



Fonte: Elaborado pelo autor.

ALGORITMO 15: Pseudo-código de treinamento da Rede Adversarial Generativa Relacional.

Entrada: Conjunto de treinamento $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$, discriminador D, gerador G, número de épocas E_A, tamanho do lote l, taxa de treinamento do discriminador η_D, taxa de treinamento do gerador η_G, temperatura máxima $\Delta_{\max}$ e função sigmoide σ.

Salida: Parâmetros otimizados do discriminador (θ_D^*) e gerador (θ_G^*).

1 Pré-treinar gerador G por Estimação da Verosimilhança Máxima em exemplos de $\mathcal{X}$;
2 $e \leftarrow 1$;
3 **mientras** $(e <= E_A)$ **hacer**
4 Amostrar $\{G(\boldsymbol{z}_1), G(\boldsymbol{z}_2), \ldots, G(\boldsymbol{z}_l\})$ da distribuição artificial;
5 Amostrar $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_l\}$ da distribuição real;
6 $D \leftarrow \nabla_{\theta_D} \frac{1}{l} \sum_{j=1}^{l} [\log \sigma(D(\boldsymbol{x}_j) - D(G(\boldsymbol{z}_j)))]$;
7 $\theta_D \leftarrow \theta_D - \eta_D \cdot \text{Adam}(\theta_D, D)$;
8 $G \leftarrow -D$;
9 $\theta_G \leftarrow \theta_G - \eta_G \cdot \text{Adam}(\theta_G, G)$;
10 $\Delta \leftarrow \Delta_{\max}^{e/E}$;
11 $e \leftarrow e + 1$;
12 $\theta_D^* \leftarrow \theta_D$;
13 $\theta_G^* \leftarrow \theta_G$;

A linha 2 inicializa o contador de épocas, enquanto as linhas 3-11 definem uma época de treinamento. A linha 6 utiliza uma amostra artificial (linha 4) e uma amostra real (linha 5) para calcular os gradientes (linha 6) e atualizar os parâmetros do discriminador (linha 7). A linha 8 atribui o oposto dos gradientes do discriminador e os utiliza para atualizar os parâmetros do gerador (linha 9), enquanto a linha 10 atualiza o parâmetro de temperatura. Por fim, a

linha 11 incrementa o contador de épocas, enquanto as linhas 12 e 13 atribuem os parâmetros pós-treinamento como os parâmetros otimizados do discriminador e gerador, respectivamente.

Nie et al. propuseram a parte experimental em duas categorias de dados: sintéticos e reais. Os dados sintéticos são amostrados pelo mesmo Oráculo proposto pela SeqGAN, enquanto os dados reais advêm das bases de dados *COCO Image Captions* e *EMNLP2017 WMT News*. Considerando os dados sintéticos, os autores utilizaram 10.000 amostras com tamanhos máximos de sequência iguais a 20 e 40, e constataram que a RelGAN atinge desempenho superior de NLL quando comparada com outras arquiteturas, tais como SeqGAN ($23,5\%$ superior), RankGAN ($19,0\%$ superior), LeakGAN ($5,1\%$ superior) e LSTM pré-treinada via MLE ($26,1\%$ superior).

Em relação aos experimentos reais, os autores utilizaram conjuntos de treinamento e teste da base de dados *COCO Image Captions* com 10.000 amostras cada. Nesta base, os autores constataram que a RelGAN conseguiu atingir melhores taxas de BLEU quando comparada com todas as outras arquiteturas avaliadas (em média $14,7\%$ superior). Considerando a base de dados *EMNLP2017 WMT News*, os autores utilizaram um conjunto de treinamento de 270.000 sequências e um conjunto de teste com 10.000 sequências. Também foi possível constatar que a RelGAN atingiu desempenho superior considerando a métrica BLEU (em média $14,4\%$).

O desempenho superior da RelGAN está diretamente relacionado à sua arquitetura recorrente com memória relacional, a qual é mais poderosa do que as tradicionais LSTM e GRU. A RMC utiliza mecanismos de auto-atenção e uma memória relacional para melhor captar dependências temporais e, consequentemente, aprimorar a geração de texto. Não obstante, o custo de treinamento desta arquitetura é elevado, dado que a recorrência é obtida através da utilização de várias camadas de auto-atenção. Por outro lado, outro ponto não tão positivo da RelGAN é a utilização da distribuição *Gumbel-Softmax* ao invés do Aprendizado por Reforço, podendo inibir o real potencial de tal arquitetura dado que os gradientes serão atualizados apenas com respostas do discriminador e não com mecanismos de recompensa[20].

[20] Os mecanismos de recompensa geralmente usam abordagens baseadas em *Monte Carlo* para explorar várias possibilidades de dados gerados e, consequentemente, podem fornecer recompensas mais robustas.

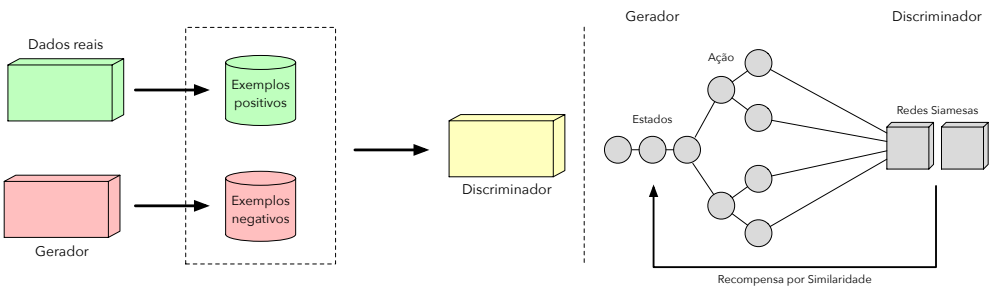
6 Rede Adversarial Generativa por Similaridade Textual

A instabilidade do treinamento de modelos adversariais é bastante comum em tarefas de geração de texto, dado que o discriminador precisa diferenciar textos reais de artificiais. Usualmente, esta diferenciação é realizada a partir da extração de características do texto, seja por uma arquitetura recorrente ou convolucional, seguida de um mapeamento linear para um rótulo positivo ou negativo.

Não obstante, tal estratégia é extremamente dependente de um pré-treinamento adequado do discriminador, o qual não é realizado em todos os modelos adversariais textuais relacionados. Outro problema decorrente desta estratégia é a utilização de apenas uma informação textual no discriminador, isto é, o discriminador decide a veracidade de apenas um texto por vez. Logo, é possível perceber que o discriminador está sempre dependente de um único contexto e não se beneficia de possíveis informações de similaridade que possam existir entre os textos discriminados.

Desta forma, a presente tese propõe um modelo adversarial textual baseado em similaridade com o objetivo de melhorar a discriminação entre dados artificiais e reais, bem como atacar o problema da instabilidade de convergência do treinamento. A Rede Adversarial Generativa por Similaridade Textual, do inglês *Text-Similarity Generative Adversarial Network* (TS-GAN) é estruturada a partir de um discriminador com funções de perda baseadas em similaridade, tais como Perda Contrastiva e Perda Tripla. A Figura 18 ilustra a arquitetura proposta para a TS-GAN.

Figura 18 – Arquitetura de uma Rede Adversarial Generativa por Similaridade Textual.



Fonte: Elaborado pelo autor.

Adicionalmente, a TS-GAN utiliza o Aprendizado por Reforço para solucionar o problema da diferenciabilidade de informações textuais. Portanto, o discriminador é responsável por analisar a similaridade de pares de textos e transformá-la em sinais de recompensa utilizados na estimação dos gradientes de atualização.

6.1 Discriminador por Similaridade

Os modelos adversariais textuais utilizam discriminadores baseados em Redes Neurais Convolucionais e Redes Neurais Recorrentes focados em extrações de características textuais. As Redes Neurais Convolucionais utilizam camadas de convolução e amostragem (*pooling*) para criar vetores de características, enquanto as Redes Neurais Recorrentes analisam recorrentemente etapas individuais da sequência e constroem o vetor de características (KIM, 2014). Contudo, tais abordagens avaliam exclusivamente uma sequência por vez, ou seja, apenas uma informação textual (real ou artificial) é passada pelo discriminador e transformada em recompensa. Alternativamente, a presente tese utiliza um discriminador composto por Redes Siamesas, as quais são capazes de processar pares ou trios de sequências, bem como projetá-las em espaços latentes e classificá-las de acordo com uma função de perda baseada em similaridade. Desta forma, a Rede Siamesa é treinada com o objetivo de minimizar a diferença entre conjuntos similares e maximizar a diferença entre conjuntos dissimilares.

Seja D o discriminador da TS-GAN composto por uma Rede Siamesa[21], enquanto seja $\tilde{\mathcal{X}}$ o conjunto de dados composto por pares ou trios de dados. Note que $\tilde{\mathcal{X}}$ é composto através da criação de pares ou trios de amostras aleatórias do conjunto real de dados e sequências amostradas pelo gerador. O primeiro passo é pré-treinar a Rede Siamesa de acordo com uma função de perda baseada em similaridade no conjunto de dados $\tilde{\mathcal{X}}$. Posteriormente, durante cada época E_A do aprendizado adversarial, D é treinado por E_{A_D} iterações em conjuntos $\tilde{\mathcal{X}}$ reamostrados[22]. Finalmente, D gera um mecanismo de recompensa de acordo com as Equações 65 e 66 para as funções de Perda Contrastiva e Perda Tripla, respectivamente:

$$R = \frac{D(\boldsymbol{x}, \hat{\boldsymbol{x}})}{1 - D(\boldsymbol{x}, \hat{\boldsymbol{x}})}, \tag{65}$$

$$R = \frac{D(\boldsymbol{x}, \hat{\boldsymbol{x}}, \tilde{\boldsymbol{x}})}{1 - D(\boldsymbol{x}, \hat{\boldsymbol{x}}, \tilde{\boldsymbol{x}})}, \tag{66}$$

onde $\boldsymbol{x}$ é um dado real, $\hat{\boldsymbol{x}}$ e $\tilde{\boldsymbol{x}}$ são dados amostrados pelo gerador, $(\boldsymbol{x}, \hat{\boldsymbol{x}})$ constitui um par de dados e $(\boldsymbol{x}, \hat{\boldsymbol{x}}, \tilde{\boldsymbol{x}})$ constitui um trio de dados.

6.2 Estratégia de Geração

A TS-GAN utiliza o Aprendizado por Reforço para solucionar o problema da diferenciabilidade de informações textuais e, portanto, torna-se indiferente à arquitetura do gerador. Desta forma, a presente tese utiliza geradores baseados em LSTMs, embora também propõem um experimento adicional que compara diferentes arquiteturas de geradores.

[21] Esta tese utiliza LSTMs como arquiteturas base, porém, quaisquer modelos podem ser utilizados.

[22] A reamostragem permite com que, ao passar das épocas, os exemplos amostrados pelo gerador sejam melhores e, consequentemente, diminuam a recompensa do discriminador.

6.3 Algoritmo de Treinamento

O algoritmo de treinamento da TS-GAN assemelha-se ao da MaliGAN, exceto por três diferenças. A TS-GAN utiliza um mecanismo de recompensa próprio (Equações 65 e 66), necessita de um pré-treinamento do discriminador através de uma função de perda baseada em similaridade e requer uma quantidade diferente de dados de entrada para o discriminador, isto é, pares de dados para a Perda Contrastiva e trios de dados para a Perda Tripla. O Algoritmo 16 descreve o pseudo-código de treinamento da TS-GAN com função de Perda Contrastiva (TS-GAN-C), enquanto o Algoritmo 17 descreve o pseudo-código de treinamento da TS-GAN com função de Perda Tripla (TS-GAN-T). Ambas arquiteturas utilizam geradores e discriminadores baseados em LSTMs e distância Euclidiana para o cálculo da similaridade.

ALGORITMO 16: Pseudo-código do treinamento da Rede Adversarial Generativa por Similaridade Textual com Perda Contrastiva.

Entrada: Conjunto de treinamento $\mathcal{X} = \{(\boldsymbol{x}_1, \hat{\boldsymbol{x}}_1), (\boldsymbol{x}_2, \hat{\boldsymbol{x}}_2), \ldots, (\boldsymbol{x}_m, \hat{\boldsymbol{x}}_m)\}$, Perda Contrastiva L_C, discriminador D, gerador G, número de épocas E_A, número de épocas do discriminador E_{A_D}, tamanho do lote l, taxa de treinamento do discriminador η_D e taxa de treinamento do gerador η_G.

Salida: Parâmetros otimizados do discriminador (θ_D^*) e gerador (θ_G^*).

1 Pré-treinar gerador G por Estimação da Verosimilhança Máxima em exemplos de $\mathcal{X}$;
2 $\{(\boldsymbol{x}_1, \hat{\boldsymbol{x}}_1), (\boldsymbol{x}_2, \hat{\boldsymbol{x}}_2), \ldots, (\boldsymbol{x}_m, \hat{\boldsymbol{x}}_m)\} \in \tilde{\mathcal{X}} \leftarrow G(\cdot) \cup \mathcal{X}$;
3 Pré-treinar discriminador D via Perda Contrastiva em exemplos de $\tilde{\mathcal{X}}$;
4 $e \leftarrow 1$;
5 **mientras** $(e <= E_A)$ **hacer**
6 **para** i *de* 1 *até* E_{A_D} **hacer**
7 $\{(\boldsymbol{x}_1, \hat{\boldsymbol{x}}_1), (\boldsymbol{x}_2, \hat{\boldsymbol{x}}_2), \ldots, (\boldsymbol{x}_m, \hat{\boldsymbol{x}}_m)\} \in \tilde{\mathcal{X}} \leftarrow G(\cdot) \cup \mathcal{X}$;
8 $D \leftarrow \frac{1}{l} \sum_{j=1}^{l} [L_C(\boldsymbol{x}_j, \hat{\boldsymbol{x}}_j)]$;
9 $\theta_D \leftarrow \theta_D - \eta_D \cdot \nabla_{\theta_D} D$;
10 $R \leftarrow \frac{1}{l} \sum_{j=1}^{l} [\frac{D(\boldsymbol{x}_j, \hat{\boldsymbol{x}}_j)}{1 - D(\boldsymbol{x}_j, \hat{\boldsymbol{x}}_j)}]$;
11 $\theta_G \leftarrow \theta_G + \eta_G \cdot \nabla_{\theta_R} R$;
12 $e \leftarrow e + 1$;
13 $\theta_D^* \leftarrow \theta_D$;
14 $\theta_G^* \leftarrow \theta_G$;

A linha 2 corresponde à união do conjunto de pares/trios de amostras artificias e reais, o qual é utilizado para o pré-treinamento do discriminador (linha 3). A linha 4 inicializa o contador de épocas, enquanto as linhas 5-12 definem uma época de treinamento. A linha 7 gera um par/trio de amostras falsa, o qual é utilizado no cálculo dos gradientes (linha 8) e atualização dos parâmetros do discriminador (linha 9). A linha 10 calcula os gradientes do gerador e a linha 11 atualiza os seus parâmetros. Por fim, a linha 12 incrementa o contador de épocas, enquanto as linhas 13 e 14 atribuem os parâmetros pós-treinamento como os parâmetros otimizados do discriminador e gerador, respectivamente.

ALGORITMO 17: Pseudo-código do treinamento da Rede Adversarial Generativa por Similaridade Textual com Perda Tripla.

Entrada: Conjunto de treinamento $\mathcal{X} = \{(\boldsymbol{x}_1, \hat{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_1), (\boldsymbol{x}_2, \hat{\boldsymbol{x}}_2, \tilde{\boldsymbol{x}}_2), \ldots, (\boldsymbol{x}_m, \hat{\boldsymbol{x}}_m, \tilde{\boldsymbol{x}}_m)\}$, Perda Tripla L_T, discriminador D, gerador G, número de épocas E_A, número de épocas do discriminador E_{A_D}, tamanho do lote l, taxa de treinamento do discriminador η_D e taxa de treinamento do gerador η_G.

Salida: Parâmetros otimizados do discriminador (θ_D^*) e gerador (θ_G^*).

Pré-treinar gerador G por Estimação da Verosimilhança Máxima em exemplos de $\mathcal{X}$;
$\{(\boldsymbol{x}_1, \hat{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_1), (\boldsymbol{x}_2, \hat{\boldsymbol{x}}_2, \tilde{\boldsymbol{x}}_2), \ldots, (\boldsymbol{x}_m, \hat{\boldsymbol{x}}_m, \tilde{\boldsymbol{x}}_m)\} \in \tilde{\mathcal{X}} \leftarrow G(\cdot) \cup \mathcal{X}$;
Pré-treinar discriminador D via Perda Tripla em exemplos de $\tilde{\mathcal{X}}$;
$e \leftarrow 1$;
mientras $(e <= E_A)$ **hacer**
    para i *de* 1 *até* E_{A_D} **hacer**
        $\{(\boldsymbol{x}_1, \hat{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_1), (\boldsymbol{x}_2, \hat{\boldsymbol{x}}_2, \tilde{\boldsymbol{x}}_2), \ldots, (\boldsymbol{x}_m, \hat{\boldsymbol{x}}_m, \tilde{\boldsymbol{x}}_m)\} \in \tilde{\mathcal{X}} \leftarrow G(\cdot) \cup \mathcal{X}$;
        $D \leftarrow \frac{1}{l} \sum_{j=1}^{l} [L_T(\boldsymbol{x}_j, \hat{\boldsymbol{x}}_j, \tilde{\boldsymbol{x}}_j)]$;
        $\theta_D \leftarrow \theta_D - \eta_D \cdot \nabla_{\theta_D} D$;
    $R \leftarrow \frac{1}{l} \sum_{j=1}^{l} [\frac{D(\boldsymbol{x}_j, \hat{\boldsymbol{x}}_j, \tilde{\boldsymbol{x}}_j)}{1 - D(\boldsymbol{x}_j, \hat{\boldsymbol{x}}_j, \tilde{\boldsymbol{x}}_j)}]$;
    $\theta_G \leftarrow \theta_G + \eta_G \cdot \nabla_{\theta_R} R$;
    $e \leftarrow e + 1$;
$\theta_D^* \leftarrow \theta_D$;
$\theta_G^* \leftarrow \theta_G$;

7 Metodologia

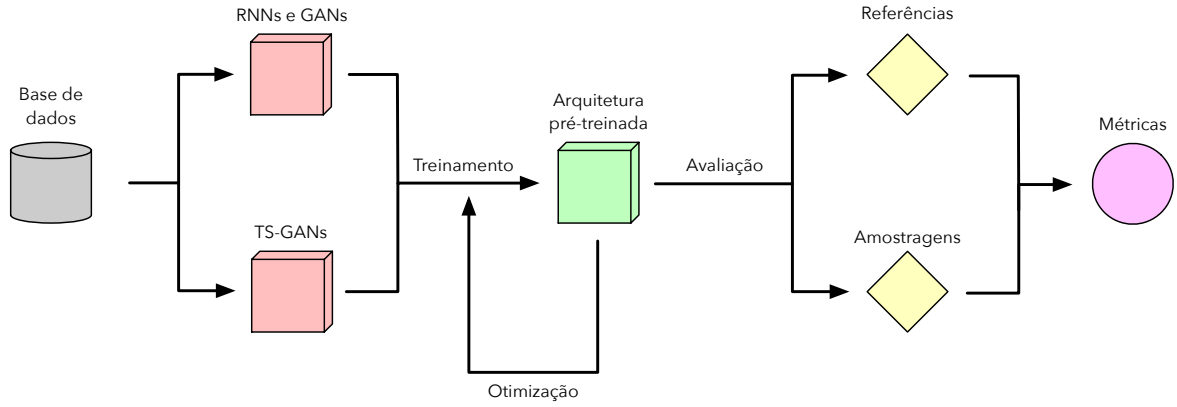
Conforme descrito no Capítulo 1, a presente tese objetiva aprimorar a estabilidade do treinamento de modelos adversariais textuais e contribuir cientificamente com a comunidade a partir de duas frentes de pesquisa inéditas:

a) Frente I: proposição de uma arquitetura denotada por Rede Adversarial Generativa por Similaridade Textual, a qual possui um discriminador baseado em funções de perda de similaridade aprendidas por Redes Siamesas;

b) Frente II: otimização meta-heurística dos hiperparâmetros da Rede Adversarial Generativa por Similaridade Textual.

O primeiro passo para conduzir a metodologia é a avaliação dos modelos propostos na Frente I nas mesmas bases de dados utilizadas pelas arquiteturas estado da arte, a fim de estabelecer uma análise comparativa coerente. Posteriormente, na Frente II, os modelos propostos são otimizados e novamente avaliados nestas mesmas bases de dados. A Figura 19 ilustra o fluxograma da metodologia em questão, enquanto as próximas seções descrevem com maiores detalhes as bases de dados, modelagens das frentes e configurações experimentais.

Figura 19 – Fluxograma da metodologia proposta.



Fonte: Elaborado pelo autor.

7.1 Bases de Dados

Esta seção descreve as bases de dados utilizadas para a validação dos experimentos propostos por esta tese.

Avaliação de Consumidores da Amazon

A base de dados Avaliação de Consumidores da Amazon, do inglês *Amazon Customer Reviews*[23] (NI et al., 2019) e conhecida por *Product Reviews*, é composta por mais de $233.000.000$ avaliações de usuários que expressaram suas opiniões e descreveram suas experiências em produtos comprados no site da Amazon. Adicionalmente, a base de dados fornece um conjunto de meta-dados, tais como nota de avaliação, identificadores de produtos e usuários, informações relacionadas aos produtos comprados, data de compra, dentre outros. A presente tese está interessada na criação de um modelo generativo de linguagem e portanto, utiliza apenas a parte textual da base de dados (avaliações). Entretanto, devido à uma grande quantidade de avaliações, é utilizado um conjunto de dados com 20.000 amostras da categoria de produtos *Movies and TV*, vocabulário de 3.989 *tokens*, tamanho máximo de sequência igual a 25 e tamanho do lote igual a 64. Finalmente, a Tabela 2 descreve alguns exemplos da base de dados em questão.

Avaliação
"one of my favorite shows to watch . i could watch this again and again . i will miss this series when it ends ."
"i loved this movie when i was a kid and still love it 20 years later ."
"it 's a must see for any fans of the book series as well as mark harmon fans !"
"this was an upgrade to hd dvd . it was what i requested and came in a timely manner ."

Tabela 2 – Exemplos de avaliações da base de dados Avaliação de Consumidores da Amazon.

[23] <https://nijianmo.github.io/amazon/index.html>

Avaliações do Yelp

A base de dados Avaliações do Yelp, do inglês *Yelp Reviews*[24] (ASGHAR, 2016), é composta por $8.635.403$ avaliações no idioma inglês de estabelecimentos de 8 áreas metropolitanas dos Estados Unidos e Canadá. Ademais, nesta tese, são utilizadas apenas 5.000 avaliações aleatórias com vocabulário de 12.635 *tokens*, tamanho máximo de sequência igual a 40 e tamanho do lote igual a 64. A Tabela 3 apresenta alguns exemplos da base de dados em questão.

Avaliações
"5 stars for the location, it's absolutely wonderful. I ordered the shamrock bubble tea, which was one of the best I've ever had. I am subtracting a star for service however. It was the barista that was so put out by waiting on us and then the 10-15 minutes it took to make the bubble tea... That's a bit long for a drink."
"A cafe sister restaurant to the ever popular Chambar Restaurant, the owners are so smart. This place had a line waiting in front just for a table when we got there at noon, but luckily we only waited for about 10 minutes to get seated. I soon saw why as there's really only a few tables in the long narrow space.."
"This is only a review for the auto shop, not the gas. This is the only place I've been for an inspection sticker. They seem to know what they are doing as I have never had a problem with any inspection sticker. They are honest and all business. I even once drove from Boston because I know I can trust them."

Tabela 3 – Exemplos de avaliações da base de dados Avaliações do Yelp.

