

**COORDENADORIA DE ENGENHARIA COMPUTAÇÃO**

**GUILHERME PROENÇA CRAVO DA COSTA**

**RENATO DRUZIAN**

**SUMARIZAÇÃO DE NOTÍCIAS**

**Sorocaba/SP**

**2020**

**Guilherme Proença Cravo da Costa**

**Renato Druzian**

**SUMARIZAÇÃO DE NOTÍCIAS**

Trabalho de conclusão de curso apresentado ao Centro Universitário Facens como exigência parcial para obtenção do diploma de graduação em Engenharia da Computação.

Orientador: Prof. Johannes von Lotcher

**Sorocaba/SP**

**2020**

FICHA CATALOGRAFICA

C837s

Costa, Guilherme.

Sumarização de notícias / por Guilherme Costa, Renato Druzian. – Sorocaba, SP: [s.n.], 2020.

60f.

Trabalho de Conclusão de Curso (Graduação) – Centro Universitário Facens – Curso de Engenharia de Computação, 2020.

Orientador: Prof.(a) Johannes von Lotcher

1. Inteligência Artificial. 2. Sumarização Extrativa. 3. Sumarização Abstrata. I. Druzian, Renato. II. Centro Universitário Facens. III. Título.

CDD 621.39

ELABORADA PELA “BIBLIOTECA FACENS”

**Guilherme Proença Cravo da Costa**

**