Legendas de Imagens COCO

A Legendas de Imagens COCO, do inglês *COCO Image Captions*[25] (CHEN et al., 2015), foi desenvolvida através da coleta de legendas para imagens pertencentes à base de dados *Microsoft Common Objects in Context* (MS COCO). Em 2014, humanos criaram aproximadamente $1.026.459$ legendas a partir dos mesmos conjuntos de dados utilizados pela MS COCO e constantemente vem adicionando novas anotações. Entretanto, a presente tese utiliza um sub-conjunto de apenas 1.000 legendas, com vocabulário de 1.347 *tokens*, tamanho máximo de sequência igual a 10 e tamanho do lote igual a 32, pois pretende validar a abordagem proposta em bases de diferentes tamanhos. Finalmente, a Tabela 4 detalha alguns exemplos da base de dados em questão.

[24] <https://www.yelp.com/dataset>
[25] <https://cocodataset.org>

Legendas
"a large yellow bus parked in some grass"
"a school bus sitting in some overgrown grass"
"three traffic signs sit on on a corner in front of a brick wall"
"a garage door leading out to a fancy car."

Tabela 4 – Exemplos de amostras da base de dados Legendas de Imagens COCO.

Notícias do WMT EMNLP2017

A base de dados Notícias do WMT EMNLP2017, do inglês *EMNLP2017 WMT News*[26] (ONDREJ et al., 2017), foi desenvolvida para fomentar a tarefa de tradução de máquina e é composta por pares de documentos, isto é, conjuntos de sentenças em duas linguagens distintas. Entretanto, é possível adaptá-la para a tarefa de geração de texto e utilizar apenas frases correspondentes ao idioma desejado. Nesta tese é utilizado o conjunto de documentos de notícias, as quais foram obtidas através de diversas fontes, tais como ABC News, BBC, Daily Mail, New York Times, Reuters, dentre outras. Ademais, tal conjunto é composto por 27.858 sentenças em inglês, vocabulário de 5.216 *tokens*, tamanho máximo de sequência igual a 25 e tamanho do lote igual a 64. A Tabela 5 descreve alguns exemplos da base de dados em questão.

Sentenças
"My sources have suggested that so far the company sees no reason to change its tax structures , which are perfectly legal ."
"The striker was the centre of headlines the last time these two sides met in the season and he proved so again today ."
"The broadcaster has a fine track record in covering major sporting events ."

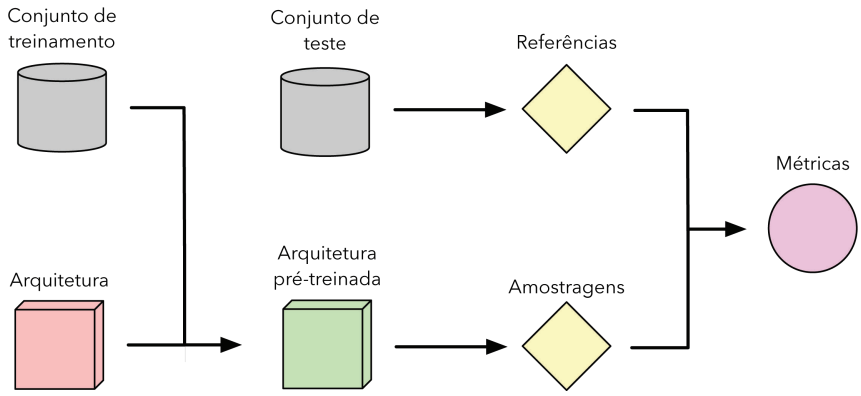
Tabela 5 – Exemplos de sentenças da base de dados Notícias do WMT EMNLP2017.

[26] <http://statmt.org/wmt17/results.html>

7.2 Frente I

O objetivo da Frente I consiste no treinamento dos modelos recorrentes e adversariais propostos, bem como a avaliação de suas capacidades de geração de texto, isto é, a validação do processo de aprendizado e geração através das métricas de avaliação propostas. A Figura 20 ilustra o fluxograma proposto pela Frente I.

Figura 20 – Fluxograma proposto para a Frente I.



Fonte: Elaborado pelo autor.

Seja A uma arquitetura (recorrente ou adversarial), enquanto seja $\mathcal{X}$ o conjunto de dados. Primeiramente, o conjunto de dados é pré-processado com os seguintes passos: transformação para caracteres minúsculos, remoção de caracteres inválidos e pontuações, e tokenização em *tokens* de palavras. Posteriormente, o conjunto de dados pré-processado é separado em conjuntos de treinamento e teste, denotados por $\mathcal{X}_{tr}$ e $\mathcal{X}_{ts}$, respectivamente. O processo de aprendizado dos modelos recorrentes consiste no treinamento da arquitetura A no conjunto de dados $\mathcal{X}_{tr}$ durante uma quantidade de épocas pré-estabelecida. Ademais, o processo de aprendizado dos modelos adversariais consiste no pré-treinamento do gerador e discriminador da arquitetura A no conjunto de dados $\mathcal{X}_{tr}$, seguido do treinamento adversarial. Note que cada modelo adversarial possui um treinamento específico e descrito pelo seu respectivo algoritmo (Capítulos 5 e 6).

Após o treinamento da arquitetura A, os dados do conjunto de teste $\mathcal{X}_{ts}$ são utilizados como sementes[27] de 3 *tokens* para iniciar o processo de geração de texto. Cada semente do conjunto $\mathcal{X}_{ts}$ é amostrada através da Busca Gulosa, Amostragem por Temperatura ($\tau = 0,5$) e Amostragem por k-*Tokens* ($k = 50$) em conjunto da Amostragem Nuclear ($p = 0,95$). Após a amostragem de todas as sementes presentes no conjunto de teste, as sequências amostradas são utilizadas em conjunto das próprias sequências de teste (referências) para o cálculo das métricas BLEU (BLEU-1, BLEU-2 e BLEU3), METEOR e ROUGE (ROUGE-1, ROUGE-2 e ROUGE-L). Finalmente, o valor de cada métrica é calculado a partir da média aritmética entre

[27] O conceito de semente refere-se à utilização de parte da sequência como entrada no processo de geração de texto.

os valores obtidos por cada uma das amostragens. Por exemplo, a métrica final BLEU-1 é a média aritmética entre BLEU-1 com Busca Gulosa, BLEU-1 com Amostragem por Temperatura e BLEU-1 com Amostragem por k-*Tokens*/Nuclear.

7.2.1 Experimentos Propostos

Os experimentos propostos pela Frente I estão divididos em quatro etapas: convergência do treinamento, qualidade da geração de texto, influência da arquitetura dos geradores e discriminadores, e comparação entre funções de perda. A primeira e a segunda etapas estão estritamente relacionadas e constituem um mesmo experimento, o qual é realizado a partir do treinamento e avaliação das arquiteturas RNN, LSTM, GRU, RMC, SeqGAN, MaliGAN, GSGAN, RelGAN, TS-GAN-C e TS-GAN-T nos conjuntos de treinamento e teste, respectivamente. A convergência do treinamento refere-se à análise da evolução da função de perda (NLL) ao longo das épocas de treinamento, enquanto a qualidade da geração de texto está relacionada ao cálculo das métricas de avaliação de texto, tais como, BLEU-1, BLEU-2, BLEU-3, METEOR, ROUGE-1, ROUGE-2 e ROUGE-L.

A terceira etapa constitui o segundo experimento, o qual avalia a influência de diferentes geradores e discriminadores em arquiteturas baseadas em TS-GAN. Em suma, a TS-GAN com Perda Contrastiva (TS-GAN-C) e TS-GAN com Perda Tripla (TS-GAN-T) são novamente treinadas e avaliadas com diferentes geradores (RNN, LSTM, GRU e RMC) e diferentes discriminadores (RNN, LSTM e GRU). Entretanto, este experimento avalia apenas a qualidade do texto gerado através das três métricas mais descritivas, isto é, BLEU-3, METEOR e ROUGE-L.

Finalmente, a quarta etapa constitui o terceiro e último experimento, o qual compara o desempenho da melhor TS-GAN obtida no segundo experimento quando treinada e avaliada com diferentes funções de perda e funções de distância. Em outras palavras, a melhor TS-GAN, isto é, com o melhor gerador e discriminador encontrado no experimento anterior, é treinada e avaliada com quatro funções de perda distintas (Entropia Cruzada, Perda Contrastiva, Perda Tripla com trios difíceis, e Perda Tripla com trios semi-difíceis) e duas funções de distância para a Entropia Cruzada (concatenação e diferença), e três funções de distância para as restantes (*Manhattan*, Euclidiana e Angular).

7.2.2 Configuração Experimental

A configuração experimental da Frente I está dividida em duas tabelas e é válida para todos os experimentos propostos por esta mesma frente. A Tabela 6 descreve o dicionário dos hiperparâmetros utilizados, ou seja, descrição e símbolos matemáticos dos hiperparâmetros de modelos recorrentes e adversariais. Entretanto, note que os hiperparâmetros comuns à todos os modelos foram omitidos, isto é, porcentagens dos conjuntos de treinamento, validação e teste

($80\%/10\%/10\%$), otimizador *Adam*, *tokens* baseados em palavras, frequência mínimas de *tokens* presentes nos dados[28] e temperatura ($\Delta = 5$). Adicionalmente, os hiperparâmetros já citados nas bases de dados, tais como, número de amostras, tamanho do vocabulário, tamanho máximo de sequência e tamanho do lote, permanecem os mesmos para todas as arquiteturas.

Hiperparâmetro	**Símbolo**
Épocas	E
Épocas do pré-treinamento do gerador	E_{P_G}
Épocas do pré-treinamento do discriminador	E_{P_D}
Épocas adversariais	E_A
Épocas adversariais do discriminador	E_{A_D}
Taxa de aprendizado	η
Taxa de aprendizado do gerador	η_G
Taxa de aprendizado do discriminador	η_D
Taxa de pré-aprendizado	η_P
Taxa de pré-aprendizado do gerador	η_{P_G}
Taxa de pré-aprendizado do discriminador	η_{P_D}
Unidades de vetorização	n_e
Unidades ocultas	n_h
Unidades de memória	n_m
Unidades de atenção (tamanho)	$n_a\ (\cdot)$
Blocos de atenção (camadas)	$B\ (\cdot)$
Filtro de convolução i (tamanho)	$F_i\ (\cdot)$
Dropout	p
Quantidade de pares	n_p

Tabela 6 – Dicionário (nome e símbolo) dos hiperparâmetros utilizados na Frente I.

Adicionalmente, a Tabela 7 descreve a configuração experimental dos hiperparâmetros de cada arquitetura para cada base de dados. Esta tabela é primordial para os experimentos em questão, pois detalha a parametrização utilizada nos experimentos propostos, bem como estabelece um padrão de configuração para a avaliação e comparação das arquiteturas.

[28] Todos os *tokens* foram considerados.

Arquitetura	Hiperparâmetro	Amazon	Yelp	COCO	EMNLP
RNN, LSTM, GRU	E	100	100	50	100
	η	$0,0001$	$0,0001$	$0,0001$	$0,0001$
	n_e	512	784	256	512
	n_h	1.024	1.536	512	1.024
RMC	E	100	100	50	100
	η	$0,0001$	$0,0001$	$0,0001$	$0,0001$
	n_e	512	784	256	512
	n_m	1	1	1	1
	n_a	4 (256)	4 (384)	4 (128)	4 (256)
	B	1 (5)	1 (5)	1 (3)	1 (5)
SeqGAN, MaliGAN	E_{P_G}	100	100	50	100
	E_{P_D}	25	25	25	25
	E_A	10	10	10	10
	E_{A_D}	5	5	5	5
	η_P	$0,01$	$0,01$	$0,01$	$0,01$
	η_G	$0,00001$	$0,00001$	$0,00001$	$0,00001$
	η_D	$0,001$	$0,001$	$0,001$	$0,001$
	p	$0,25$	$0,25$	$0,25$	$0,25$
	n_e	512	784	256	512
	n_h	1.024	1.536	512	1.024
	F_0	3 (128)	5 (128)	3 (64)	3 (128)
	F_1	5 (256)	7 (256)	5 (128)	5 (256)
	F_2	7 (512)	9 (512)	7 (256)	7 (512)

Arquitetura	Hiperparâmetro	Amazon	Yelp	COCO	EMNLP
GSGAN	E_{P_G}	100	100	50	100
	E_A	10	10	10	10
	η_P	$0,01$	$0,01$	$0,01$	$0,01$
	η_G	$0,00001$	$0,00001$	$0,00001$	$0,00001$
	η_D	$0,001$	$0,001$	$0,001$	$0,001$
	n_e	512	784	256	512
	n_h	1.024	1.536	512	1.024
RelGAN	E_{P_G}	100	100	50	100
	E_A	10	10	10	10
	η_P	$0,01$	$0,01$	$0,01$	$0,01$
	η_G	$0,00001$	$0,00001$	$0,00001$	$0,00001$
	η_D	$0,001$	$0,001$	$0,001$	$0,001$
	n_e	512	784	256	512
	n_m	1	1	1	1
	n_a	4 (256)	4 (384)	4 (128)	4 (256)
	B	1 (5)	1 (5)	1 (3)	1 (5)
	F_0	3 (128)	5 (128)	3 (64)	3 (128)
	F_1	5 (256)	7 (256)	5 (128)	5 (256)
	F_2	7 (512)	9 (512)	7 (256)	7 (512)

Arquitetura	Hiperparâmetro	Amazon	Yelp	COCO	EMNLP
TS-GAN-C, TS-GAN-T	E_{P_G}	100	100	50	100
	E_{P_D}	25	25	25	25
	E_A	10	10	10	10
	E_{A_D}	5	5	5	5
	η_{P_G}	$0,01$	$0,01$	$0,01$	$0,01$
	η_{P_D}	$0,01$	$0,01$	$0,01$	$0,01$
	η_G	$0,00001$	$0,00001$	$0,00001$	$0,00001$
	η_D	$0,001$	$0,001$	$0,001$	$0,001$
	n_e	512	784	256	512
	n_h	1.024	1.536	512	1.024
	n_p	256	256	128	256

Tabela 7 – Configuração experimental das arquiteturas e de seus hiperparâmetros para cada base de dados.

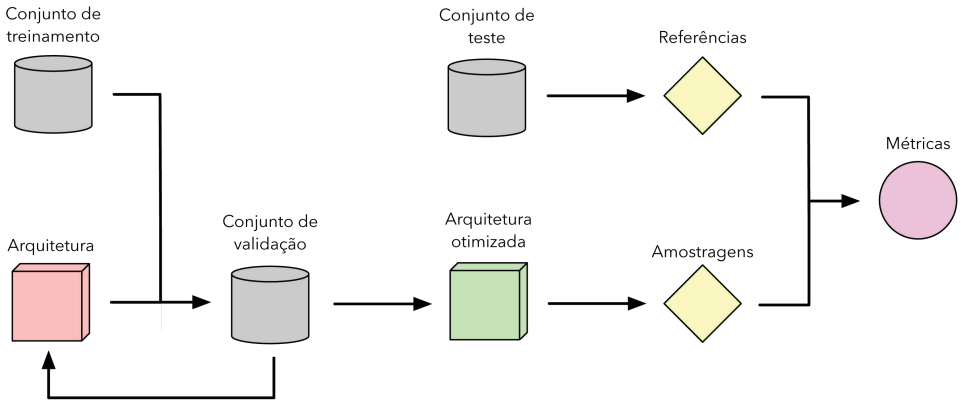
Finalmente, todas as arquiteturas foram avaliadas 10 vezes para cada base de dados, ou seja, as bases de dados foram particionadas utilizando um procedimento de validação cruzada *hold-out* de 10 iterações, dividindo-as em 10 conjuntos de treinamento, validação (utilizada apenas na otimização) e teste, com porcentagens iguais à 80%, 10% e 10%, respectivamente. Também foi aplicado o teste de sinais de Wilcoxon (5% de significância) nos resultados do conjunto de teste a fim de estabelecer uma análise estatística robusta. Em relação ao código-fonte, foram utilizadas as bibliotecas Dualing[29] e NALP[30], as quais estão disponíveis para livre acesso na plataforma GitHub. Todas as implementações e *scripts* dos experimentos estão disponíveis na mesma plataforma[31].

[29] <https://github.com/gugarosa/dualing>
[30] <https://github.com/gugarosa/nalp>
[31] <https://github.com/gugarosa/tsgan_nlg>

7.3 Frente II

O objetivo da Frente II consiste na análise do processo de otimização dos hiperparâmetros da TS-GAN. A ideia por trás de tal metodologia é verificar se existem conjuntos de hiperparâmetros mais aptos para cada base de dados em questão, bem como investigar possíveis aperfeiçoamentos para a rede adversarial proposta. A Figura 21 ilustra o fluxograma proposto pela Frente II.

Figura 21 – Fluxograma proposto para a Frente II.



Fonte: Elaborado pelo autor.

Seja A uma arquitetura da TS-GAN, enquanto seja $\mathcal{X}$ o conjunto de dados. Para o procedimento de otimização, é necessário separar o conjunto de dados em conjuntos de treinamento, validação e teste, dados por $\mathcal{X}_{tr}$, $\mathcal{X}_{val}$ e $\mathcal{X}_{ts}$, respectivamente. Adicionalmente, seja $\boldsymbol{v}$ o vetor de variáveis de decisão (hiperparâmetros), dado que $\boldsymbol{v} = (v_1, v_2, \ldots, v_n) \mid \boldsymbol{v} \in \Re^n$ e seja n o número de hiperparâmetros a serem otimizados. O objetivo da otimização é encontrar o conjunto mais apto de hiperparâmetros, denotado por $\boldsymbol{v}^*$, onde $L(\boldsymbol{v}^*) \leq L(\boldsymbol{v})$, tal que $L(\cdot)$ seja a métrica $1 - \text{BLEU-3}$ obtida através da avaliação de T no conjunto de validação. Desta forma, é possível descrever o processo de otimização de acordo com a Equação 67:

$$\min_{\forall \boldsymbol{v} \in \Re^n} L(\boldsymbol{v}), \tag{67}$$

Após o processo de otimização, uma nova arquitetura A é treinada com o conjunto de hiperparâmetros $\boldsymbol{v}^*$ e avaliada da mesma forma proposta pela Frente I.

7.3.1 Experimento Proposto

O experimento proposto (quarto experimento) pela Frente II está dividido em três etapas: convergência da otimização, influência dos hiperparâmetros otimizados e análise do custo computacional. Todas as três etapas estão estritamente relacionadas e constituem o mesmo experimento, o qual é realizado a partir da otimização da melhor arquitetura de TS-GAN,

isto é, a arquitetura que obteve as melhores métricas no terceiro experimento. Tal proposição objetiva reduzir os custos computacionais ao apenas otimizar a melhor arquitetura possível, dado que o terceiro experimento encontra a melhor função de perda e função de distância de uma TS-GAN que já possui o melhor gerador e discriminador encontrados pelo segundo experimento. Ademais, tal arquitetura será otimizada com as meta-heurísticas BA, CS, FA, GP e PSO.

Sua convergência de otimização refere-se à análise da evolução da minimização da função objetivo ($1 -$ BLEU-3) ao longo das iterações da otimização, enquanto a influência dos hiperparâmetros otimizados está relacionada à avaliação do conjunto de teste e ao cálculo das métricas de avaliação, tais como, BLEU-1, BLEU-2, BLEU-3, METEOR, ROUGE-1, ROUGE-2 e ROUGE-L. Finalmente, a análise do custo computacional consiste na comparação entre os tempos de treinamento das arquitetura sem otimização e das arquiteturas otimizadas (tempo de otimização $+$ tempo de treinamento com parâmetros otimizados).

7.3.2 Configuração Experimental

A configuração experimental da Frente II está dividida em duas tabelas a fim de facilitar a estruturação e clareza do experimento proposto. A Tabela 8 descreve a configuração experimental referente aos algoritmos de otimização meta-heurística, enquanto a Tabela 9 apresenta o intervalo de busca dos hiperparâmetros a serem otimizados. Note que o dicionário de hiperparâmetros e os conjuntos de hiperparâmetros para cada arquitetura foram omitidos pois são os mesmos descritos pelas Tabelas 6 e 7, respectivamente. Adicionalmente, as quantidades de agentes utilizados (5) e iterações de otimização (10) também foram omitidas pois são as mesmas para todos os algoritmos utilizados.

Meta-heurística	**Parâmetros**
BA	$f_{\min} = 0,\ f_{\max} = 2,\ A = 0,5,\ r = 0,5$
CS	$\alpha = 0,3,\ \lambda = 1,5,\ p_a = 0,2$
FA	$\alpha = 0,5,\ \beta_0 = 0,2,\ \gamma = 1,0$
GP	$p_r = 0,25,\ p_m = 0,1,\ p_c = 0,2$
PSO	$\mu = 0,7,\ c_1 = 1,7,\ c_2 = 1,7$

Tabela 8 – Configuração experimental das meta-heurísticas.

Hiperparâmetro	Amazon	Yelp	COCO	EMNLP
η_G	$[0,00001,\ 0,1]$	$[0,00001,\ 0,1]$	$[0,00001,\ 0,1]$	$[0,00001,\ 0,1]$
η_D	$[0,001,\ 0,1]$	$[0,001,\ 0,1]$	$[0,001,\ 0,1]$	$[0,001,\ 0,1]$
n_e	$[128,\ 1.024]$	$[128,\ 1.536]$	$[128,\ 512]$	$[128,\ 1.024]$
n_h	$[128,\ 2.048]$	$[128,\ 3.072]$	$[128,\ 1.024]$	$[128,\ 2.048]$
Δ	$[1,\ 9]$	$[1,\ 9]$	$[1,\ 9]$	$[1,\ 9]$

Tabela 9 – Configuração experimental do intervalo de busca dos hiperparâmetros para cada base de dados.

A otimização foi realizada uma única vez com as meta-heurísticas e hiperparâmetros mencionados nas Tabelas 8 e 9. Após o processo de busca, todas as arquiteturas foram novamente avaliadas com os hiperparâmetros otimizados, seguindo os mesmos protocolos estabelecidos pela Frente I[32]. Em relação ao código-fonte, foram utilizadas as mesmas implementações propostas pela Frente I em conjunto da biblioteca de otimização Opytimizer[33] (ROSA et al., 2019), a qual está disponível para livre acesso na plataforma GitHub. Novamente, todos as implementações e executáveis dos experimentos estão disponíveis no mesmo repositório descrito pela Frente I.

[32] Os protocolos definidos são as 10 rodadas de experimentos, mesma partições das bases de dados e a aplicação do teste de sinais de Wilcoxon com significância de 5%.

[33] <https://github.com/gugarosa/opytimizer>

8 Experimentos e Resultados

O presente capítulo descreve os resultados experimentais obtidos através da condução do Capítulo 7. Ademais, ele está organizado de acordo com as definições do capítulo anterior: Frente I (arquitetura proposta) e Frente II (otimização).

8.1 Frente I

A Frente I propõe o treinamento da arquitetura Rede Adversarial Generativa por Similaridade Textual em bases de dados e sua comparação com arquiteturas estado da arte. Os experimentos estão divididos em quatro análises: convergência do treinamento (análise da função de perda), qualidade da geração de texto (análise de métricas), influência das arquiteturas dos geradores e discriminadores, e comparação entre funções de perda.

8.1.1 Convergência do Treinamento

O treinamento de uma arquitetura neural para a geração de texto envolve a otimização de sua função de perda em conjunto dos dados de treinamento. Ao analisarmos a convergência desta função, é possível observar a estabilidade e extensibilidade do treinamento. Usualmente, as funções de perda seguem uma taxa de convergência mais acelerada nos períodos iniciais e atingem patamares mais estáveis ao passar das iterações.

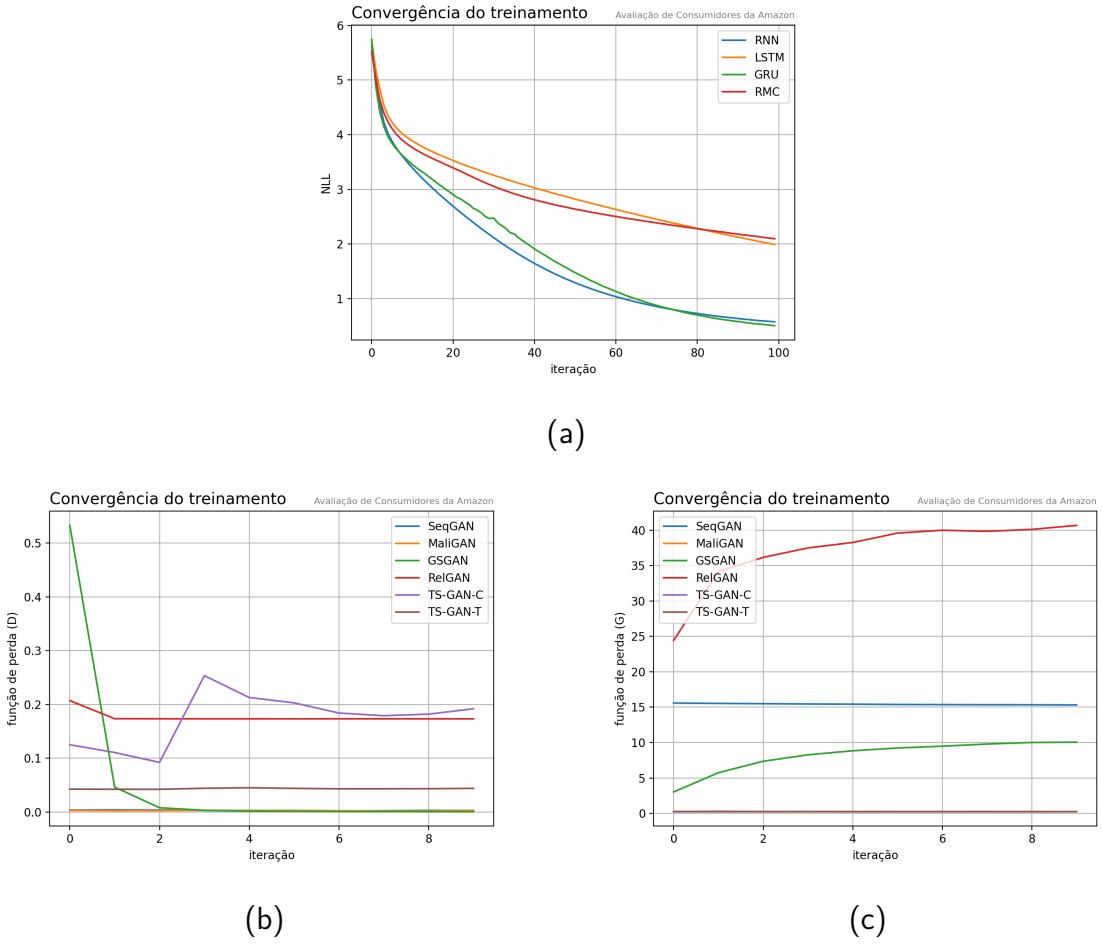
A Figura 22 ilustra a convergência da função de perda de treinamento na base de dados Avaliação dos Consumidores da Amazon. O item (a) da figura demonstra o comportamento das redes recorrentes, onde é possível observar que as RNNs e GRUs obtiveram uma melhor convergência (em comparação às LSTMs e RMCs) considerando um mesmo período de tempo. Tal comportamento reforça a rápida convergência da RNNs e GRUs nas épocas iniciais de treinamento, enquanto as LSTMs e RMCs (derivadas das LSTMs) são mais efetivas em treinamento mais longos[34]. Ademais, a Figura 22 ilustra que ainda há espaço de convergência para as LSTMs e RMCs, e possivelmente, equiparação de suas curvas quando treinadas por mais iterações.

Os itens (b) e (c) da Figura 22 ilustram o comportamento do treinamento dos discriminadores e geradores das redes adversariais, respectivamente. O comportamento dos discriminadores das TS-GAN-C e GSGAN aparentaram uma maior instabilidade, enquanto os discriminadores da SeqGAN, MaliGAN, RelGAN e TS-GAN-T não sofreram modificações ao longo do treinamento. Por outro lado, os geradores da RelGAN e GSGAN sofreram oscilações ao

[34] Os mecanismos de portões das LSTMs e o mecanismo de atenção das RMCs induzem uma robustez maior, e consequentemente, necessitam de mais dados para a convergência.

longo de seus treinamentos, possivelmente para acomodar as atualizações introduzidas por seus discriminadores. Embora o discriminador da TS-GAN-C tenha apresentado um comportamento instável, o seu gerador continuou estável ao longo do treinamento, indicando o sucesso da discriminação para o contexto apresentado.

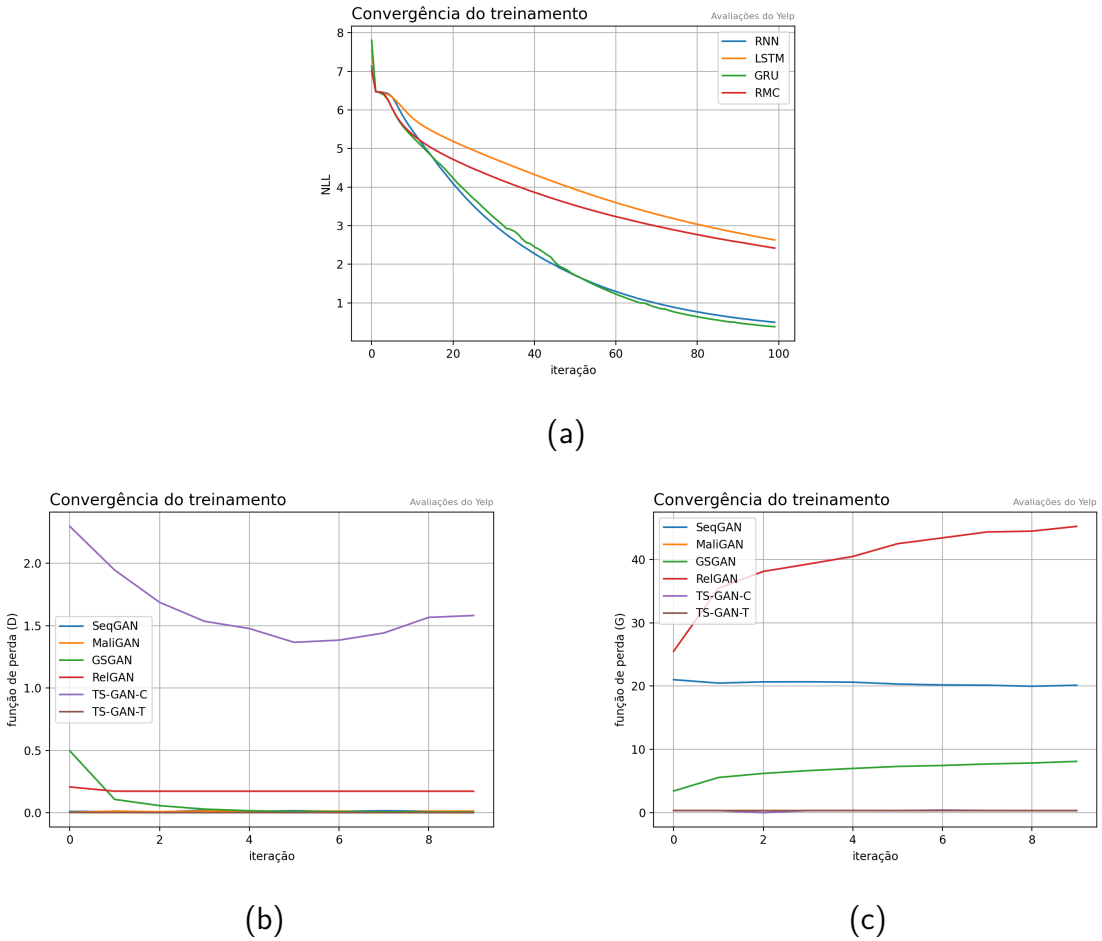
Figura 22 – Convergência da função de perda de treinamento na base de dados Avaliação dos Consumidores da Amazon: (a) recorrente, (b) discriminador adversarial e (c) gerador adversarial.



Fonte: Elaborado pelo autor.

A Figura 23 ilustra a convergência da função de perda de treinamento na base de dados Avaliações do Yelp: (a) recorrente, (b) discriminador adversarial e (c) gerador adversarial. O mesmo comportamento da Figura 22 é ilustrado pela Figura 23, isto é, o discriminador da TS-GAN apresenta maiores instabilidades, enquanto o gerador da RelGAN possui as maiores oscilações. Tal comportamento é explicado devido as similaridades entre os hiperparâmetros das duas bases de dados (tamanho de vocabulário, tamanho de sequência e quantidade de amostras).

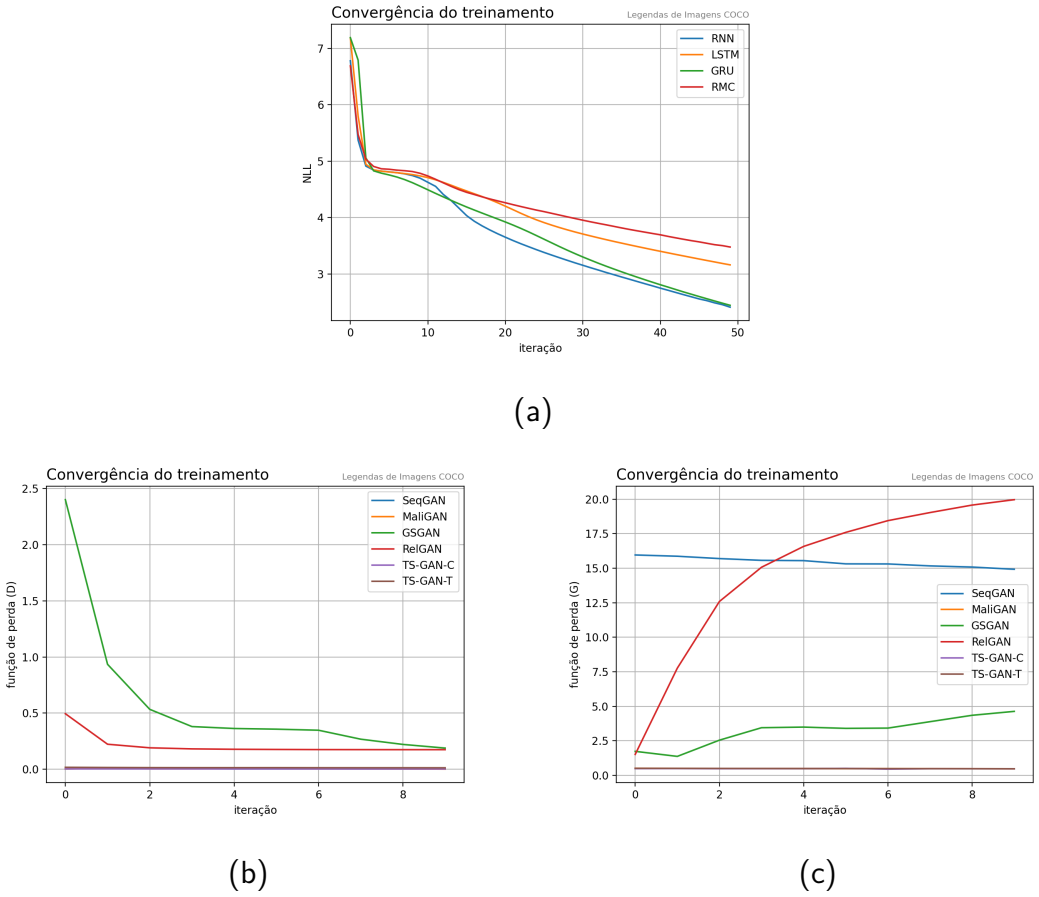
Figura 23 – Convergência da função de perda de treinamento na base de dados Avaliações do Yelp: (a) recorrente, (b) discriminador adversarial e (c) gerador adversarial.



Fonte: Elaborado pelo autor.

A Figura 24 ilustra a convergência da função de perda de treinamento na base de dados Legendas de Imagens COCO: (a) recorrente, (b) discriminador adversarial e (c) gerador adversarial. Nesta figura, é possível observar um comportamento diferente das figuras anteriores. i possui apenas 1.000 amostras, tamanho de sequência igual a 10, 1.347 *tokens* e é treinada com a metade de amostras por lote (32). O item (a) ilustra que as arquiteturas recorrentes estão em processo de convergência, enquanto os itens (b) e (c) demonstram que a GSGAN obteve um discriminador com oscilações, e os geradores da GSGAN, MaliGAN e SeqGAN sofreram pequenas variações ao longo do treinamento.

Figura 24 – Convergência da função de perda de treinamento na base de dados Legendas de Imagens COCO: (a) recorrente, (b) discriminador adversarial e (c) gerador adversarial.

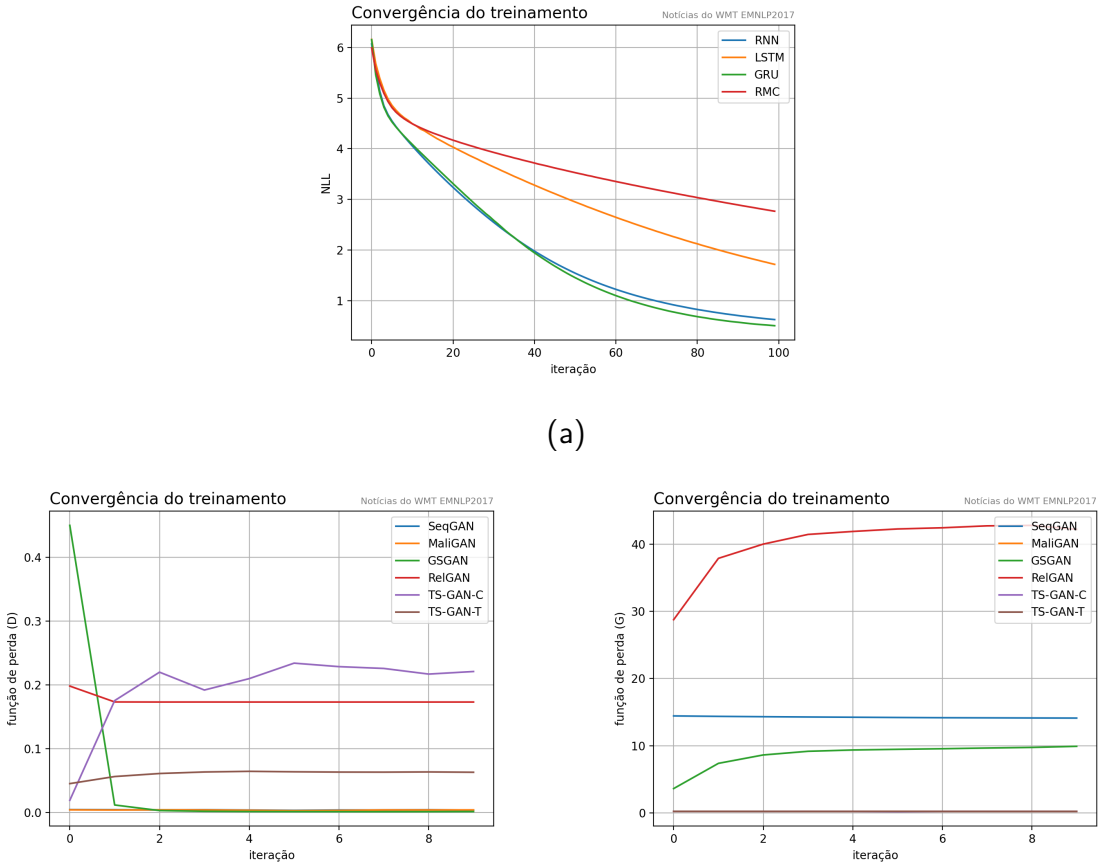


Fonte: Elaborado pelo autor.

Finalmente, a Figura 25 apresenta a convergência da função de perda de treinamento na base de dados Notícias do WMT EMNLP2017: (a) recorrente, (b) discriminador adversarial e (c) gerador adversarial. Nesta figura, as RNNs e GRUs obtiveram convergências mais acentuadas, a TS-GAN-C e GSGAN sofreram com instabilidade em seus discriminadores, e os geradores da RelGAN e GSGAN oscilaram. Novamente, tal comportamento é explicado pela similaridade desta base de dados em relação às bases de dados Avaliações de Consumidores da Amazon e Avaliações do Yelp.

Uma análise comparativa das funções de perda de treinamento em diferentes arquiteturas de redes neurais para geração de texto mostra que as RNNs e GRUs têm uma convergência mais rápida nas épocas iniciais, enquanto as LSTMs e RMCs são mais efetivas em treinamentos mais longos. O comportamento dos discriminadores das TS-GAN-C e GSGAN é mais instável, enquanto os geradores da RelGAN e GSGAN sofrem oscilações ao longo do treinamento. No entanto, os resultados dos testes de Wilcoxon mostram que as TS-GAN-C e TS-GAN-T têm desempenho estatisticamente igual às outras arquiteturas, mas apresentam efetividades superiores em números absolutos.

Figura 25 – Convergência da função de perda de treinamento na base de dados Notícias do WMT EMNLP2017: (a) recorrente, (b) discriminador adversarial e (c) gerador adversarial.



Fonte: Elaborado pelo autor.

8.1.2 Qualidade da Geração de Texto

A qualidade da geração de texto é analisada por métricas aplicadas no conjunto de teste do processo de geração de texto. Cada semente é amostrada através da Busca Gulosa, Amostragem por Temperatura ($\tau = 0,5$) e Amostragem por k-*Tokens* ($k = 50$) em conjunto da Amostragem Nuclear ($p = 0,95$), e a saída gerada é usada para calcular as métricas BLEU (BLEU-1, BLEU-2 e BLEU3), METEOR e ROUGE (ROUGE-1, ROUGE-2 e ROUGE-L)[35].

A Tabela 10 descreve as métricas dos modelos avaliados no conjunto de teste da base de dados Avaliação de Consumidores da Amazon. Nesta tabela, é possível observar que o treinamento acentuado das RNNs e GRUs auxiliaram a sua capacidade de geração de texto e obtiveram os melhores resultados de acordo com o teste de Wilcoxon para todas as métricas analisadas (BLEU-1, BLEU-2, BLEU-3, METEOR, ROUGE-1 e ROUGE-L). Por outro lado, dentre os modelos adversariais, a TS-GAN-C obteve quase todos os melhores resultados (exceto

[35] As métricas finais são as médias aritméticas entre os valores obtidos pelas amostragens.

pelas métricas baseadas em ROUGE), indicando que a rede favoreceu a precisão ao invés da sensitividade, isto é, gerou textos menos diversos porém mais similares às referências.

Tipo	Modelo	BLEU-1	BLEU-2	BLEU-3	METEOR	ROUGE-1	ROUGE-2	ROUGE-L
Recorrente	RNN	$\mathbf{35,16 \pm 0,36}$	$\mathbf{24,38 \pm 0,37}$	$\mathbf{18,87 \pm 0,39}$	$\mathbf{29,35 \pm 0,36}$	$\mathbf{35,22 \pm 0,36}$	$\mathbf{16,93 \pm 0,35}$	$\mathbf{31,92 \pm 0,36}$
	LSTM	$24,34 \pm 1,11$	$16,39 \pm 0,81$	$11,85 \pm 0,65$	$23,48 \pm 0,47$	$33,72 \pm 0,50$	$15,07 \pm 0,45$	$31,41 \pm 0,31$
	GRU	$\mathbf{34,25 \pm 0,32}$	$\mathbf{23,32 \pm 0,39}$	$\mathbf{17,75 \pm 0,42}$	$\mathbf{28,47 \pm 0,33}$	$\mathbf{34,49 \pm 0,29}$	$\mathbf{15,99 \pm 0,38}$	$\mathbf{31,27 \pm 0,29}$
	RMC	$22,58 \pm 12,03$	$14,88 \pm 7,70$	$10,77 \pm 5,44$	$22,87 \pm 4,37$	$33,16 \pm 2,28$	$14,90 \pm 0,54$	$30,88 \pm 1,09$
Adversarial	SeqGAN	$26,06 \pm 0,64$	$17,16 \pm 0,40$	$12,46 \pm 0,30$	$21,98 \pm 0,45$	$26,68 \pm 0,71$	$11,58 \pm 0,30$	$23,95 \pm 0,59$
	MaliGAN	$26,09 \pm 0,58$	$17,19 \pm 0,42$	$12,47 \pm 0,37$	$21,96 \pm 0,47$	$26,74 \pm 0,50$	$11,62 \pm 0,29$	$24,00 \pm 0,39$
	GSGAN	$26,00 \pm 2,61$	$16,55 \pm 1,61$	$11,66 \pm 1,16$	$21,68 \pm 1,72$	$28,11 \pm 2,29$	$11,39 \pm 0,89$	$25,48 \pm 1,61$
	RelGAN	$24,52 \pm 0,39$	$16,13 \pm 0,31$	$11,52 \pm 0,29$	$21,26 \pm 0,39$	$29,68 \pm 0,26$	$13,05 \pm 0,37$	$27,06 \pm 0,32$
	TS-GAN-C	$28,36 \pm 0,89$	$18,59 \pm 0,59$	$13,47 \pm 0,47$	$23,52 \pm 0,71$	$29,28 \pm 1,13$	$12,59 \pm 0,51$	$26,19 \pm 1,05$
	TS-GAN-T	$26,24 \pm 0,21$	$17,24 \pm 0,13$	$12,49 \pm 0,09$	$22,06 \pm 0,16$	$26,87 \pm 0,25$	$11,61 \pm 0,08$	$24,09 \pm 0,24$

Tabela 10 – Métricas dos modelos avaliados no conjunto de teste da base de dados Avaliação de Consumidores da Amazon.

Outro ponto a ser ponderado é o baixo desvio padrão dos modelos adversariais, especialmente a TS-GAN-T. O discriminador com função de perda baseado em triplas (âncora, positivo e negativo) produz uma concentração mais dinâmica de informações, ou seja, consegue explorar a "falta"de dados ao criar mais amostras através da combinação de triplas. Tal mecanismo aumenta a estabilidade do processo de treinamento (visto nas curvas de seu discriminador e gerador) e previne a variabilidade dos resultados quando treinada por repetidas vezes[36].

As arquiteturas que utilizam o mecanismo de atenção (RMC e RelGAN) apresentaram as piores médias de métricas, enquanto a RMC obteve o pior desvio padrão, fato explicável pela falta de informação (poucas amostras), tamanho de arquitetura e tempo de treinamento. É importante lembrar que o mecanismo de atenção introduz uma grande capacidade ao modelo neural, entretanto, tal capacidade precisa ser explorada para que o mecanismo seja efetivo (maiores quantidades de dados, treinamentos mais longos e arquiteturas maiores).

A Tabela 11 descreve as métricas dos modelos avaliados no conjunto de teste da base de dados Avaliações do Yelp. Nesta tabela, é possível verificar o mesmo comportamento apresentado no início da análise, ou seja, as RNNs e GRUs com efetividades maiores às LSTMs e RMCs. Adicionalmente, a SeqGAN, TS-GAN-C e TS-GAN-T também obtiveram os melhores resultados de acordo com o teste de Wilcoxon.

[36] Instruções aleatórias fixas por uma semente de aleatoriedade.

Tipo	Modelo	BLEU-1	BLEU-2	BLEU-3	METEOR	ROUGE-1	ROUGE-2	ROUGE-L
Recorrente	RNN	$\mathbf{31,33 \pm 0,66}$	$\mathbf{21,27 \pm 0,74}$	$\mathbf{17,34 \pm 0,79}$	$\mathbf{25,34 \pm 0,63}$	$\mathbf{32,53 \pm 0,64}$	$\mathbf{14,85 \pm 0,73}$	$\mathbf{27,86 \pm 0,63}$
	LSTM	$17,51 \pm 2,53$	$9,48 \pm 1,20$	$6,09 \pm 0,69$	$15,97 \pm 0,54$	$26,17 \pm 0,41$	$7,69 \pm 0,27$	$22,55 \pm 0,44$
	GRU	$\mathbf{35,15 \pm 1,02}$	$\mathbf{25,41 \pm 1,17}$	$\mathbf{21,58 \pm 1,23}$	$\mathbf{29,00 \pm 1,09}$	$\mathbf{35,55 \pm 1,00}$	$\mathbf{18,53 \pm 1,18}$	$\mathbf{30,99 \pm 0,99}$
	RMC	$22,90 \pm 6,45$	$12,25 \pm 3,32$	$8,01 \pm 2,12$	$18,32 \pm 1,93$	$28,13 \pm 1,38$	$8,23 \pm 0,49$	$23,83 \pm 0,49$
Adversarial	SeqGAN	$\underline{\mathbf{39,07 \pm 7,98}}$	$\underline{\mathbf{31,04 \pm 8,58}}$	$\underline{\mathbf{27,76 \pm 8,79}}$	$\underline{\mathbf{34,04 \pm 7,68}}$	$\underline{\mathbf{39,91 \pm 7,10}}$	$\underline{\mathbf{25,11 \pm 7,94}}$	$\underline{\mathbf{36,07 \pm 7,19}}$
	MaliGAN	$\mathbf{32,55 \pm 7,96}$	$24,13 \pm 8,32$	$20,70 \pm 8,46$	$27,71 \pm 7,49$	$\mathbf{33,92 \pm 7,26}$	$18,60 \pm 7,62$	$29,97 \pm 7,24$
	GSGAN	$25,82 \pm 4,70$	$16,59 \pm 4,88$	$12,87 \pm 4,96$	$21,23 \pm 4,29$	$28,78 \pm 3,92$	$11,89 \pm 4,18$	$24,46 \pm 3,88$
	RelGAN	$25,82 \pm 4,70$	$16,59 \pm 4,88$	$12,87 \pm 4,96$	$21,23 \pm 4,29$	$28,78 \pm 3,92$	$11,89 \pm 4,18$	$24,46 \pm 3,88$
	TS-GAN-C	$\mathbf{33,85 \pm 8,21}$	$\mathbf{25,36 \pm 8,77}$	$\mathbf{21,91 \pm 8,97}$	$\mathbf{28,91 \pm 7,88}$	$\mathbf{35,35 \pm 7,13}$	$\mathbf{19,77 \pm 8,08}$	$\mathbf{31,38 \pm 7,22}$
	TS-GAN-T	$\mathbf{32,77 \pm 6,18}$	$\mathbf{23,90 \pm 6,86}$	$\mathbf{20,34 \pm 7,08}$	$\mathbf{27,55 \pm 6,13}$	$\mathbf{34,15 \pm 5,67}$	$\mathbf{18,11 \pm 6,51}$	$\mathbf{30,05 \pm 5,85}$

Tabela 11 – Métricas dos modelos avaliados no conjunto de teste da base de dados Avaliações do Yelp.

Esta base de dados utiliza um tamanho de sequência maior (60% maior), o qual introduz um dinamismo ao treinamento dos modelos, deixando-os mais instáveis. Tal aumento de sequência dificulta o aprendizado dos modelos pois adiciona um conjunto de informações a serem analisadas. Também é interessante salientar, que apesar da dificuldade da tarefa da Avaliações do Yelp, as redes adversariais obtiveram os melhores resultados, mostrando a sua versatilidade em relação aos modelos recorrentes. Por outro lado, a alta variabilidade é algo que precisa ser aprimorado pois indica uma instabilidade do processo de treinamento. Note que as TS-GAN-C e TS-GAN-T diferem da MaliGAN apenas em seus discriminadores (mesmo algoritmo de treinamento) e obtiveram resultados superiores dentre as três. Apesar dos resultados serem estatisticamente iguais pelo teste de Wilcoxon, as TS-GAN-C e TS-GAN-T atingiram efetividades 4% e $0,67\%$ superiores em números absolutos, respectivamente.

A Tabela 12 descreve as métricas dos modelos avaliados no conjunto de teste da base de dados Legendas de Imagens COCO. Nesta base de dados (tarefa mais fácil dentre as quatro), as redes adversariais, exceto pela RelGAN, obtiveram os melhores resultados de acordo com o teste de Wilcoxon. A MaliGAN obteve o melhor resultado absoluto em todas as métricas, seguida da TS-GAN-T e TS-GAN-C. Todos os modelos adversariais, acrescentados da RNN e GRU, obtiveram desvios padrões superior a $1,47$.

Tipo	Modelo	BLEU-1	BLEU-2	BLEU-3	METEOR	ROUGE-1	ROUGE-2	ROUGE-L
Recorrente	RNN	$32,90 \pm 5,01$	$25,85 \pm 3,10$	$19,43 \pm 2,03$	$25,98 \pm 4,50$	$33,35 \pm 5,28$	$20,59 \pm 1,96$	$33,17 \pm 5,18$
	LSTM	$40,47 \pm 0,75$	$29,82 \pm 0,73$	$21,86 \pm 0,58$	$32,80 \pm 1,11$	$45,54 \pm 0,58$	$24,44 \pm 0,81$	$45,25 \pm 0,55$
	GRU	$30,81 \pm 1,75$	$24,47 \pm 0,95$	$18,51 \pm 0,61$	$23,53 \pm 1,42$	$30,84 \pm 1,74$	$19,46 \pm 0,43$	$30,58 \pm 1,66$
	RMC	$39,29 \pm 0,89$	$29,48 \pm 0,69$	$21,88 \pm 0,62$	$29,08 \pm 1,42$	$41,51 \pm 1,72$	$23,33 \pm 1,07$	$40,92 \pm 1,77$
Adversarial	SeqGAN	$\mathbf{58,64 \pm 3,55}$	$\mathbf{49,13 \pm 3,89}$	$\mathbf{42,74 \pm 4,20}$	$\mathbf{51,51 \pm 4,06}$	$\mathbf{59,01 \pm 3,40}$	$\mathbf{41,23 \pm 3,90}$	$\mathbf{58,30 \pm 3,33}$
	MaliGAN	$\underline{\mathbf{59,65 \pm 1,47}}$	$\underline{\mathbf{50,12 \pm 1,75}}$	$\underline{\mathbf{43,78 \pm 1,93}}$	$\underline{\mathbf{52,52 \pm 1,90}}$	$\underline{\mathbf{59,87 \pm 1,44}}$	$\underline{\mathbf{42,09 \pm 1,91}}$	$\underline{\mathbf{59,18 \pm 1,49}}$
	GSGAN	$\mathbf{58,64 \pm 1,94}$	$\mathbf{48,99 \pm 2,10}$	$\mathbf{42,58 \pm 2,17}$	$\mathbf{51,28 \pm 2,46}$	$\mathbf{59,03 \pm 1,92}$	$\mathbf{41,02 \pm 2,08}$	$\mathbf{58,26 \pm 1,84}$
	RelGAN	$43,69 \pm 1,53$	$33,67 \pm 1,54$	$26,48 \pm 1,54$	$36,31 \pm 2,01$	$46,85 \pm 1,60$	$27,59 \pm 1,53$	$46,41 \pm 1,64$
	TS-GAN-C	$\mathbf{58,78 \pm 2,88}$	$\mathbf{49,17 \pm 3,17}$	$\mathbf{42,69 \pm 3,41}$	$\mathbf{51,62 \pm 3,29}$	$\mathbf{59,16 \pm 2,58}$	$\mathbf{41,19 \pm 3,09}$	$\mathbf{58,43 \pm 2,45}$
	TS-GAN-T	$\mathbf{59,24 \pm 1,77}$	$\mathbf{49,78 \pm 2,20}$	$\mathbf{43,44 \pm 2,37}$	$\mathbf{51,99 \pm 2,26}$	$\mathbf{59,49 \pm 1,76}$	$\mathbf{41,80 \pm 2,44}$	$\mathbf{58,70 \pm 1,84}$

Tabela 12 – Métricas dos modelos avaliados no conjunto de teste da base de dados Legendas de Imagens COCO.

Embora a RelGAN obteve o pior resultado dentre os modelos adversariais, todos os modelos adversariais foram capazes de aprimorar o processo de treinamento dos modelos recorrentes. A LSTM foi o único modelo recorrente a passar da marca de 40 pontos para a métrica BLEU-1, enquanto todos os modelos adversariais atingiram mais de $43,69$ pontos. A efetividade dos modelos adversariais é significativa em todas as métricas, inclusive atingindo picos de 136% de aumento (BLEU-3 da MaliGAN comparado ao BLEU-3 da GRU).

Finalmente, a Tabela 13 apresenta o mesmo comportamento descrito pela Tabela 10, novamente explicado pela similaridade dos hiperparâmetros utilizados para as bases de dados Notícias do WMT EMNLP2017 e Avaliação de Consumidores de Amazon. As RNNs e GRUs atingiram as melhores métricas de acordo com o teste de Wilcoxon, superando a performance de todos os outros modelos analisados. Entretanto, é importante salientar que a TS-GAN-T foi o primeiro modelo inferior às RNNs e GRUs, atingindo resultados superiores a $4,25\%$ no pior caso (BLEU-1 da LSTM) e $15,47\%$ no melhor caso (BLEU-1 da RMC). Esta superioridade demonstra a capacidade do discriminador contrastivo quando comparado aos discriminadores de outras arquiteturas adversariais.

As Tabelas 10 e 11 mostram que o treinamento aprimorado de RNNs e GRUs ajudou a melhorar sua capacidade de geração de texto e obteve os melhores resultados de acordo com o teste de Wilcoxon para todas as métricas avaliadas (BLEU-1, BLEU-2, BLEU-3, METEOR, ROUGE-1 e ROUGE-L). Dentre os modelos adversariais, a TS-GAN-C obteve quase todos os melhores resultados (exceto pelas métricas baseadas em ROUGE), indicando que a rede favoreceu a precisão em vez da diversidade, ou seja, gerou textos menos diversos, mas mais similares às referências. Outro ponto a ser observado é o baixo desvio padrão dos modelos adversariais, especialmente a TS-GAN-T. O discriminador com função de perda baseada em triplas (âncora, positivo e negativo) produz uma distribuição de informações mais dinâmica, ou seja, é capaz de explorar a "falta"de dados ao criar mais amostras através da combinação

Tipo	Modelo	BLEU-1	BLEU-2	BLEU-3	METEOR	ROUGE-1	ROUGE-2	ROUGE-L
Recorrente	RNN	$\underline{\mathbf{34,05 \pm 0,17}}$	$\underline{\mathbf{25,64 \pm 0,21}}$	$\underline{\mathbf{21,06 \pm 0,23}}$	$\underline{\mathbf{29,87 \pm 0,19}}$	$\underline{\mathbf{34,12 \pm 0,17}}$	$\underline{\mathbf{19,35 \pm 0,22}}$	$\mathbf{32,09 \pm 0,18}$
	LSTM	$23,98 \pm 0,88$	$16,26 \pm 0,57$	$11,83 \pm 0,45$	$22,26 \pm 0,40$	$30,57 \pm 0,30$	$13,92 \pm 0,47$	$28,81 \pm 0,37$
	GRU	$\mathbf{33,92 \pm 0,61}$	$\mathbf{25,47 \pm 0,75}$	$\mathbf{20,88 \pm 0,83}$	$\mathbf{29,70 \pm 0,68}$	$\mathbf{34,10 \pm 0,57}$	$\mathbf{19,22 \pm 0,77}$	$\underline{\mathbf{32,11 \pm 0,59}}$
	RMC	$21,65 \pm 7,05$	$14,59 \pm 4,46$	$10,54 \pm 3,11$	$21,52 \pm 2,20$	$30,54 \pm 0,87$	$14,07 \pm 1,20$	$28,93 \pm 0,64$
Adversarial	SeqGAN	$22,48 \pm 0,31$	$15,73 \pm 0,24$	$11,82 \pm 0,22$	$19,33 \pm 0,26$	$23,01 \pm 0,35$	$11,27 \pm 0,23$	$21,47 \pm 0,32$
	MaliGAN	$22,53 \pm 0,33$	$15,74 \pm 0,29$	$11,82 \pm 0,26$	$19,35 \pm 0,34$	$23,01 \pm 0,34$	$11,23 \pm 0,25$	$21,44 \pm 0,33$
	GSGAN	$25,28 \pm 2,28$	$17,10 \pm 1,48$	$12,53 \pm 1,14$	$21,53 \pm 1,63$	$26,84 \pm 1,88$	$12,31 \pm 0,85$	$24,84 \pm 1,41$
	RelGAN	$22,19 \pm 0,45$	$15,60 \pm 0,49$	$11,59 \pm 0,48$	$20,38 \pm 0,52$	$27,31 \pm 0,45$	$13,67 \pm 0,60$	$25,89 \pm 0,51$
	TS-GAN-C	$25,00 \pm 0,79$	$17,35 \pm 0,55$	$13,07 \pm 0,44$	$21,43 \pm 0,71$	$25,90 \pm 0,91$	$12,49 \pm 0,45$	$24,05 \pm 0,86$
	TS-GAN-T	$22,26 \pm 0,83$	$15,57 \pm 0,69$	$11,70 \pm 0,61$	$19,19 \pm 0,79$	$22,72 \pm 0,95$	$11,12 \pm 0,63$	$21,18 \pm 0,95$

Tabela 13 – Métricas dos modelos avaliados no conjunto de teste da base de dados Notícias do WMT EMNLP2017.

de triplas. Esse mecanismo aumenta a estabilidade do processo de treinamento (como visto nas curvas do discriminador e do gerador) e evita a variabilidade dos resultados quando o treinamento é repetido com instruções aleatórias fixas por uma semente de aleatoriedade.

8.1.3 Influência da Arquitetura dos Geradores e Discriminadores

A influência das arquiteturas dos geradores e discriminadores é um experimento adicional para verificar possíveis configurações para as Redes Adversariais Generativas por Similaridade Textual. As TS-GAN-C e TS-GAN-T não possuem geradores ou discriminadores padrões, apenas introduzem a noção de similaridade ao processo de discriminação do treinamento adversarial. Portanto, isso possibilita a comparação entre diferentes geradores e discriminadores, auxiliando o entendimento do modelo proposto e verificando lacunas de aprimoramento. É importante ressaltar que as TS-GAN-C e TS-GAN-T apresentadas nas tabelas anteriores foram treinadas com geradores e discriminadores baseados em LSTM (definição empírica).

A Tabela 14 introduz as métricas da TS-GAN-C e TS-GAN-T com diferentes geradores e discriminadores avaliadas no conjunto de teste da base de dados Avaliação de Consumidores da Amazon. Nesta tabela, os melhores resultados de acordo com o teste de Wilcoxon são advindos do gerador baseado em LSTMs. Dentre as três possíveis opções de discriminador (RNN, LSTM e GRU), o discriminador baseado em RNNs obteve os maiores números absolutos para 5 de 6 métricas (exceto por METEOR obtido pelo discriminador GRU). Note que não há discriminador baseado em RMC devido ao alto custo de treinamento do mecanismo de atenção, fato que impossibilita a análise experimental com os recursos disponíveis.

		TS-GAN-C			TS-GAN-T		
Gerador	**Discriminador**	**BLEU-3**	**METEOR**	**ROUGE-L**	**BLEU-3**	**METEOR**	**ROUGE-L**
RNN	RNN	$11,02 \pm 0,54$	$20,09 \pm 0,79$	$24,98 \pm 1,96$	$9,52 \pm 0,94$	$17,84 \pm 2,13$	$21,95 \pm 4,54$
	LSTM	$10,46 \pm 0,54$	$19,45 \pm 0,96$	$24,39 \pm 2,20$	$9,88 \pm 0,70$	$18,68 \pm 1,52$	$23,70 \pm 3,55$
	GRU	$10,82 \pm 0,58$	$19,88 \pm 0,71$	$25,24 \pm 0,98$	$9,99 \pm 0,74$	$18,90 \pm 1,59$	$24,27 \pm 3,27$
LSTM	RNN	$\underline{\mathbf{13,53 \pm 0,43}}$	$\underline{\mathbf{23,76 \pm 0,67}}$	$\underline{\mathbf{26,63 \pm 0,77}}$	$\underline{\mathbf{12,65 \pm 0,43}}$	$\mathbf{22,05 \pm 0,55}$	$\underline{\mathbf{24,22 \pm 0,69}}$
	LSTM	$\mathbf{13,05 \pm 0,27}$	$\mathbf{22,93 \pm 0,27}$	$\mathbf{25,56 \pm 0,40}$	$\mathbf{12,41 \pm 0,19}$	$\mathbf{21,97 \pm 0,31}$	$\mathbf{23,93 \pm 0,36}$
	GRU	$\mathbf{13,38 \pm 0,38}$	$\mathbf{23,48 \pm 0,65}$	$\mathbf{26,20 \pm 0,87}$	$\mathbf{12,53 \pm 0,25}$	$\underline{\mathbf{22,08 \pm 0,35}}$	$\mathbf{24,19 \pm 0,36}$
GRU	RNN	$7,68 \pm 0,46$	$12,40 \pm 1,45$	$13,45 \pm 1,26$	$7,41 \pm 0,14$	$12,12 \pm 0,83$	$12,76 \pm 0,55$
	LSTM	$7,64 \pm 0,33$	$12,50 \pm 1,16$	$13,55 \pm 1,06$	$7,46 \pm 0,16$	$12,38 \pm 0,81$	$12,87 \pm 0,63$
	GRU	$7,69 \pm 0,25$	$12,64 \pm 1,00$	$13,55 \pm 0,90$	$7,58 \pm 0,13$	$12,76 \pm 0,60$	$13,34 \pm 0,50$
RMC	RNN	$11,72 \pm 0,90$	$20,57 \pm 1,63$	$24,23 \pm 2,50$	$9,13 \pm 0,52$	$15,56 \pm 0,81$	$17,10 \pm 1,16$
	LSTM	$11,01 \pm 1,41$	$19,05 \pm 2,36$	$21,84 \pm 3,52$	$9,30 \pm 0,45$	$15,77 \pm 0,67$	$17,41 \pm 0,87$
	GRU	$11,25 \pm 1,07$	$19,65 \pm 1,94$	$22,92 \pm 2,97$	$9,16 \pm 0,35$	$15,59 \pm 0,63$	$17,22 \pm 0,80$

Tabela 14 – Métricas da TS-GAN-C e TS-GAN-T com diferentes geradores e discriminadores avaliadas no conjunto de teste da base de dados Avaliação de Consumidores da Amazon.

A Tabela 15 descreve as métricas da TS-GAN-C e TS-GAN-T com diferentes geradores e discriminadores avaliadas no conjunto de teste da base de dados Avaliações do Yelp. Nesta tabela, existem semelhanças à tabela anterior, onde o gerador LSTM, obteve as melhores métricas de acordo com o teste de Wilcoxon. Dentre os discriminadores usados em conjunto do gerador LSTM, a RNN obteve as melhores métricas para a TS-GAN-C enquanto a LSTM da TS-GAN-T obteve os melhores resultados. A seção anterior salientou que a base de dados Avaliações da Yelp introduziu bastante desvio padrão para os modelos adversariais, repetindo este comportamento para as TS-GANs com geradores baseados em LSTMs. Os outros geradores obtiveram resultados em torno de $6,00$, indicando uma aleatoriedade do treinamento.

Gerador	Discriminador	TS-GAN-C			TS-GAN-T		
		BLEU-3	**METEOR**	**ROUGE-L**	**BLEU-3**	**METEOR**	**ROUGE-L**
RNN	RNN	$6,35 \pm 0,36$	$13,41 \pm 1,37$	$15,17 \pm 1,73$	$6,02 \pm 0,53$	$12,33 \pm 1,68$	$14,20 \pm 2,39$
	LSTM	$6,34 \pm 0,36$	$13,98 \pm 0,78$	$16,47 \pm 1,09$	$5,88 \pm 0,30$	$12,38 \pm 0,74$	$13,33 \pm 1,37$
	GRU	$6,08 \pm 0,36$	$12,64 \pm 0,76$	$13,82 \pm 0,85$	$6,20 \pm 0,45$	$12,84 \pm 0,98$	$14,41 \pm 2,32$
LSTM	RNN	$\underline{\mathbf{24,75 \pm 8,78}}$	$\underline{\mathbf{31,41 \pm 7,72}}$	$\underline{\mathbf{33,48 \pm 7,48}}$	$\mathbf{24,07 \pm 6,17}$	$\mathbf{30,73 \pm 5,44}$	$\mathbf{32,95 \pm 5,14}$
	LSTM	$\mathbf{23,20 \pm 10,82}$	$\mathbf{29,83 \pm 9,84}$	$\mathbf{31,97 \pm 9,52}$	$\underline{\mathbf{25,37 \pm 9,88}}$	$\underline{\mathbf{31,66 \pm 9,22}}$	$\underline{\mathbf{33,90 \pm 8,38}}$
	GRU	$\mathbf{24,04 \pm 9,88}$	$\mathbf{30,84 \pm 8,46}$	$\mathbf{33,19 \pm 7,87}$	$\mathbf{21,28 \pm 10,00}$	$\mathbf{28,31 \pm 8,76}$	$\mathbf{30,58 \pm 8,45}$
GRU	RNN	$6,48 \pm 1,90$	$12,78 \pm 3,06$	$14,21 \pm 3,60$	$6,10 \pm 1,03$	$12,46 \pm 2,27$	$13,59 \pm 2,71$
	LSTM	$6,10 \pm 0,66$	$12,43 \pm 1,79$	$14,21 \pm 2,42$	$6,04 \pm 1,14$	$11,88 \pm 2,62$	$12,62 \pm 2,71$
	GRU	$5,89 \pm 0,68$	$11,79 \pm 2,36$	$12,94 \pm 2,40$	$5,82 \pm 0,77$	$11,38 \pm 1,85$	$12,28 \pm 1,90$
RMC	RNN	$6,63 \pm 0,41$	$14,60 \pm 0,95$	$17,95 \pm 1,75$	$6,45 \pm 0,33$	$13,87 \pm 0,60$	$16,59 \pm 0,70$
	LSTM	$6,65 \pm 0,32$	$14,84 \pm 0,74$	$18,55 \pm 1,81$	$6,56 \pm 0,26$	$14,28 \pm 0,53$	$16,83 \pm 0,83$
	GRU	$6,49 \pm 0,59$	$14,05 \pm 1,30$	$16,87 \pm 1,60$	$6,51 \pm 0,35$	$14,03 \pm 0,70$	$16,85 \pm 0,90$

Tabela 15 – Métricas da TS-GAN-C e TS-GAN-T com diferentes geradores e discriminadores avaliadas no conjunto de teste da base de dados Avaliações do Yelp.

A Tabela 16 apresenta as métricas da TS-GAN-C e TS-GAN-T com diferentes geradores e discriminadores avaliadas no conjunto de teste da base de dados Legendas de Imagens COCO. Embora as redes com geradores baseados em RNN, GRU e RMC tenha conseguido inicializar corretamente o treinamento e atingir resultados satisfatórios, todas as redes com gerador baseado em LSTM obtiveram os melhores resultados (valores em ênfase) de acordo com o teste de Wilcoxon. Nesta base de dados, os valores absolutos são maiores comparados às tabelas anteriores devido a facilidade da tarefa da Legendas de Imagens COCO. O valor absoluto obtido pela métrica BLEU-3 da melhor arquitetura TS-GAN-C (gerador LSTM e discriminador RNN) é $12,96\%$ superior ao da apresentada na Tabela 11, enquanto o da TS-GAN-T (gerador LSTM e discriminador LSTM) é $24,73\%$ superior. Além da BLEU-3, todas as outras métricas apresentam valores absolutos maiores quando comparadas com a Tabela 11.

Gerador	Discriminador	TS-GAN-C			TS-GAN-T		
		BLEU-3	**METEOR**	**ROUGE-L**	**BLEU-3**	**METEOR**	**ROUGE-L**
RNN	RNN	$33,04 \pm 5,95$	$42,82 \pm 5,22$	$51,30 \pm 4,15$	$33,59 \pm 5,52$	$43,34 \pm 5,53$	$51,67 \pm 3,74$
	LSTM	$31,02 \pm 8,31$	$39,45 \pm 11,02$	$46,84 \pm 11,30$	$34,81 \pm 5,19$	$44,53 \pm 4,72$	$52,01 \pm 3,83$
	GRU	$36,26 \pm 3,88$	$45,54 \pm 4,02$	$53,43 \pm 2,74$	$33,04 \pm 5,84$	$42,44 \pm 5,89$	$50,76 \pm 4,60$
LSTM	RNN	$\mathbf{41,98 \pm 4,67}$	$\mathbf{50,63 \pm 4,31}$	$\mathbf{57,92 \pm 3,63}$	$\mathbf{43,34 \pm 2,05}$	$\mathbf{52,36 \pm 2,13}$	$\mathbf{58,86 \pm 1,67}$
	LSTM	$\mathbf{40,54 \pm 5,03}$	$\mathbf{49,46 \pm 4,45}$	$\mathbf{56,55 \pm 3,82}$	$\mathbf{43,60 \pm 2,30}$	$\mathbf{52,03 \pm 2,25}$	$\mathbf{58,82 \pm 1,73}$
	GRU	$\underline{\mathbf{43,16 \pm 1,89}}$	$\underline{\mathbf{51,70 \pm 1,94}}$	$\underline{\mathbf{58,64 \pm 1,70}}$	$\underline{\mathbf{44,05 \pm 1,73}}$	$\underline{\mathbf{52,46 \pm 2,11}}$	$\underline{\mathbf{59,35 \pm 1,50}}$
GRU	RNN	$26,27 \pm 2,24$	$34,77 \pm 2,99$	$42,22 \pm 2,79$	$26,10 \pm 2,36$	$34,62 \pm 3,03$	$42,28 \pm 3,08$
	LSTM	$26,21 \pm 1,00$	$34,45 \pm 1,55$	$41,94 \pm 1,76$	$26,10 \pm 2,23$	$34,71 \pm 2,95$	$42,23 \pm 2,51$
	GRU	$24,94 \pm 1,48$	$33,34 \pm 1,90$	$41,00 \pm 2,10$	$26,01 \pm 1,89$	$34,23 \pm 2,57$	$41,49 \pm 2,38$
RMC	RNN	$28,35 \pm 1,67$	$38,87 \pm 2,13$	$48,42 \pm 1,60$	$29,18 \pm 2,05$	$38,99 \pm 2,54$	$47,86 \pm 2,04$
	LSTM	$28,98 \pm 1,76$	$39,03 \pm 1,78$	$48,33 \pm 1,56$	$28,56 \pm 2,69$	$38,36 \pm 2,82$	$48,07 \pm 1,78$
	GRU	$28,50 \pm 2,30$	$38,42 \pm 2,62$	$48,15 \pm 2,06$	$29,30 \pm 2,10$	$39,15 \pm 2,55$	$48,81 \pm 1,43$

Tabela 16 – Métricas da TS-GAN-C e TS-GAN-T com diferentes geradores e discriminadores avaliadas no conjunto de teste da base de dados Legendas de Imagens COCO.

Finalmente, a Tabela 17 descreve as métricas da TS-GAN-C e TS-GAN-T com diferentes geradores e discriminadores avaliadas no conjunto de teste da base de dados Notícias do WMT EMNLP2017. Note que esta base de dados segue o mesmo comportamento apresentado pela Tabela 14, onde as redes com geradores baseados em LSTM atingiram as melhores métricas, exceto pela ROUGE-L com a TS-GAN-T. O gerador RNN obteve os melhores resultados de acordo com o teste de Wilcoxon para as métricas ROUGE-L (ambas TS-GAN-C e TS-GAN-T). Entretanto, apesar do gerador baseado em RNN ter as melhores ROUGE-L, todas as outras métricas (BLEU-3 e METEOR) foram consideravelmente inferiores (por volta de 25%) ao gerador baseado em LSTM. Tal fenômeno pode ser observado diretamente na qualidade dos textos gerados por ambas arquiteturas, onde o texto gerado pela RNN é mais diverso (menos repetições) porém um pouco menos compreensível do que o texto gerado pela LSTM (Apêndice A).

Gerador	Discriminador	TS-GAN-C			TS-GAN-T		
		BLEU-3	METEOR	ROUGE-L	BLEU-3	METEOR	ROUGE-L
RNN	RNN	$10,51 \pm 0,55$	$19,12 \pm 0,38$	$\underline{\mathbf{24,75 \pm 0,84}}$	$9,87 \pm 0,23$	$18,69 \pm 0,30$	$\underline{\mathbf{24,71 \pm 0,92}}$
	LSTM	$10,35 \pm 0,96$	$18,71 \pm 1,52$	$\mathbf{23,92 \pm 2,75}$	$9,87 \pm 0,27$	$18,56 \pm 0,64$	$\mathbf{24,34 \pm 1,40}$
	GRU	$10,53 \pm 0,81$	$18,73 \pm 1,54$	$\mathbf{23,63 \pm 3,08}$	$9,58 \pm 0,59$	$17,92 \pm 1,70$	$23,27 \pm 3,37$
LSTM	RNN	$\mathbf{13,02 \pm 0,57}$	$\mathbf{21,48 \pm 0,79}$	$\mathbf{24,24 \pm 1,07}$	$\mathbf{11,72 \pm 0,28}$	$\mathbf{19,21 \pm 0,32}$	$21,32 \pm 0,36$
	LSTM	$\mathbf{12,88 \pm 0,68}$	$\mathbf{21,11 \pm 1,02}$	$\mathbf{23,65 \pm 1,29}$	$\mathbf{11,75 \pm 0,26}$	$\mathbf{19,28 \pm 0,40}$	$21,38 \pm 0,47$
	GRU	$\underline{\mathbf{13,30 \pm 0,53}}$	$\underline{\mathbf{21,77 \pm 0,75}}$	$\mathbf{24,57 \pm 1,02}$	$\underline{\mathbf{11,82 \pm 0,25}}$	$\underline{\mathbf{19,32 \pm 0,36}}$	$21,48 \pm 0,29$
GRU	RNN	$7,38 \pm 0,14$	$11,37 \pm 0,42$	$12,10 \pm 0,46$	$7,38 \pm 0,07$	$11,53 \pm 0,29$	$12,20 \pm 0,26$
	LSTM	$7,39 \pm 0,14$	$11,49 \pm 0,51$	$12,21 \pm 0,45$	$7,46 \pm 0,16$	$11,85 \pm 0,55$	$12,52 \pm 0,53$
	GRU	$7,37 \pm 0,18$	$11,28 \pm 0,80$	$12,08 \pm 0,51$	$7,44 \pm 0,09$	$11,77 \pm 0,34$	$12,41 \pm 0,32$
RMC	RNN	$9,72 \pm 1,68$	$15,76 \pm 3,26$	$18,28 \pm 4,29$	$8,41 \pm 0,36$	$13,24 \pm 0,55$	$14,48 \pm 0,79$
	LSTM	$8,98 \pm 1,30$	$14,34 \pm 2,50$	$16,25 \pm 3,09$	$8,18 \pm 0,26$	$12,97 \pm 0,33$	$14,13 \pm 0,56$
	GRU	$9,99 \pm 1,16$	$16,41 \pm 2,19$	$19,03 \pm 3,27$	$8,21 \pm 0,22$	$12,98 \pm 0,37$	$14,16 \pm 0,48$

Tabela 17 – Métricas da TS-GAN-C e TS-GAN-T com diferentes geradores e discriminadores avaliadas no conjunto de teste da base de dados Notícias do WMT EMNLP2017.

O experimento adicional sobre a influência das arquiteturas dos geradores e discriminadores em Redes Adversariais Generativas por Similaridade Textual (TS-GANs) permite comparar diferentes geradores e discriminadores e entender melhor o modelo proposto, identificando possíveis pontos de melhoria. É importante notar que as TS-GANs apresentadas nas tabelas anteriores foram treinadas com geradores e discriminadores baseados em LSTM (definição empírica).

As Tabelas 14, 15 e 16 mostram que os melhores resultados de acordo com o teste de Wilcoxon são obtidos pelos geradores baseados em LSTM. Entre os discriminadores (RNN, LSTM e GRU), o discriminador baseado em RNNs obteve os maiores valores absolutos para cinco das seis métricas avaliadas (exceto METEOR obtido pelo discriminador GRU). É importante lembrar que não há discriminador baseado em RMC devido ao alto custo de treinamento do mecanismo de atenção. As Tabelas 14 e 15 também mostram que o desvio padrão dos modelos adversariais é mais alto na base de dados Avaliações do Yelp, repetindo esse comportamento nas TS-GANs com geradores baseados em LSTM. Os outros geradores obtiveram resultados em torno de 6,0, indicando aleatoriedade no treinamento. A Tabela 16 mostra que, embora as redes com geradores baseados em RNN, GRU e RMC tenham conseguido inicializar corretamente o treinamento e atingir resultados satisfatórios, todas as redes com gerador baseado em LSTM obtiveram os melhores resultados de acordo com o teste de Wilcoxon. Nesta base de dados, os valores absolutos são maiores em comparação com as tabelas anteriores, indicando que as TS-GANs podem aprender de forma mais eficiente com maiores quantidades de dados.

8.1.4 Comparação entre Funções de Perda

Esta seção analisa o desempenho da TS-GAN utilizando diferentes funções de perda e suas respectivas medidas de distância. Tal comparação é efetuada na tarefa de geração de texto a partir das três métricas de avaliação mais consistentes, isto é, BLEU-3, METEOR e ROUGE-L.

A Tabela 18 descreve as métricas da TS-GAN (gerador LSTM e discriminador RNN) com diferentes funções de perda e distância avaliadas no conjunto de teste da base de dados Avaliação de Consumidores da Amazon. A função de perda Contrastiva (TS-GAN-C) em conjunto da função de distância Euclidiana obteve os melhores resultados dentre as comparações realizadas e reforça os resultados obtidos na Tabela 10. Por outro lado, quaisquer redes com funções de perda Tripla (difícil e semi-difícil) não atingiram a efetividade da função de perda Contrastiva, especialmente considerando a distância Euclidiana ($8,7\%$ inferior que a tripla difícil e $10,96\%$ inferior que a tripla semi-difícil).

Função de Perda	**Função de Distância**	**BLEU-3**	**METEOR**	**ROUGE-L**
Entropia Cruzada	Concatenação	$12,44 \pm 0,32$	$21,98 \pm 0,35$	$24,10 \pm 0,35$
	Diferença	$12,56 \pm 0,27$	$22,17 \pm 0,42$	$24,34 \pm 0,40$
Contrastiva	Manhattan	$8,13 \pm 2,30$	$11,74 \pm 5,42$	$13,86 \pm 5,53$
	Euclidiana	$\mathbf{13,74 \pm 0,33}$	$\mathbf{24,09 \pm 0,46}$	$\mathbf{27,04 \pm 0,54}$
	Angular	$12,63 \pm 0,23$	$22,18 \pm 0,23$	$24,43 \pm 0,29$
Tripla (Difícil)	Manhattan	$12,53 \pm 0,21$	$21,79 \pm 0,25$	$23,76 \pm 0,35$
	Euclidiana	$12,45 \pm 0,14$	$21,76 \pm 0,22$	$23,74 \pm 0,23$
	Angular	$12,65 \pm 0,27$	$22,16 \pm 0,43$	$24,22 \pm 0,47$
Tripla (Semi-Difícil)	Manhattan	$12,50 \pm 0,20$	$21,77 \pm 0,25$	$23,82 \pm 0,29$
	Euclidiana	$12,35 \pm 0,35$	$21,71 \pm 0,40$	$23,57 \pm 0,53$
	Angular	$12,65 \pm 0,37$	$22,09 \pm 0,59$	$24,08 \pm 0,63$

Tabela 18 – Métricas da TS-GAN (gerador LSTM e discriminador RNN) com diferentes funções de perda e distâncias avaliadas no conjunto de teste da base de dados Avaliação de Consumidores da Amazon.

A incapacidade da distância de Manhattan em obter métricas plausíveis com a função de perda Contrastiva é justificada pela simplicidade matemática do cálculo entre as amostras positivas e negativas[37]. Todavia, a distância de Manhattan com funções de perda Tripla obteve

[37] O espaço da distância de Manhattan é simples comparado ao espaço euclidiano e incapaz de representar distribuições mais complexas

resultados superiores em relação à perda Contrastiva devido a utilização de triplas, as quais aumentam o espaço de valores que a função de perda consegue codificar.

A Tabela 19 apresenta as métricas da TS-GAN (gerador LSTM e discriminador RNN) com diferentes funções de perda e distâncias avaliadas no conjunto de teste da base de dados Avaliações do Yelp. A função de perda Contrastiva (exceto pela distância Angular na métrica BLEU-3) e Entropia Cruzada obtiveram os melhores resultados de acordo com o teste de Wilcoxon. Esta última função é a mesma responsável pelo pré-treinamento dos geradores nos modelos adversariais e beneficia-se em tarefas nas quais a convergência é estável e prolongada.

Função de Perda	**Função de Distância**	**BLEU-3**	**METEOR**	**ROUGE-L**
Entropia Cruzada	Concatenação	$\mathbf{24,26 \pm 9,71}$	$\mathbf{30,97 \pm 8,40}$	$\mathbf{33,22 \pm 7,87}$
	Diferença	$\mathbf{22,64 \pm 9,37}$	$\mathbf{29,53 \pm 8,26}$	$\mathbf{31,78 \pm 7,82}$
Contrastiva	Manhattan	$\mathbf{21,50 \pm 10,06}$	$\mathbf{27,77 \pm 10,27}$	$\mathbf{29,70 \pm 10,38}$
	Euclidiana	$\underline{\mathbf{27,38 \pm 6,87}}$	$\underline{\mathbf{33,72 \pm 6,03}}$	$\underline{\mathbf{35,70 \pm 5,70}}$
	Angular	$19,19 \pm 8,78$	$\mathbf{26,65 \pm 7,57}$	$\mathbf{28,97 \pm 7,08}$
Tripla (Difícil)	Manhattan	$13,56 \pm 5,48$	$21,40 \pm 4,98$	$23,95 \pm 4,83$
	Euclidiana	$11,24 \pm 3,19$	$19,66 \pm 2,92$	$22,35 \pm 2,91$
	Angular	$16,10 \pm 7,30$	$23,73 \pm 6,50$	$26,08 \pm 6,23$
Tripla (Semi-Difícil)	Manhattan	$17,17 \pm 7,87$	$24,88 \pm 6,82$	$\mathbf{27,21 \pm 6,52}$
	Euclidiana	$11,40 \pm 3,27$	$19,36 \pm 3,60$	$21,82 \pm 4,09$
	Angular	$14,05 \pm 7,07$	$22,16 \pm 5,99$	$24,71 \pm 5,83$

Tabela 19 – Métricas da TS-GAN (gerador LSTM e discriminador RNN) com diferentes funções de perda e distâncias avaliadas no conjunto de teste da base de dados Avaliações do Yelp.

Lembrando da Figura 23 (a), o gerador LSTM está em processo de convergência com um valor adequado de perda (por volta de $2,60$ NLL) e com indícios de manter o aprendizado. Portanto, a Entropia Cruzada é capaz de atingir bons valores de métricas pós-treinamento adversarial. Note que ao compararmos as Tabelas 19 e 11, o experimento de análise de função de perda manteve o acréscimo obtido no experimento anterior (descoberta de gerador LSTM e discriminador RNN) e obteve $24,97\%$ mais BLEU-3 que o experimento inicial (Tabela 11) para a TS-GAN-C e $18,46\%$ menos BLEU-3 para a TS-GAN-T.

A Tabela 20 apresenta as métricas da TS-GAN (gerador LSTM e discriminador GRU) com diferentes funções de perda e distância avaliadas no conjunto de teste da base de dados Legendas de Imagens COCO. Nesta tabela, a função de perda Contrastiva obteve os melhores

resultados de acordo com o teste de Wilcoxon, exceto pela distância Angular. Por outro lado, quase todas as funções de perda Tripla conseguiram usufruir de uma tarefa mais fácil e fazer parte dos resultados estatisticamente iguais por Wilcoxon. Embora a Tripla difícil com distância de Manhattan e a Tripla semi-difícil com distância Euclidiana não tenham obtido os melhores resultados, elas obtiveram resultados proporcionalmente melhores em relação à base de dados anterior.

Função de Perda	**Função de Distância**	**BLEU-3**	**METEOR**	**ROUGE-L**
Entropia Cruzada	Concatenação	$42,15 \pm 1,48$	$50,98 \pm 1,48$	$57,90 \pm 1,10$
	Diferença	$40,03 \pm 1,16$	$48,71 \pm 0,81$	$56,27 \pm 0,93$
Contrastiva	Manhattan	$\mathbf{43,65 \pm 2,39}$	$\mathbf{52,06 \pm 2,46}$	$\mathbf{58,73 \pm 1,89}$
	Euclidiana	$\mathbf{43,32 \pm 2,43}$	$\mathbf{52,09 \pm 2,58}$	$\mathbf{58,86 \pm 1,98}$
	Angular	$41,34 \pm 3,54$	$50,42 \pm 3,27$	$57,41 \pm 2,80$
Tripla (Difícil)	Manhattan	$38,09 \pm 0,00$	$47,51 \pm 0,00$	$54,48 \pm 0,00$
	Euclidiana	$\mathbf{43,56 \pm 2,19}$	$\mathbf{52,15 \pm 2,29}$	$\mathbf{58,87 \pm 1,62}$
	Angular	$\mathbf{42,92 \pm 2,91}$	$\mathbf{51,80 \pm 3,00}$	$\mathbf{58,53 \pm 2,24}$
Tripla (Semi-Difícil)	Manhattan	$\mathbf{42,92 \pm 2,91}$	$\mathbf{51,80 \pm 3,00}$	$\mathbf{58,53 \pm 2,24}$
	Euclidiana	$40,97 \pm 4,65$	$49,86 \pm 4,30$	$56,64 \pm 3,98$
	Angular	$\mathbf{43,49 \pm 2,43}$	$\mathbf{52,30 \pm 2,30}$	$\mathbf{58,78 \pm 1,93}$

Tabela 20 – Métricas da TS-GAN (gerador LSTM e discriminador GRU) com diferentes funções de perda e distâncias avaliadas no conjunto de teste da base de dados Legendas de Imagens COCO.

A Tabela 21 descreve as métricas da TS-GAN (gerador LSTM e discriminador GRU) com diferentes funções de perda e distância avaliadas no conjunto de teste da base de dados Notícias do WMT EMNLP2017. Novamente, a função de perda Contrastiva com distância Euclidiana obteve os melhores resultados de acordo com o teste de Wilcoxon. Similar à Tabela 18, as funções de perda Contrastiva e Tripla obtiveram resultados próximos com desvios padrões baixos quando comparadas à Tabela 19. As métricas finais das bases de dados Avaliações de Consumidores da Amazon e Notícias do WMT EMNLP2017 (problemas de nível médio) foram inferiores as da Avaliações do Yelp (problema difícil), demonstrando que tarefas de nível médio beneficiar-se-ão de arquiteturas mais robustas (maior quantidade de neurônios de vetorização e ocultos).

Função de Perda	Função de Distância	BLEU-3	METEOR	ROUGE-L
Entropia Cruzada	Concatenação	$8,81 \pm 2,23$	$13,28 \pm 4,76$	$15,18 \pm 4,97$
	Diferença	$8,42 \pm 2,19$	$12,37 \pm 4,57$	$14,23 \pm 4,75$
Contrastiva	Manhattan	$11,79 \pm 0,24$	$19,38 \pm 0,32$	$21,48 \pm 0,35$
	Euclidiana	$\underline{\mathbf{12,96 \pm 0,89}}$	$\underline{\mathbf{21,29 \pm 1,28}}$	$\underline{\mathbf{23,94 \pm 1,60}}$
	Angular	$11,94 \pm 0,20$	$19,55 \pm 0,24$	$21,71 \pm 0,21$
Tripla (Difícil)	Manhattan	$11,34 \pm 0,20$	$18,63 \pm 0,22$	$20,54 \pm 0,24$
	Euclidiana	$11,38 \pm 0,14$	$18,67 \pm 0,17$	$20,60 \pm 0,15$
	Angular	$11,77 \pm 0,36$	$19,19 \pm 0,55$	$21,19 \pm 0,66$
Tripla (Semi-Difícil)	Manhattan	$11,30 \pm 0,23$	$18,54 \pm 0,31$	$20,48 \pm 0,35$
	Euclidiana	$11,40 \pm 0,28$	$18,69 \pm 0,35$	$20,60 \pm 0,43$
	Angular	$11,43 \pm 0,21$	$18,71 \pm 0,27$	$20,63 \pm 0,38$

Tabela 21 – Métricas da TS-GAN (gerador LSTM e discriminador GRU) com diferentes funções de perda e distâncias avaliadas no conjunto de teste da base de dados Notícias do WMT EMNLP2017.

Por fim, ao compararmos todos os resultados obtidos na Frente I, as metodologias de busca propostas (busca de gerador e discriminador, busca de função de perda e função de distância) aprimoraram os experimentos iniciais descritos nas Tabelas 10 e 12, e atingiram resultados comparáveis ao da Tabela 13. Note que no experimento de análise de diferentes funções de perda, todas as TS-GAN-Ts foram inferiores as TS-GAN-Cs e, por isso, não estão totalmente presentes (pontos verdes) nas figuras a seguir.

De acordo com os resultados, a função de perda Contrastiva em conjunto com a distância Euclidiana obteve os melhores resultados no conjunto de teste da base de dados Avaliação de Consumidores da Amazon. Além disso, a função de perda Contrastiva e Entropia Cruzada obtiveram os melhores resultados no conjunto de teste da base de dados Avaliações do Yelp. No geral, a função de perda Contrastiva foi capaz de obter resultados superiores às funções de perda Tripla (difícil e semi-difícil).

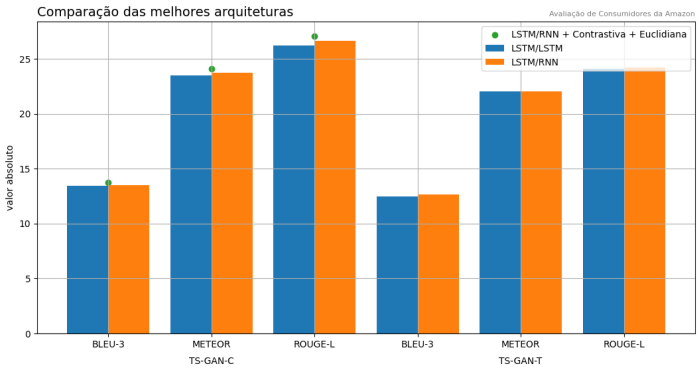
De maneira geral, a distância Euclidiana se mostrou mais eficiente que a distância de Manhattan no caso da função de perda Contrastiva. Isso pode ser justificado pela simplicidade matemática da distância de Manhattan em comparação à distância Euclidiana, o que a torna inadequada para representar distribuições mais complexas. Entretanto, a distância de Manhattan se mostrou superior à distância Euclidiana quando utilizada com funções de perda Tripla, pois essas funções permitem codificar um espaço de valores mais amplo.

Em resumo, os resultados mostram que a função de perda Contrastiva em conjunto

com a distância Euclidiana é a opção mais eficiente para o modelo TS-GAN quando utilizado na tarefa de geração de de texto. Além disso, a função de perda Contrastiva e a distância Euclidiana foram capazes de aumentar o desempenho em relação a outras funções de perda e medidas de distância.

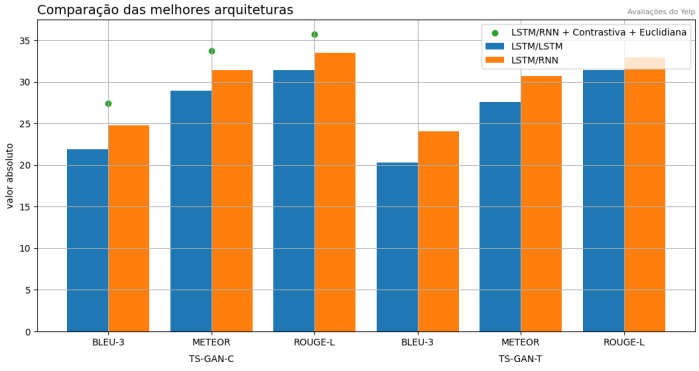
A Figura 26 ilustra a comparação da efetividade do processo de busca e modelagem da Rede Adversarial Generativa por Similaridade Textual na base de dados Avaliação dos Consumidores da Amazon. Nesta figura, as TS-GANs com os melhores geradores/discriminadores e com as melhores funções de perda/funções de distância superaram em torno de 10% as arquiteturas utilizadas nos dois primeiros experimentos. Por outro lado, na Figura 27, a efetividade das melhores arquiteturas apenas superaram os experimentos anteriores por 2%.

Figura 26 – Análise comparativa da efetividade do processo de busca e modelagem da Rede Adversarial Generativa por Similaridade Textual na base de dados Avaliação dos Consumidores da Amazon.



Fonte: Elaborado pelo autor.

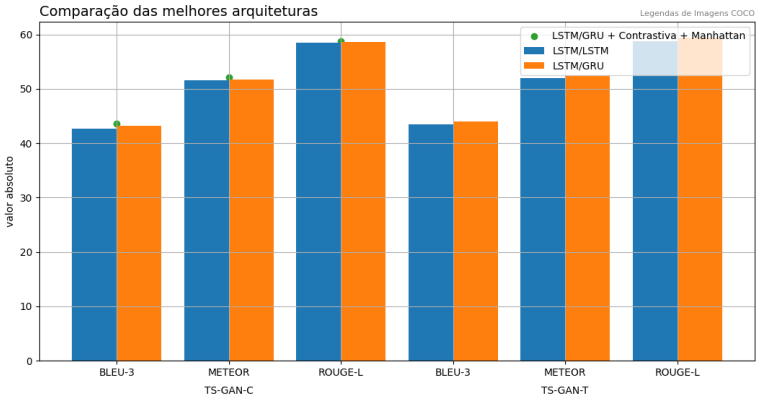
Figura 27 – Análise comparativa da efetividade do processo de busca e modelagem da Rede Adversarial Generativa por Similaridade Textual na base de dados Avaliações do Yelp.



Fonte: Elaborado pelo autor.

A Figura 28 apresenta a comparação da efetividade do processo de busca e modelagem da Rede Adversarial Generativa por Similaridade Textual na base de dados Legendas de Imagens COCO. Nesta base de dados, similar à anterior, as melhores arquiteturas foram capazes de incrementar a efetividade das arquiteturas anteriores (em torno de 3%).

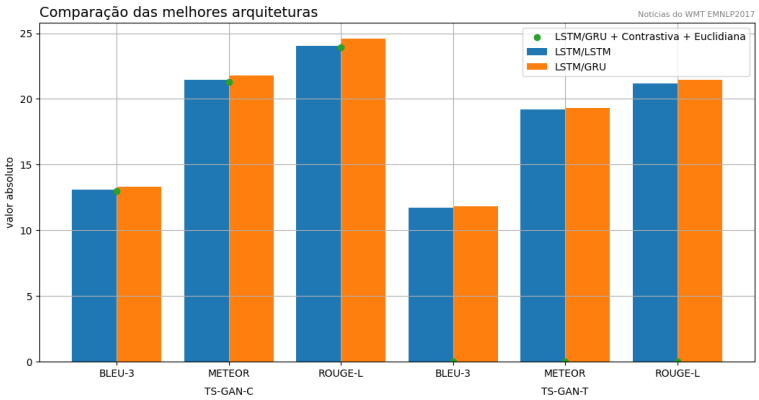
Figura 28 – Análise comparativa da efetividade do processo de busca e modelagem da Rede Adversarial Generativa por Similaridade Textual na base de dados Legendas de Imagens COCO.



Fonte: Elaborado pelo autor.

Finalmente, a Figura 29 ilustra a comparação da efetividade do processo de busca e modelagem da Rede Adversarial Generativa por Similaridade Textual na base de dados Notícias do WMT EMNLP2017. Nesta base de dados, apenas o experimento de busca de geradores e discriminadores foi capaz de aumentar a efetividade da arquitetura inicial nas métricas analisadas.

Figura 29 – Análise comparativa da efetividade do processo de busca e modelagem da Rede Adversarial Generativa por Similaridade Textual na base de dados Notícias do WMT EMNLP2017.



Fonte: Elaborado pelo autor.

8.2 Frente II

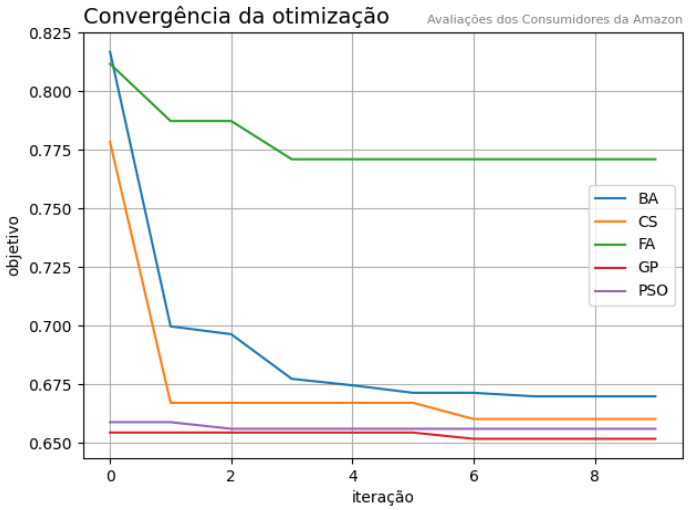
A Frente II propõe a otimização otimização meta-heurística dos hiperparâmetros da Rede Adversarial Generativa por Similaridade Textual. Os experimentos estão divididos em três análises: convergência da otimização (análise da função objetivo), influência dos hiperparâmetros otimizados (análise de métricas) e análise do custo computacional (tempo de otimização + tempo de treinamento).

8.2.1 Convergência da Otimização

O processo de otimização objetiva encontrar o conjunto mais apto de hiperparâmetros e o menor valor possível para a função objetivo[38].

A Figura 30 ilustra a convergência da função objetivo na base de dados Avaliação dos Consumidores da Amazon. Esta figura demonstra que todas as meta-heurísticas analisadas, exceto pela FA, atingiram valores de função objetivo menores que $0,675$. Ademais, é possível observar que apenas BA e CS convergiram ao longo das iterações de otimização, enquanto GP e PSO melhoraram uma única vez o valor da função objetivo pós-inicialização. Considerando este último caso (GP e PSO), as meta-heurísticas obtiveram um valor adequado pós-inicialização e não foram capazes de encontrar valores mais aptos durante o seu processo de busca. Adicionalmente, suas configurações de parâmetros incentivam a exploração do espaço de busca (encontrar valores iniciais) ao invés da intensificação (melhorar valores aptos já encontrados).

Figura 30 – Convergência da função objetivo de otimização na base de dados Avaliações de Consumidores da Amazon.

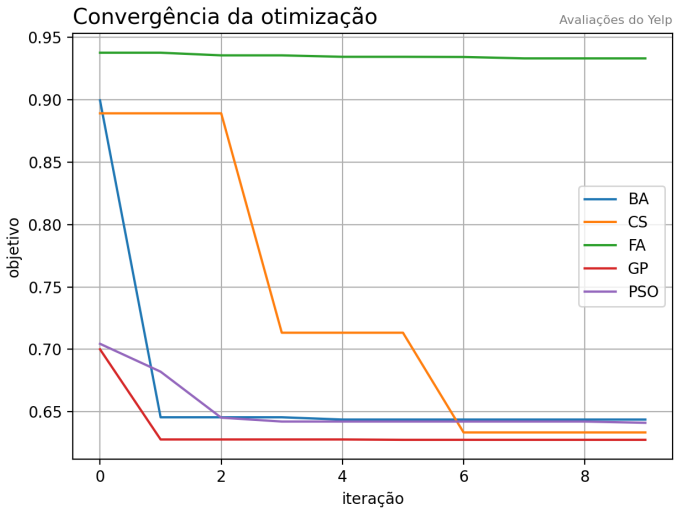


Fonte: Elaborado pelo autor.

[38] Esta tese considera a otimização como um problema de minimização e a função objetivo como a métrica $1 -$ BLEU-3 obtida a partir do conjunto de validação.

A Figura 31 ilustra a convergência da função objetivo na base de dados Avaliações do Yelp. Nesta base de dados, é possível verificar um comportamento semelhante à base de dados anterior, onde a meta-heurística FA não obteve convergência durante o processo de otimização, enquanto todas as outras técnicas convergiram para um ponto em comum (abaixo de $0,65$). Não obstante, as técnicas GP e PSO obtiveram as melhores inicializações aleatórias, enquanto BA e CS conquistaram os maiores decaimentos da função objetivo durante as 10 iterações da otimização.

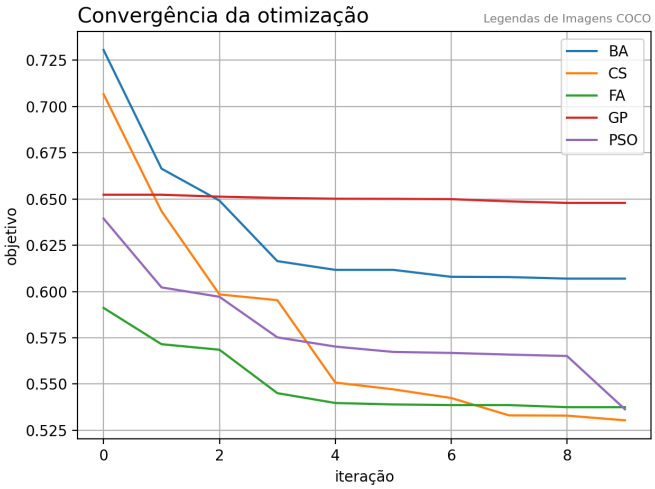
Figura 31 – Convergência da função objetivo de otimização na base de dados Avaliações do Yelp.



Fonte: Elaborado pelo autor.

A base de dados Legendas de Imagens COCO constitui o problema mais fácil dentre os quatro analisados e, consequentemente, é menos dependente de uma configuração de hiperparâmetros específica. Desta forma, o espaço de busca é simples e provável de ser bem explorado pelas meta-heurísticas. A Figura 32 ilustra a convergência da função objetivo na base de dados Legendas de Imagens COCO. Nesta figura, podemos observar o comportamento mencionado no parágrafo anterior, onde todas as meta-heurísticas, exceto pela GP, foram capazes de convergir ao longo das iterações de otimização. Surpreendentemente, considerando o espaço de busca simples, a FA obteve um bom processo de convergência e atingiu um resultado final tão bom quanto a PSO, apenas superados pela CS.

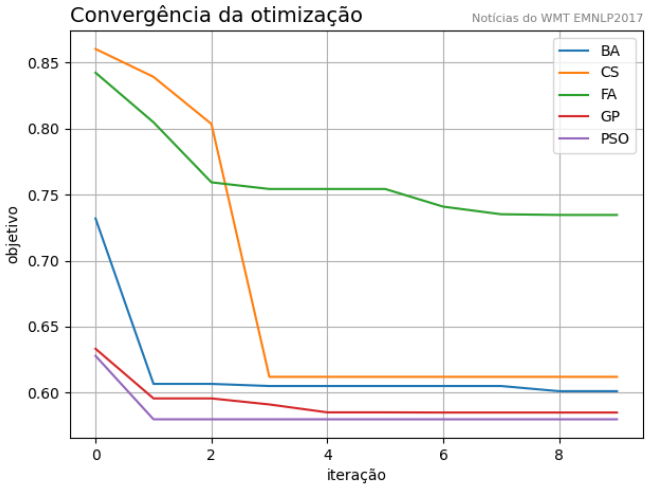
Figura 32 – Convergência da função objetivo de otimização na base de dados Legendas de Imagens COCO.



Fonte: Elaborado pelo autor.

Finalmente, a Figura 33 ilustra a convergência da função objetivo na base de dados Notícias do WMT EMNLP2017. Nesta figura, é possível observar um comportamento semelhante às duas primeiras bases de dados, onde GP e PSO tiveram as melhores inicializações e mantiver-se-am estáveis ao longo da otimização, localizados nos melhores pontos de mínimo quando comparadas às outras técnicas avaliadas. Ademais, a meta-heurística CS obteve a melhor convergência dentre todas as meta-heurísticas, principalmente devido à sua capacidade de exploração, onde à cada iteração, uma parcela de agentes são descartados e inicializados aleatoriamente, o qual proporciona um caráter de diversidade à busca.

Figura 33 – Convergência da função objetivo de otimização na base de dados Notícias do WMT EMNLP2017.



Fonte: Elaborado pelo autor.

A análise comparativa mostra que as meta-heurísticas BA e CS foram as que melhor convergiram para o menor valor possível da função objetivo em todas as bases de dados analisadas. As meta-heurísticas GP e PSO apresentaram um bom desempenho inicial, mas não foram capazes de encontrar valores mais aptos durante o processo de otimização. A meta-heurística FA não atingiu convergência em nenhuma das bases de dados. Isso sugere que BA e CS são as melhores opções para o processo de otimização em busca do conjunto mais apto de hiperparâmetros.

8.2.2 Influência dos Hiperparâmetros Otimizados

A análise da influência dos hiperparâmetros otimizados objetiva quantificar o processo de otimização através das métricas utilizadas na Frente I. Neste experimento, as TS-GANs foram treinadas utilizado o protocolo definido pela Frente I, exceto pelos hiperparâmetros (η_G, η_D, n_e, n_h e Δ) que foram selecionados de acordo com o experimento anterior.

A Tabela 22 descreve os hiperparâmetros encontrados durante a otimização da TS-GAN (gerador LSTM, discriminador RNN, função de perda Contrastiva e função de distância Euclidiana) na base de dados Avaliações de Consumidores da Amazon. Note que a péssima convergência da meta-heurística FA está diretamente relacionada à escolha dos hiperparâmetros, os quais são o intervalo inferior da Tabela 9. Por outro lado, a capacidade das técnicas BA, CS, GP e PSO em encontrar melhores valores para o número de neurônios de vetorização e escondidos auxiliaram a convergência da rede e, consequentemente, atingiram menores valores para a função objetivo (1 - BLEU-3 avaliada no conjunto de validação).

Adicionalmente, as taxas de aprendizado para o gerador mantiver-se-am em $0,00001$ para todas as meta-heurísticas, denotando que este valor é plausível para o problema em questão. As melhores convergências (GP e PSO) estão correlacionadas às técnicas que não encontraram valores muito distintos para a quantidade de neurônios (vetorização e escondidos), reforçando a ideia de que não é aconselhável efetuar mapeamentos que alterem fortemente o tamanho do espaço dimensional (possibilidade de perda de informação).

A Tabela 23 descreve as métricas pós-otimização da TS-GAN (gerador LSTM, discriminador RNN, função de perda Contrastiva e função de distância Euclidiana) avaliada no conjunto de teste da base de dados Avaliação de Consumidores da Amazon. Ao compararmos as Tabelas 18 e 23, e considerarmos a meta-heurística CS, é possível observar que o processo de otimização trouxe mais instabilidade às métricas (maior desvio padrão), embora tenha aprimorado o valor absoluto para a métrica METEOR ($27,32$ x $24,09$). Adicionalmente, GP atingiu o maior valor absoluto para a métrica BLEU-3, o que era esperado devido à melhor convergência no conjunto de validação (Figura 30). Este resultado, embora seja menor em termos absolutos à TS-GAN sem otimização ($13,13$ x $13,74$), é estatisticamente semelhante de acordo com o teste de Wilcoxon e demonstra que o processo de otimização foi capaz de gerar um novo conjunto de hiperparâmetros tão bom quantos os iniciais.

Modelo	Otimizador	η_G	η_D	n_e	n_h	Δ
TS-GAN	BA	$0,00001$	$0,00100$	1.023	381	8
	CS	$0,00001$	$0,01701$	244	966	7
	FA	$0,00001$	$0,00100$	128	128	1
	GP	$0,00001$	$0,10000$	1.024	775	9
	PSO	$0,00001$	$0,00100$	452	596	3

Tabela 22 – Hiperparâmetros encontrados durante a otimização da TS-GAN (gerador LSTM, discriminador RNN, função de perda Contrastiva e função de distância Euclidiana) na base de dados Avaliações de Consumidores da Amazon.

Modelo	Otimizador	BLEU-1	BLEU-2	BLEU-3	METEOR	ROUGE-1	ROUGE-2	ROUGE-L
TS-GAN	BA	$\mathbf{23,76 \pm 8,28}$	$\mathbf{16,24 \pm 4,58}$	$\mathbf{11,87 \pm 3,20}$	$\mathbf{23,90 \pm 9,41}$	$\mathbf{24,89 \pm 9,03}$	$\mathbf{11,67 \pm 2,62}$	$\mathbf{22,80 \pm 7,65}$
	CS	$\underline{\mathbf{27,13 \pm 5,39}}$	$\underline{\mathbf{17,99 \pm 2,98}}$	$\mathbf{13,03 \pm 2,08}$	$\mathbf{27,32 \pm 6,08}$	$\mathbf{28,34 \pm 5,80}$	$\mathbf{12,49 \pm 1,69}$	$\mathbf{25,54 \pm 4,87}$
	FA	$\mathbf{26,07 \pm 5,69}$	$\mathbf{17,32 \pm 3,02}$	$\mathbf{12,40 \pm 1,98}$	$\underline{\mathbf{30,90 \pm 7,55}}$	$\underline{\mathbf{29,05 \pm 6,03}}$	$\underline{\mathbf{12,90 \pm 1,85}}$	$\underline{\mathbf{26,57 \pm 5,21}}$
	GP	$\mathbf{26,95 \pm 5,32}$	$\mathbf{17,97 \pm 2,95}$	$\underline{\mathbf{13,13 \pm 2,09}}$	$\mathbf{26,82 \pm 5,87}$	$\mathbf{28,27 \pm 5,77}$	$\mathbf{12,59 \pm 1,70}$	$\mathbf{25,51 \pm 4,86}$
	PSO	$\mathbf{25,39 \pm 7,15}$	$\mathbf{16,99 \pm 3,89}$	$\mathbf{12,33 \pm 2,69}$	$\mathbf{25,26 \pm 7,92}$	$\mathbf{26,54 \pm 7,73}$	$\mathbf{11,93 \pm 2,16}$	$\mathbf{24,03 \pm 6,48}$

Tabela 23 – Métricas pós-otimização da TS-GAN (gerador LSTM, discriminador RNN, função de perda Contrastiva e função de distância Euclidiana) avaliada no conjunto de teste da base de dados Avaliação de Consumidores da Amazon.

A Tabela 24 apresenta os hiperparâmetros encontrados durante a otimização da TS-GAN (gerador LSTM, discriminador RNN, função de perda Contrastiva e função de distância Euclidiana) na base de dados Avaliações do Yelp. Novamente, dentre todas as meta-heurísticas analisadas, apenas a FA não convergiu, enquanto todas as outras técnicas atingiram valores próximos de função objetivo (reforçados pela similaridades dos conjuntos de hiperparâmetros descritos na tabela).

A Tabela 25 descreve as métricas pós-otimização da TS-GAN (gerador LSTM, discriminador RNN, função de perda Contrastiva e função de distância Euclidiana) avaliada no conjunto de teste da base de dados Avaliações do Yelp. Note que em termos absolutos, o processo de otimização foi incapaz de aprimorar os resultados das arquiteturas sem otimização (Tabela 19). A meta-heurística PSO (melhores resultados absolutos) atingiu resultados $9,73\%$, $6,54\%$ e $5,19\%$ inferiores para as métricas BLEU-3, METEOR e ROUGE-L, respectivamente, embora todos os valores sejam estatisticamente iguais de acordo com o teste de Wilcoxon. Ademais, é importante ressaltar que as técnicas que obtiveram o melhor equilíbrio entre números de neurônios de vetorização e escondidos na Tabela 24 (CS, GP e PSO), foram as técnicas mais próximas dos resultados da Tabela 19 em termos absolutos e atingiram os melhores resultados (de acordo com o teste de Wilcoxon) na Tabela 25.

Modelo	Otimizador	η_G	η_D	n_e	n_h	Δ
TS-GAN	BA	$0,00001$	$0,00100$	1.535	516	8
	CS	$0,00001$	$0,01394$	1.510	1.471	4
	FA	$0,00001$	$0,00100$	128	128	1
	GP	$0,00001$	$0,10000$	1.536	1.131	9
	PSO	$0,00001$	$0,00100$	1.018	1.201	6

Tabela 24 – Hiperparâmetros encontrados durante a otimização da TS-GAN (gerador LSTM, discriminador RNN, função de perda Contrastiva e função de distância Euclidiana) na base de dados Avaliações do Yelp.

Modelo	Otimizador	BLEU-1	BLEU-2	BLEU-3	METEOR	ROUGE-1	ROUGE-2	ROUGE-L
TS-GAN	BA	$28,74 \pm 9,54$	$20,52 \pm 9,97$	$17,16 \pm 10,14$	$24,58 \pm 8,88$	$31,75 \pm 8,02$	$16,09 \pm 8,83$	$28,01 \pm 8,02$
	CS	$\mathbf{36,14 \pm 9,66}$	$\mathbf{27,85 \pm 10,33}$	$\mathbf{24,48 \pm 10,55}$	$\mathbf{31,13 \pm 9,28}$	$\mathbf{37,30 \pm 8,60}$	$\mathbf{22,15 \pm 9,47}$	$\mathbf{33,40 \pm 8,67}$
	FA	$17,61 \pm 0,80$	$9,57 \pm 0,35$	$6,20 \pm 0,23$	$14,86 \pm 0,30$	$22,70 \pm 0,36$	$6,93 \pm 0,17$	$19,26 \pm 0,20$
	GP	$\mathbf{35,51 \pm 9,26}$	$\mathbf{27,12 \pm 10,12}$	$\mathbf{23,70 \pm 10,42}$	$\mathbf{30,52 \pm 9,02}$	$\mathbf{36,88 \pm 8,15}$	$\mathbf{21,52 \pm 9,40}$	$\mathbf{32,95 \pm 8,35}$
	PSO	$\underline{\mathbf{36,77 \pm 8,81}}$	$\underline{\mathbf{28,36 \pm 9,68}}$	$\underline{\mathbf{24,95 \pm 9,96}}$	$\underline{\mathbf{31,65 \pm 8,65}}$	$\underline{\mathbf{37,94 \pm 7,97}}$	$\underline{\mathbf{22,50 \pm 9,00}}$	$\underline{\mathbf{33,94 \pm 8,14}}$

Tabela 25 – Métricas pós-otimização da TS-GAN (gerador LSTM, discriminador RNN, função de perda Contrastiva e função de distância Euclidiana) avaliada no conjunto de teste da base de dados Avaliações do Yelp.

A Tabela 26 informa os hiperparâmetros encontrados durante a otimização da TS-GAN (gerador LSTM, discriminador GRU, função de perda Contrastiva e função de distância Manhattan) na base de dados Legendas de Imagens COCO. Ao analisarmos os hiperparâmetros encontrados, é possível observar que embora a técnica FA tenha convergido na Figura 32, ela foi incapaz de encontrar parâmetros diferentes do intervalo inferior estabelecido pela Tabela 8. A convergência, neste caso, deve-se à simplicidade do problema em questão, onde um número menor de neurônios é capaz de atingir resultados satisfatórios e comparáveis às outras técnicas.

A Tabela 27 apresenta as métricas pós-otimização da TS-GAN (gerador LSTM, discriminador GRU, função de perda Contrastiva e função de distância Manhattan) avaliada no conjunto de teste da base de dados Legendas de Imagens COCO. Ao compararmos as Tabelas 20 e 27, apenas a meta-heurística PSO foi capaz de aprimorar os resultados sem otimização em termos absolutos (METEOR e ROUGE-L), enquanto todas as outras meta-heurísticas atingiram resultados pouco inferiores (CS e FA) e muito inferiores (BA e GP). A péssima efetividade das técnicas BA e GP é explicada pelos valores de taxa de aprendizado de gerador encontradas (3 ordens de grandeza maiores que as de CS, FA e PSO), os quais foram responsáveis por atribuir passos largos ao otimizador e dificultar a procura de ótimos locais/globais. Ao observamos a Figura 32, as meta-heurísticas BA e GP não convergiram tão efetivamente quanto as outras

Modelo	Otimizador	η_G	η_D	n_e	n_h	Δ
TS-GAN	BA	$0,02647$	$0,05401$	419	563	7
	CS	$0,00001$	$0,04735$	327	429	5
	FA	$0,00001$	$0,00100$	128	128	1
	GP	$0,01850$	$0,02565$	359	768	7
	PSO	$0,00001$	$0,05955$	320	471	7

Tabela 26 – Hiperparâmetros encontrados durante a otimização da TS-GAN (gerador LSTM, discriminador GRU, função de perda Contrastiva e função de distância Manhattan) na base de dados Legendas de Imagens COCO.

meta-heurísticas e reforçam a incapacidade de treinamento quando utilizadas com uma taxa de aprendizado alta.

Modelo	Otimizador	BLEU-1	BLEU-2	BLEU-3	METEOR	ROUGE-1	ROUGE-2	ROUGE-L
TS-GAN	BA	$25,34 \pm 0,09$	$21,47 \pm 0,04$	$16,64 \pm 0,02$	$19,04 \pm 0,65$	$25,37 \pm 0,10$	$18,20 \pm 0,03$	$25,36 \pm 0,09$
	CS	$\mathbf{58,20 \pm 3,44}$	$\mathbf{48,54 \pm 3,60}$	$\mathbf{42,10 \pm 3,84}$	$\mathbf{51,20 \pm 3,72}$	$\mathbf{58,64 \pm 3,23}$	$\mathbf{40,55 \pm 3,52}$	$\mathbf{57,87 \pm 3,14}$
	FA	$\mathbf{57,65 \pm 2,01}$	$\mathbf{48,04 \pm 2,13}$	$\mathbf{41,69 \pm 2,22}$	$\mathbf{50,71 \pm 2,34}$	$\mathbf{58,91 \pm 1,65}$	$\mathbf{40,52 \pm 2,02}$	$\mathbf{58,13 \pm 1,63}$
	GP	$25,52 \pm 0,25$	$21,56 \pm 0,12$	$16,70 \pm 0,08$	$19,23 \pm 0,72$	$25,54 \pm 0,25$	$18,22 \pm 0,04$	$25,53 \pm 0,25$
	PSO	$\underline{\mathbf{59,54 \pm 1,52}}$	$\underline{\mathbf{49,78 \pm 1,92}}$	$\underline{\mathbf{43,36 \pm 2,06}}$	$\underline{\mathbf{52,29 \pm 1,91}}$	$\underline{\mathbf{59,84 \pm 1,57}}$	$\underline{\mathbf{41,61 \pm 2,13}}$	$\underline{\mathbf{58,97 \pm 1,68}}$

Tabela 27 – Métricas pós-otimização da TS-GAN (gerador LSTM, discriminador GRU, função de perda Contrastiva e função de distância Manhattan) avaliada no conjunto de teste da base de dados Legendas de Imagens COCO.

A Tabela 28 descreve os hiperparâmetros encontrados durante a otimização da TS-GAN (gerador LSTM, discriminador GRU, função de perda Contrastiva e função de distância Euclidiana) na base de dados Notícias do WMT EMNLP2017. Novamente, a meta-heurística FA foi incapaz de selecionar valores diferentes do intervalo inferior, enquanto todas as outras técnicas encontraram valores plausíveis durante o processo de otimização, fato que é reforçado pelas curvas de convergência ilustradas pela Figura 33.

A Tabela 29 apresenta as métricas pós-otimização da TS-GAN (gerador LSTM, discriminador GRU, função de perda Contrastiva e função de distância Euclidiana) avaliada no conjunto de teste da base de dados Notícias do WMT EMNLP2017. Ao compararmos as Tabelas 21 e 29, apenas as meta-heurísticas CS e PSO atingiram baixos desvios padrões e resultados comparáveis à TS-GAN-C[39] sem otimização. Ademais, PSO foi capaz de aprimorar todas as métricas analisadas (BLEU-3, METEOR e ROUGE-L) quando comparada com a arquitetura sem otimização. Este acréscimo de efetividade foi capaz de alavancar a posição da

[39] A sigla TS-GAN-C é utilizada pois estamos considerando a função de perda Contrastiva neste experimento.

Modelo	Otimizador	η_G	η_D	n_e	n_h	Δ
TS-GAN	BA	$0,00001$	$0,00100$	1.023	381	8
	CS	$0,00001$	$0,07720$	288	785	1
	FA	$0,00001$	$0,00100$	128	128	1
	GP	$0,00001$	$0,10000$	1.024	775	9
	PSO	$0,00001$	$0,00100$	519	817	5

Tabela 28 – Hiperparâmetros encontrados durante a otimização da TS-GAN (gerador LSTM, discriminador GRU, função de perda Contrastiva e função de distância Euclidiana) na base de dados Notícias do WMT EMNLP2017.

TS-GAN-C dentre as arquiteturas adversariais, atingindo o melhor resultado quando comparada com os resultados estado da arte (anteriormente, apenas a GSGAN tinha obtido resultados melhores que a TS-GAN-C).

Modelo	Otimizador	BLEU-1	BLEU-2	BLEU-3	METEOR	ROUGE-1	ROUGE-2	ROUGE-L
TS-GAN	BA	$\mathbf{24,07 \pm 4,32}$	$\mathbf{16,90 \pm 2,56}$	$\mathbf{12,65 \pm 1,90}$	$\mathbf{25,26 \pm 5,30}$	$\mathbf{25,20 \pm 4,71}$	$\mathbf{12,40 \pm 1,60}$	$\mathbf{23,63 \pm 4,19}$
	CS	$\underline{\mathbf{25,23 \pm 1,08}}$	$\mathbf{17,34 \pm 0,89}$	$\mathbf{12,88 \pm 0,78}$	$\mathbf{25,49 \pm 2,38}$	$\mathbf{25,99 \pm 1,28}$	$\mathbf{12,28 \pm 0,82}$	$\mathbf{24,10 \pm 1,28}$
	FA	$18,01 \pm 5,65$	$12,39 \pm 2,65$	$8,88 \pm 1,67$	$14,08 \pm 4,90$	$18,26 \pm 5,95$	$8,77 \pm 1,33$	$16,87 \pm 4,89$
	GP	$\mathbf{22,71 \pm 5,80}$	$\mathbf{16,14 \pm 3,45}$	$\mathbf{12,20 \pm 2,61}$	$\mathbf{23,64 \pm 7,10}$	$\mathbf{23,84 \pm 6,37}$	$\mathbf{12,06 \pm 2,20}$	$\mathbf{22,42 \pm 5,66}$
	PSO	$\mathbf{25,21 \pm 1,10}$	$\underline{\mathbf{17,58 \pm 0,78}}$	$\underline{\mathbf{13,25 \pm 0,63}}$	$\underline{\mathbf{26,21 \pm 1,95}}$	$\underline{\mathbf{26,23 \pm 1,37}}$	$\underline{\mathbf{12,76 \pm 0,67}}$	$\underline{\mathbf{24,45 \pm 1,28}}$

Tabela 29 – Métricas pós-otimização da TS-GAN (gerador LSTM, discriminador GRU, função de perda Contrastiva e função de distância Euclidiana) avaliada no conjunto de teste da base de dados Notícias do WMT EMNLP2017.

8.2.3 Análise do Custo Computacional

O último experimento visa comparar o custo computacional (tempo em horas) entre as arquiteturas analisadas (recorrentes, adversariais e otimizadas). As Tabelas 30, 31, 32 e 33 apresentam os tempos médios de treinamento e otimização, em horas, dos modelos recorrentes e adversariais nas bases de dados Avaliação do Consumidores da Amazon, Avaliações do Yelp, Legendas de Imagens COCO e Notícias do WMT EMNLP2017, respectivamente.

Tipo	Modelo	Treino	BA	CS	FA	GP	PSO
Recorrente	RNN	$\underline{\mathbf{0,71 \pm 0,07}}$	–	–	–	–	–
	LSTM	$1,08 \pm 0,10$	–	–	–	–	–
	GRU	$1,04 \pm 0,11$	–	–	–	–	–
	RMC	$5,07 \pm 0,15$	–	–	–	–	–
Adversarial	SeqGAN	$36,18 \pm 9,92$	–	–	–	–	–
	MaliGAN	$5,57 \pm 1,19$	–	–	–	–	–
	GSGAN	$1,35 \pm 0,14$	–	–	–	–	–
	RelGAN	$5,75 \pm 0,29$	–	–	–	–	–
	TS-GAN	$14,67 \pm 2,53$	$135,05 \pm 6,18$	$192,90 \pm 8,31$	$65,00 \pm 0,43$	$70,45 \pm 4,12$	$67,70 \pm 5,38$

Tabela 30 – Tempo médio de treinamento e otimização, em horas, dos modelos recorrentes e adversariais na base de dados Avaliação do Consumidores da Amazon.

Tipo	Modelo	Treino	BA	CS	FA	GP	PSO
Recorrente	RNN	$\underline{\mathbf{0,79 \pm 0,01}}$	–	–	–	–	–
	LSTM	$1,16 \pm 0,01$	–	–	–	–	–
	GRU	$1,05 \pm 0,01$	–	–	–	–	–
	RMC	$4,40 \pm 0,01$	–	–	–	–	–
Adversarial	SeqGAN	$17,72 \pm 0,49$	–	–	–	–	–
	MaliGAN	$3,13 \pm 0,07$	–	–	–	–	–
	GSGAN	$1,96 \pm 0,02$	–	–	–	–	–
	RelGAN	$7,09 \pm 0,04$	–	–	–	–	–
	TS-GAN	$7,21 \pm 0,09$	$105,58 \pm 9,47$	$161,75 \pm 8,23$	$42,57 \pm 0,75$	$56,22 \pm 3,73$	$53,45 \pm 4,01$

Tabela 31 – Tempo médio de treinamento e otimização, em horas, dos modelos recorrentes e adversariais na base de dados Avaliações do Yelp.

Tipo	Modelo	Treino	BA	CS	FA	GP	PSO
Recorrente	RNN	$\mathbf{0,20 \pm 0,32}$	–	–	–	–	–
	LSTM	$\mathbf{0,37 \pm 0,56}$	–	–	–	–	–
	GRU	$\underline{\mathbf{0,19 \pm 0,36}}$	–	–	–	–	–
	RMC	$\mathbf{0,52 \pm 0,95}$	–	–	–	–	–
Adversarial	SeqGAN	$3,51 \pm 3,34$	–	–	–	–	–
	MaliGAN	$2,10 \pm 2,68$	–	–	–	–	–
	GSGAN	$0,72 \pm 1,11$	–	–	–	–	–
	RelGAN	$0,46 \pm 0,27$	–	–	–	–	–
	TS-GAN	$1,26 \pm 0,55$	$36,67 \pm 13,12$	$51,82 \pm 7,47$	$10,84 \pm 0,25$	$21,87 \pm 4,66$	$18,40 \pm 5,40$

Tabela 32 – Tempo médio de treinamento e otimização, em horas, dos modelos recorrentes e adversariais na base de dados Legendas de Imagens COCO.

Tipo	Modelo	Treino	BA	CS	FA	GP	PSO
Recorrente	RNN	$\underline{\mathbf{0,98 \pm 0,00}}$	–	–	–	–	–
	LSTM	$1,44 \pm 0,01$	–	–	–	–	–
	GRU	$1,38 \pm 0,01$	–	–	–	–	–
	RMC	$6,73 \pm 0,01$	–	–	–	–	–
Adversarial	SeqGAN	$33,37 \pm 0,55$	–	–	–	–	–
	MaliGAN	$6,00 \pm 0,04$	–	–	–	–	–
	GSGAN	$1,75 \pm 0,00$	–	–	–	–	–
	RelGAN	$7,45 \pm 0,01$	–	–	–	–	–
	TS-GAN	$14,97 \pm 0,10$	$189,10 \pm 11,05$	$278,65 \pm 8,11$	$92,15 \pm 0,17$	$100,50 \pm 4,51$	$97,45 \pm 6,07$

Tabela 33 – Tempo médio de treinamento e otimização, em horas, dos modelos recorrentes e adversariais na base de dados Notícias do WMT EMNLP2017.

Note que para todos as bases de dados, exceto pela Legendas de Imagens COCO, as RNNs obtiveram o menor tempo de treinamento de acordo com o teste de Wilcoxon. A exceção (Legendas de Imagens COCO) deve-se à simplicidade da base de dados e ao pequeno tamanho de sequência utilizado, o qual é estatisticamente indiferente quando comparados às outras arquiteturas recorrentes.

Ao compararmos as arquiteturas adversariais, é possível observar que as SeqGANs obtiveram os piores tempos de treinamento devido à utilização das cadeias de Monte Carlo (operações de alta complexidade) para a geração de recompensas utilizadas nos gradientes do gerador. Em contrapartida, as TS-GAN demonstraram alta eficiência quando utilizadas em tamanhos de sequência pequenos (Legendas de Imagens COCO), superando os tempos de treinamento da MaliGAN. Entretanto, em bases de dados com tamanhos de sequência maiores, a MaliGAN obteve um tempo médio de treinamento menor que a TS-GAN[40]. Por fim, as meta-heurísticas atingiram os piores tempos de treinamento devido à intrinsidade do processo de otimização, o qual avalia n vezes a função objetivo[41].

[40] Ambas arquiteturas utilizam o mesmo algoritmo de treinamento, porém, a TS-GAN necessita de mais tempo para treinar o discriminador baseado em similaridade com tamanhos de sequência grandes

[41] Funções objetivos mais demoradas, por exemplo, treinamento e validação de arquiteturas, ocasionam em um maior tempo de otimização.

9 Conclusões e Trabalhos Futuros

O Processamento de Linguagem Natural tornou-se uma ferramenta amplamente utilizada em nosso cotidiano. A possibilidade de compreender e reproduzir automaticamente a capacidade linguística do ser humano fomenta e desenvolve esta área. Entretanto, os problemas desta área não são triviais, necessitando de conhecimentos mais robustos para a solução dos mesmos. Na última década, boa parte da comunidade de Inteligência Artificial direcionou seus recursos para a resolução de problemas de Processamento de Linguagem Natural, atingindo resultados estado da arte nas mais diversas tarefas, tais como, classificação de textos, análise de sentimentos, reconhecimento de entidades nomeadas, dentre outras.

Uma tarefa que desperta grande interesse é a geração de texto, pois serve como base de aprendizado de como uma linguagem é estruturada e utilizada. Em outras palavras, a geração de texto permite com que a máquina entenda a estrutura de nossa forma de comunicação, bem como a utilize para realizar tarefas mais complexas e que necessitem de uma base de compreensão. Entretanto, tal tarefa é de difícil execução, dado que o texto gerado precisa ser lexicalmente correto, sintaticamente adequado e semanticamente plausível. Adicionalmente, sequências de textos são estruturas discretas, não sendo perfeitamente adequáveis para a metodologia das Redes Neurais Recorrentes ou Redes Adversariais Generativas. Outro problema do Aprendizado de Máquina em geral é a vasta utilização de hiperparâmetros, os quais são extremamente sensíveis ao contexto de aplicação. Recentemente, técnicas inspiradas pela natureza, denominadas de meta-heurísticas, vem sendo utilizadas para modelar tal problema em um processo de otimização, onde seus agentes guiam uma função objetivo em busca dos valores mais aptos à minimizá-la.

Esta tese aborda duas frentes a fim de solucionar os problemas supracitados, isto é, geração artificial de texto e seleção de hiperparâmetros de modelos de Processamento de Linguagem Natural. A primeira frente é estruturada através da proposição da Rede Adversarial Generativa por Similaridade Textual, a qual é composta por um discriminador com funções de perda baseadas em similaridade. Ademais, esta arquitetura utiliza a similaridade com o objetivo de melhorar a discriminação entre dados artificiais e reais, bem como atacar o problema da instabilidade de convergência do treinamento de modelos adversariais. Os experimentos conduzidos na Frente I demonstram a capacidade da arquitetura proposta, atingindo resultados comparativos às arquiteturas estado da arte e aprimorando o estado da arte em duas de quatro bases de dados. Ademais, o experimento de busca pela arquitetura ideal (tipo de gerador, discriminador, função de perda e função de distância) foi capaz de aprimorar os resultados obtidos pelas versões iniciais da TS-GAN.

A segunda frente foi estruturada através da utilização de meta-heurísticas, tais como,

Algoritmo do Morcego, Algoritmo do Vagalume, Busca do Cuckoo, Otimização por Enxame de Partículas e Programação Genética, para selecionar os hiperparâmetros dos modelos aplicados na primeira frente. O intuito é verificar a existência de conjuntos de hiperparâmetros melhores, bem como, possíveis aperfeiçoamentos para as arquiteturas utilizadas na Frente I. Os resultados experimentais obtidos na Frente II demonstraram a capacidade do processo de otimização em encontrar conjuntos de hiperparâmetros capazes de aprimorar as métricas analisadas, embora ocasionaram um aumento no tempo total do procedimento (treinamento + otimização).

Além das metodologias e experimentos propostos para solucionar os dois problemas mencionados, a presente tese contribuiu para a comunidade científica através da implementação de bibliotecas de Adversarial aplicadas ao Processamento de Linguagem Natural, Redes Siamesas e otimização meta-heurística, e através da publicação de resultados e inovações em conferências e periódicos correlatos.

Finalmente, conforme demonstrado nos experimentos propostos, a utilização de técnicas meta-heurísticas e, virtualmente, quaisquer processos de pós-otimização, demonstram-se bastante úteis. Entretanto, o alto custo computacional inviabiliza sua aplicação em problemas cotidianos, inclusive em aplicações *on-the-fly*, isto é, processamento e saída de dados em tempo real. Desta forma, uma interessante área que está recebendo atenção da comunidade de pesquisa é a utilização de métodos de aproximação durante o processo de otimização, ou seja, a utilização de funções objetivos mais eficientes e com menores custos computacionais, e que proporcionam mesma efetividade aos objetivos tradicionais (treinamento completo). Portanto, a pesquisa em funções objetivos mais eficientes através da exploração de técnicas de aproximação, por exemplo, ranqueamento de arquiteturas, é uma dos principais ideias a serem exploradas em trabalhos futuros.

Publicações Relacionadas

RIBEIRO, L. C. F.; ROSA, G. H.; RODRIGUES, D.; PAPA, J. P. Convolutional neural networks ensembles through single-iteration optimization. *Soft Computing*, 2022.

ROSA, G. H.; PAPA, J. P. Learning to weight similarity measures with Siamese networks: a case study on optimum-path forest. *Optimum-Path Forest: Theory, Algorithms, and Applications*, 2022.

ROSA, G. H.; RODER, M.; SANTOS, C. F. G.; PAPA, J. P. Improving Pre-Trained Weights through Meta-Heuristics Fine-Tuning. *IEEE Symposium Series on Computational Intelligence*, 2021.

ROSA, G. H.; PAPA, J. P. A Survey on Text Generation using Generative Adversarial Networks. *Pattern Recognition*, 2021.

RODER, M.; PASSOS, L. A.; ROSA, G. H.; ALBUQUERQUE, V., H.; PAPA, J. P. Reinforcing Learning in Deep Belief Networks Through Nature-Inspired Optimization. *Applied Soft Computing*, 2021.

FELIX, L. C. R.; RODER, M.; ROSA, G. H.; PASSOS, L. A.; PAPA, J. P. Enhancing Hyper-To-Real Space Projections Through Euclidean Norm Meta-Heuristic Optimization. *25th Iberoamerican Congress on Pattern Recognition*, 2021.

ROSA, G. H.; RODER, M.; PAPA, J. P. Fine-Tuning Dropout Regularization in Energy-Based Deep Learning. *25th Iberoamerican Congress on Pattern Recognition*, 2021.

ROSA, G. H.; RODER, M.; PAPA, J. P. Fast Ensemble Learning Using Adversarially-Generated Restricted Boltzmann Machines. *arXiv preprint arXiv:2101.01042*, 2021.

FERREIRA JR., A. R.; ROSA, G. H.; PAPA, J. P.; CARNEIRO, G.; FARIA, F. A. Creating Classifier Ensembles through Meta-heuristic Algorithms for Aerial Scene Classification. *International Conference on Pattern Recognition*, 2021.

RODER, M.; ROSA, G. H.; BREVE, F.; PAPA, J. P. Fine-Tuning Temperatures in Restricted Boltzmann Machines Using Meta-Heuristic Optimization. *IEEE Congress on Evolutionary Computation*, 2020.

PASSOS, L. A.; ROSA, G. H.; RODRIGUES, D.; RODER, M.; PAPA, J. P. On the Assessment of Nature-Inspired Meta-Heuristic Optimization Techniques to Fine-Tune Deep Belief Networks. *Natural Computing Series*, v. 1, p. 67-96, 2020.

ROSA, G. H.; RODRIGUES, D.; PAPA, J. P. Opytimizer: A Nature-Inspired Python Optimizer. *arXiv preprint arXiv:1912.13002*, 2019.

ROSA, G. H.; BREGA, J. R. F.; PAPA, J. P. How optimizing perplexity can affect the dimensionality reduction on word embeddings visualization?. *SN Applied Sciences*, v. 1, 2019.

ROSA, G. H.; PAPA, J. P. Soft-Tempering Deep Belief Networks Parameters Through Genetic Programming. *Journal of Artificial Intelligence and Systems*, v. 1, p. 43-59, 2019.

Publicações Realizadas

OLIVEIRA, G. C.; ROSA, G. H.; PEDRONETTE, D. C. G.; PAPA, J. P.; KUMAR, H.; PASSOS, L. A.; KUMAR, D. Which Generative Adversarial Network Yields High-Quality Synthetic Medical Images: Investigation Using AMD Image Datasets. *arXiv preprint arXiv:2203.13856, 2022*.

ROSA, G. H.; RODER, M.; PAPA, J. P. Comparative Study Between Distance Measures On Supervised Optimum-Path Forest Classification. *arXiv preprint arXiv:2202.03854, 2022*.

RODER, M.; ALMEIDA, J.; ROSA, G. H.; PASSOS, L. A.; ROSSI, A. L. D.; PAPA, J. P. From Actions to Events: Video Domain Adaptation Using Spectral Deep Belief Networks. *IEEE Symposium Series on Computational Intelligence*, 2021.

ROSA, G. H.; RODER, M.; PAPA, J. P. Neighbor-based Bag-of-Samplings for Person Identification Through Handwritten Dynamics and Convolutional Neural Networks. *Expert Systems*, 2021.

RODER, M.; ROSA, G. H.; PAPA, J. P.; PEDRONETTE, D. C. G. Enhancing Shallow Neural Networks Through Fourier-based Information Fusion for Stroke Classification. *34th SIBGRAPI - Conference on Graphics, Patterns and Images*, 2021.

ROSA, G. H.; PAPA, J. P. OPFython: A Python implementation for Optimum-Path Forest. *Software Impacts*, 2021.

ROSA, G. H.; PAPA, J. P. Speeding Up OPFython with Numba. *arXiv preprint arXiv:2106.11828*, 2021.

RODER, M.; ROSA, G. H.; ALBUQUERQUE, V. H. C.; ROSSI, A.; PAPA, J. P. Energy-Based Dropout in Restricted Boltzmann Machines: Why Not Go Random. *IEEE Transactions on Emerging Topics in Computational Intelligence*, v. 1, p. 1-11, 2020.

RODER, M.; ROSA, G. H.; PASSOS, L. A.; ROSSI, A.; PAPA, J. P. Harnessing Particle Swarm Optimization Through Relativistic Velocity. *IEEE Congress on Evolutionary Computation*, 2020.

ROSA, G. H.; PAPA, J. P.; YANG, X.-S. A nature-inspired feature selection approach based on hypercomplex information. *Applied Soft Computing*, v. 1, p. 106453, 2020.

RODRIGUES, D.; ROSA, G. H.; PASSOS, L. A.; PAPA, J. P. Adaptive Improved Flower Pollination Algorithm for Global Optimization. *Studies in Computational Intelligence*, v. 855, p. 1-21, 2020.

PASSOS, L. A.; ROSA, G. H.; RODRIGUES, D.; PAPA, J. P. Fine-tuning restricted Boltzmann machines using quaternion-based flower pollination algorithm. *Nature-Inspired Computation and Swarm Intelligence*, v. 1, p. 111-133, 2020.

ROSA, G. H.; PAPA, J. P; FALCÃO, A. X. OPFython: A Python-Inspired Optimum-Path Forest Classifier. *arXiv preprint arXiv:2001.10420*, 2020.

RODER, M.; ROSA, G. H.; PAPA, J. P. Learnergy: Energy-based Machine Learners. *arXiv preprint arXiv:2003.07443*, 2020.

AFONSO, L. C. S.; ROSA, G. H.; PEREIRA, C. R.; WEBER, S. A. T.; HOOK, C.; ALBUQUERQUE, V. H. C.; PAPA, J. P. A recurrence plot-based approach for Parkinson's disease identification. *Future Generation Computer Systems*, v. 94, p. 282-292, 2019.

AMORIM, W. P.; ROSA, G. H.; THOMAZELLA, R.; CASTANHO, J. E. C.; DOTTO, F. R. L.; JÚNIOR, O. P. R.; MARANA, A. N.; PAPA, J. P. Semi-Supervised Learning with Connectivity-driven Convolutional Neural Networks. *Pattern Recognition Letters*, v. 128, p. 16-22, 2019.

Referências

ASGHAR, N. *Yelp Dataset Challenge: Review Rating Prediction*. 2016.

BAHDANAU, D.; CHO, K.; BENGIO, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

BALDI, P.; CHAUVIN, Y. Neural networks for fingerprint recognition. *Neural Computation*, MIT Press, v. 5, n. 3, p. 402–418, 1993.

BANERJEE, S.; LAVIE, A. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In: *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*. [S.l.: s.n.], 2005. p. 65–72.

BARONE, V.; COSSI, M.; TOMASI, J. Geometry optimization of molecular structures in solution by the polarizable continuum model. *Journal of Computational Chemistry*, Wiley Online Library, v. 19, n. 4, p. 404–417, 1998.

BENGIO, Y.; DUCHARME, R.; VINCENT, P.; JAUVIN, C. A neural probabilistic language model. *Journal of machine learning research*, v. 3, n. Feb, p. 1137–1155, 2003.

BERTSEKAS, D. *Nonlinear programming*. [S.l.]: Athena Scientific, 1999.

BIGGIO, B.; ROLI, F. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, Elsevier, v. 84, p. 317–331, 2018.

BROMLEY, J.; GUYON, I.; LECUN, Y.; SäCKINGER, E.; SHAH, R. Signature verification using a siamese time delay neural network. In: *Proceedings of the 6th International Conference on Neural Information Processing Systems*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993. (NIPS'93), p. 737–744.

BROWN, T.; LIEBOVITCH, L.; GLENDON, R. Lévy flights in dobe ju/'hoansi foraging patterns. *Human Ecology*, Springer, v. 35, n. 1, p. 129–138, 2007.

CHE, T.; LI, Y.; ZHANG, R.; HJELM, R.; LI, W.; SONG, Y.; BENGIO, Y. Maximum-likelihood augmented discrete generative adversarial networks. *arXiv preprint arXiv:1702.07983*, 2017.

CHEN, X.; FANG, H.; LIN, T.-Y.; VEDANTAM, R.; GUPTA, S.; DOLLÁR, P.; ZITNICK, C. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015.

CHENG, Q.; BANDLER, J.; KOZIEL, S. Space mapping design framework exploiting tuning elements. *IEEE Transactions on Microwave Theory and Techniques*, IEEE, v. 58, n. 1, p. 136–144, 2010.

CHO, K.; MERRIENBOER, B. V.; GULCEHRE, C.; BAHDANAU, D.; BOUGARES, F.; SCHWENK, H.; BENGIO, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

DENG, L.; LIU, Y. *Deep learning in natural language processing*. [S.l.]: Springer, 2018.

DIXIT, A. *Optimization in economic theory*. [S.l.]: Oxford University Press, 1990.

DUŠEK, O.; NOVIKOVA, J.; RIESER, V. Evaluating the state-of-the-art of end-to-end natural language generation: The e2e nlg challenge. *Computer Speech & Language*, Elsevier, v. 59, p. 123–156, 2020.

EISENSTEIN, J. *Introduction to natural language processing*. [S.l.]: MIT press, 2019.

ELMAN, J. Finding structure in time. *Cognitive science*, Wiley Online Library, v. 14, n. 2, p. 179–211, 1990.

FAN, A.; LEWIS, M.; DAUPHIN, Y. *Hierarchical Neural Story Generation*. 2018.

FOSLER-LUSSIER, E. Markov models and hidden markov models: A brief tutorial. *International Computer Science Institute*, Citeseer, 1998.

FRANK, C. A step towards automatic documentation. MIT Artificial Intelligence Laboratory, 1980.

GALASSI, A.; LIPPI, M.; TORRONI, P. Attention in natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*, IEEE, v. 32, n. 10, p. 4291–4308, 2020.

GATT, A.; KRAHMER, E. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, v. 61, p. 65–170, 2018.

GIMÉNEZ, M.; PALANCA, J.; BOTTI, V. Semantic-based padding in convolutional neural networks for improving the performance in natural language processing. a case of study in sentiment analysis. *Neurocomputing*, Elsevier, v. 378, p. 315–323, 2020.

GLOROT, X.; BORDES, A.; BENGIO, Y. Deep sparse rectifier neural networks. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. [S.l.: s.n.], 2011. p. 315–323.

GOLDBERG, D.; KUO, C. Genetic algorithms in pipeline optimization. *Journal of Computing in Civil Engineering*, American Society of Civil Engineers, v. 1, n. 2, p. 128–141, 1987.

Goldberg, E.; Driedger, N.; Kittredge, R. I. Using natural-language processing to produce weather forecasts. *IEEE Expert*, v. 9, n. 2, p. 45–53, 1994.

GOODFELLOW, I.; POUGET-ABADIE, J.; MIRZA, M.; XU, B.; WARDE-FARLEY, D.; OZAIR, S.; COURVILLE, A.; BENGIO, Y. Generative adversarial nets. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2014. p. 2672–2680.

GRAVES, A.; JAITLY, N.; MOHAMED, A.-R. Hybrid speech recognition with deep bidirectional lstm. In: IEEE. *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. [S.l.], 2013. p. 273–278.

GRAVES, A.; WAYNE, G.; DANIHELKA, I. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Hadsell, R.; Chopra, S.; LeCun, Y. Dimensionality reduction by learning an invariant mapping. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. [S.l.: s.n.], 2006. v. 2, p. 1735–1742.

HARRIS, M. Building a large-scale commercial nlg system for an emr. In: *Proceedings of the Fifth International Natural Language Generation Conference*. [S.l.: s.n.], 2008. p. 157–160.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural computation*, MIT Press, v. 9, n. 8, p. 1735–1780, 1997.

HOLTZMAN, A.; BUYS, J.; DU, L.; FORBES, M.; CHOI, Y. *The Curious Case of Neural Text Degeneration*. 2020.

JAIN, P. K.; SARAVANAN, V.; PAMULA, R. A hybrid cnn-lstm: A deep learning approach for consumer sentiment analysis using qualitative user-generated contents. *Transactions on Asian and Low-Resource Language Information Processing*, ACM New York, NY, v. 20, n. 5, p. 1–15, 2021.

JANG, E.; GU, S.; POOLE, B. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

JOLICOEUR-MARTINEAU, A. The relativistic discriminator: a key element missing from standard gan. *arXiv preprint arXiv:1807.00734*, 2018.

JR., I. F.; YANG, X.-S.; FISTER, I.; BREST, J.; FISTER, D. *A Brief Review of Nature-Inspired Algorithms for Optimization*. 2013.

KENNEDY, J.; EBERHART, R.; RUSSEL, C.; KENNEDY, J.; SHI, Y. *Swarm Intelligence*. [S.l.]: Morgan Kaufmann, 2001.

KIM, Y. Convolutional neural networks for sentence classification. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, 2014. p. 1746–1751.

KINGMA, D.; BA, J. *Adam: A Method for Stochastic Optimization*. 2017.

KONNO, H.; YAMAZAKI, H. Mean-absolute deviation portfolio optimization model and its applications to tokyo stock market. *Management Science*, INFORMS, v. 37, n. 5, p. 519–531, 1991.

KOZA, J. Genetic programming as a means for programming computers by natural selection. *Statistics and computing*, Springer, v. 4, n. 2, p. 87–112, 1994.

KUSNER, M.; HERNáNDEZ-LOBATO, J. *GANS for Sequences of Discrete Elements with the Gumbel-softmax Distribution*. 2016.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, IEEE, v. 86, n. 11, p. 2278–2324, 1998.

LECUN, Y.; KAVUKCUOGLU, K.; FARABET, C. Convolutional networks and applications in vision. In: *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. [S.l.]: IEEE, 2010. p. 253–256.

LI, Y.; PAN, Q.; WANG, S.; YANG, T.; CAMBRIA, E. A generative model for category text generation. *Information Sciences*, v. 450, p. 301 – 315, 2018. ISSN 0020-0255.

LIN, C.-Y. Rouge: A package for automatic evaluation of summaries. In: *Text summarization branches out*. [S.l.: s.n.], 2004. p. 74–81.

MCDONALD, D. Natural language generation. *Handbook of Natural Language Processing*, v. 2, p. 121–144, 2010.

MORRIS, J. X.; LIFLAND, E.; YOO, J. Y.; QI, Y. Textattack: A framework for adversarial attacks in natural language processing. *Proceedings of the 2020 EMNLP*, 2020.

MULDER, W. D.; BETHARD, S.; MOENS, M.-F. A survey on the application of recurrent neural networks to statistical language modeling. *Computer Speech & Language*, Elsevier, v. 30, n. 1, p. 61–98, 2015.

NASEEM, U.; RAZZAK, I.; KHAN, S. K.; PRASAD, M. A comprehensive survey on word representation models: From classical to state-of-the-art word representation language models. *Transactions on Asian and Low-Resource Language Information Processing*, ACM New York, NY, v. 20, n. 5, p. 1–35, 2021.

NI, J.; LI, J.; MCAULEY, J. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. [S.l.: s.n.], 2019. p. 188–197.

NIE, W.; NARODYTSKA, N.; PATEL, A. RelGAN: Relational generative adversarial networks for text generation. In: *International Conference on Learning Representations*. [S.l.: s.n.], 2019.

ONDREJ, B.; CHATTERJEE, R.; CHRISTIAN, F.; YVETTE, G.; BARRY, H.; MATTHIAS, H.; PHILIPP, K.; QUN, L.; VARVARA, L.; CHRISTOF, M. et al. Findings of the 2017 conference on machine translation (wmt17). In: THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Second Conference onMachine Translation*. [S.l.], 2017. p. 169–214.

OTTER, D. W.; MEDINA, J. R.; KALITA, J. K. A survey of the usages of deep learning for natural language processing. *IEEE transactions on neural networks and learning systems*, IEEE, v. 32, n. 2, p. 604–624, 2020.

PAPADIMITRIOU, C. The euclidean travelling salesman problem is np-complete. *Theoretical Computer Science*, Elsevier, v. 4, n. 3, p. 237–244, 1977.

PAPINENI, K.; ROUKOS, S.; WARD, T.; ZHU, W.-J. Bleu: a method for automatic evaluation of machine translation. In: *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. [S.l.: s.n.], 2002. p. 311–318.

PAVLYUKEVICH, I. Lévy flights, non-local search and simulated annealing. *Journal of Computational Physics*, Elsevier, v. 226, n. 2, p. 1830–1844, 2007.

PAYNE, R. *The Cuckoos*. [S.l.]: Oxford University Press, 2005.

PORTET, F.; REITER, E.; GATT, A.; HUNTER, J.; SRIPADA, S.; FREER, Y.; SYKES, C. Automatic generation of textual summaries from neonatal intensive care data. *Artificial Intelligence*, Elsevier, v. 173, n. 7-8, p. 789–816, 2009.

RAO, R.; SAVSANI, V.; VAKHARIA, D. Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. *Computer-Aided Design*, Elsevier, v. 43, n. 3, p. 303–315, 2011.

RAO, S. *Engineering optimization: theory and practice*. [S.l.]: John Wiley & Sons, 2009.

RARDIN, R. *Optimization in operations research*. [S.l.]: Prentice Hall, 1998. v. 166.

REITER, E.; DALE, R. Building applied natural language generation systems. *Natural Language Engineering*, v. 3, 03 2002.

ROSA, G. de; RODRIGUES, D.; PAPA, J. *Opytimizer: A Nature-Inspired Python Optimizer*. 2019.

ROSA, G. H. de; PAPA, J. P. A survey on text generation using generative adversarial networks. *Pattern Recognition*, Elsevier, v. 119, p. 108098, 2021.

ROY, S.; HARANDI, M.; NOCK, R.; HARTLEY, R. Siamese networks: The tale of two manifolds. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. [S.l.: s.n.], 2019. p. 3046–3055.

SANTORO, A.; FAULKNER, R.; RAPOSO, D.; RAE, J.; CHRZANOWSKI, M.; WEBER, T.; WIERSTRA, D.; VINYALS, O.; PASCANU, R.; LILLICRAP, T. Relational recurrent neural networks. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2018. p. 7299–7310.

SARKER, I. H. Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science*, Springer, v. 2, n. 3, p. 1–21, 2021.

SAUPER, C.; BARZILAY, R. Automatically generating wikipedia articles: A structure-aware approach. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. [S.l.], 2009.

Schroff, F.; Kalenichenko, D.; Philbin, J. Facenet: A unified embedding for face recognition and clustering. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2015. p. 815–823.

SILVER, D.; HUANG, A.; MADDISON, C.; GUEZ, A.; SIFRE, L.; DRIESSCHE, G. V. D.; SCHRITTWIESER, J.; ANTONOGLOU, I.; PANNEERSHELVAM, V.; LANCTOT, M. Mastering the game of go with deep neural networks and tree search. *Nature*, Nature Publishing Group, v. 529, n. 7587, p. 484–489, 2016.

SONG, P.; GENG, C.; LI, Z. Research on text classification based on convolutional neural network. In: IEEE. *2019 International Conference on Computer Network, Electronic and Automation (ICCNEA)*. [S.l.], 2019. p. 229–232.

SUTSKEVER, I.; MARTENS, J.; HINTON, G. Generating text with recurrent neural networks. In: *ICML*. [S.l.: s.n.], 2011. p. 1017–1024.

ULLMAN, S. Using neuroscience to develop artificial intelligence. *Science*, v. 363, n. 6428, p. 692–693, 2019.

VASWANI, A.; SHAZEER, N.; PARMAR, K.; USZKOREIT, J.; JONES, L.; GOMEZ, A.; KAISER, L.; POLOSUKHIN, I. Attention is all you need. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2017. p. 5998–6008.

WEN, T.-H.; YOUNG, S. Recurrent neural network language generation for spoken dialogue systems. *Computer Speech & Language*, v. 63, p. 101017, 2020. ISSN 0885-2308.

WILKINSON, P.; LOKE, M.; MELDRUM, P.; CHAMBERS, J.; KURAS, O.; GUNN, D.; OGILVY, R. Practical aspects of applied optimized survey design for electrical resistivity tomography. *Geophysical Journal International*, Oxford University Press, v. 189, n. 1, p. 428–440, 2012.

WILLIAMS, R. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, Springer, v. 8, n. 3-4, p. 229–256, 1992.

WU, Q.; LI, L.; YU, Z. Textgail: Generative adversarial imitation learning for text generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, v. 35, n. 16, p. 14067–14075, May 2021. Disponível em: <https://ojs.aaai.org/index.php/AAAI/article/view/17656>.

YAN, Y.; SHEN, G.; ZHANG, S.; HUANG, T.; DENG, Z.-H.; YUN, U. Sequence generative adversarial nets with a conditional discriminator. *Neurocomputing*, v. 429, p. 69–76, 2021. ISSN 0925-2312.

YANG, X.-S. Firefly algorithm, stochastic test functions and design optimisation. *International Journal Bio-Inspired Computing*, Inderscience Publishers, v. 2, n. 2, p. 78–84, 2010.

YANG, X.-S. A new metaheuristic bat-inspired algorithm. In: *Nature inspired cooperative strategies for optimization (NICSO 2010)*. [S.l.]: Springer, 2010. p. 65–74.

YANG, X.-S. Review of meta-heuristics and generalised evolutionary walk algorithm. *International Journal of Bio-Inspired Computation*, Inderscience Publishers, v. 3, n. 2, p. 77–84, 2011.

YANG, X.-S.; DEB, S. Engineering optimisation by cuckoo search. *International Journal of Mathematical Modelling and Numerical Optimisation*, Inderscience Publishers, v. 1, n. 4, p. 330–343, 2010.

YU, L.; ZHANG, W.; WANG, J.; YU, Y. *SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient*. 2016.

YUAN, L.; ZENG, J.; ZHENG, X. *SparseGAN: Sparse Generative Adversarial Network for Text Generation*. arXiv, 2021. Disponível em: <https://arxiv.org/abs/2103.11578>.

ZHANG, H.; DUCKWORTH, D.; IPPOLITO, D.; NEELAKANTAN, A. Trading off diversity and quality in natural language generation. *arXiv preprint arXiv:2004.10450*, 2020.

ZHENG, J.; ZHENG, L. A hybrid bidirectional recurrent convolutional neural network attention-based model for text classification. *IEEE Access*, IEEE, v. 7, p. 106673–106685, 2019.

Apêndices

APÊNDICE A – Exemplos de Textos Gerados

Este apêndice apresenta exemplos de textos gerados a partir de sementes do conjunto de teste para todas as arquiteturas e bases de dados utilizadas. A motivação por trás destes exemplos é concretizar as diferenças semânticas proporcionadas pelas TS-GANs em comparação às arquiteturas estado da arte[42]. Ademais, os exemplos também ilustram ocorrências de processos de pós-otimização falhos (incapacidade de convergência) e a sua influência na geração de textos.

A.1 Avaliação de Consumidores da Amazon

Referência: ***although i have found that i must double check that the app is on before i let my children play i find this app to***

RNN: ***although i am not fond of netflix this is a good application to use for it netflix is an odd concept for me and i***

LSTM: *although i love that you can play this game for hours*

GRU: ***although i am not fond of netflix this is a good application to use for it netflix is an odd concept for me and***

RMC: *although i was nt able to*

SeqGAN: *although i am really good at silly time not worth paying for hidden items*

MaliGAN: *although i did not realize there was really going though it did not work on kindle fire hd he does not have a camera when they,*

GSGAN: *although i can move your parts making i mean i m on the song and it works*

RelGAN: *although i have always loved on my laptop*

TS-GAN-C: *although i love mystery hidden object games and i think th is one of the better matching game i*

TS-GAN-T: ***although i am not fond of netflix this is asking if i want to waste my time downloading***

BA: *although i 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0*

CS: *although i hate it was not to easy and takes you back to play maybe you do nt get*

[42] Os exemplos mais próximos às referências foram destacados.

bored

FA: *although i am so tiny but it s not letting me play the game on my kindle fire*

GP: *although i am not fond of netflix i have to delete it then re install*

PSO: *although i am an avid google calendar after each move i had to pay bills money out of the house fun i like the graphics are,*

A.2 Avaliações do Yelp

Referência: ***this bar louie location is always the worst experience the service is just always disappointing clearly no interest in establishing an interaction with customers very robotic and hardly ever come back to the table bringing food out wrong and not***

RNN: *this bar is a cool spot for drinking beers with your buddies and looking over the river at boston but its definitely not a place to go for a good meal or for cocktails we thought the food was not*

LSTM: *this bar is a bit of the best in the area i was in the area and i am so glad we did not a*

GRU: *this bar food is a great spot to stop for a vanilla chai latte i always wouldnt want to sell me a month so i decided to try it out i love this place my husband and i both left a,*

RMC: *this bar is a convenient place to go the place to get my food to go the food is really good and the service is so good the food is okay efficient but*

SeqGAN: *this bar is a unique place for breakfast the breakfast portion is large and pricey the food is not my taste probably i am from the us the food at this place like mix with frenchmoroccan overall this place is packed,*

MaliGAN: *this bar place is a hidden gem literally hidden as you have to walk through a chinese restaurant to get there the place is small cozy and charming inside and the food is better than what you might find in*

GSGAN: *this bar staff is attentive and i always seem to find a seat for lunch or dinner the chicken satay is decent but looks like polynesian warfare and try staying away from the mushroom appetizer reminiscent of a wolfgang pucks concoction,*

RelGAN: *this bar is one of the best spots i have ever eaten at the food has always been good and the service is spot on maybe it is because it is attached to the hotel it has a very*

TS-GAN-C: *this bar as we saw orcas and the hump was the best pizza i have to agree with steve seeing as how i was a pretty cool little place my husband and i went to the closet inspection location to where,*

TS-GAN-T: *this bar from the drinks down to the atmosphere every time ive gone here the staff has always been super friendly and willing to talk through each cocktail the cocktail isnt arent cheap but they are worth*

BA: *this bar got a great chicken tikka masala dont have to go all the way to cambridge anymore also try their chicken tikka pizza its got tikka masala instead of regular sauce chicken onions and peppers really good also i*

CS: *this bar the cocktail i came in for happy hour joints the peopleawesome for that its*

so many years so

FA: *this bar spot me and i eat*

GP: *this bar is in a brilliant location it offers a convenient spot not only for its patrons but also passersby who walk around the mall and in fact some walk by the opposing balcony by the escalator where there is*

PSO: *this bar is not a huge the restaurant that has been great ive been here a handful of times and i have never had a bad meal its the closest thing you can get to north end quality italian without having,*

A.3 Legendas de Imagens COCO

Referência: *a man is playing a fiddle in his kitchen*

RNN: *a man replica with a clock as the receiver helping ready,*

LSTM: *a man toilet with a toilet*

GRU: *a man stove stove with multiple multiple computers or just just,*

RMC: *a man is a toilet with a toilet with a toilet*

SeqGAN: *a man is playing violin in his kitchen*

MaliGAN: *a man in a wheelchair and another sitting on a*

GSGAN: *a man just sits next to his horse in an alley,*

RelGAN: *a man sitting on a street bench by a bike in,*

TS-GAN-C: *a man is playing violin in his kitchen*

TS-GAN-T: *a man sits on a street bench by a bike*

BA: *a man umbrella umbrella umbrella umbrella umbrella umbrella umbrella umbrella umbrella,*

CS: *a man is sitting on a bench next to a*

FA: *a man is playing violin in his kitchen*

GP: *a man riders riders ascending break gravel rain gravel rain gravel,*

PSO: *a man is sitting on a bench next to a*

A.4 Notícias do WMT EMNLP2017

Referência: ***after a tough year in 2014 when i was come back from back surgery i think i m now established again at the top of***

RNN: *after a couple of hours we were told it was safe to come out and we were put back on a plane home within hours*

LSTM: *after a computer finish the world s most powerful nation is just making it*

GRU: *after a certain amount of time i knew i was in big trouble white told good morning america in an exclusive interview*

RMC: *after a viewed in the oval up to the city*

SeqGAN: *after a few days later just after dark soldiers at risk on many black security and environmental protection where he calls for the obama administration has,*

MaliGAN: *after a couple of hours we were told other countries with high smartphone ownership where by the government has since been their president wrote*

GSGAN: *after a couple but now i know if you i m happy*

RelGAN: *after a lot of this to bring year anything and we are a big impact more in going terms*

TS-GAN-C: *after a certain amount of time i can play and i want to go out there and raising everything from the wonder it s*

TS-GAN-T: *after a couple of hours after being taken by looking for about 30 years myself as ever for australia and there was a foreign office*

BA: *after a few days ago more time to a new series characters from the star wars franchise america needs to include tax on the table*

CS: *after a few minutes of trading at the age of 11 times said of the previous year clinton*

FA: *after a 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0,*

GP: *after a certain amount of time in iowa is going to be the best measure of the rest of the world so i like kind of,*

PSO: *after a local young children league more than 48 hour industrial action starting on 26 january and will gather near the stage to the two days,*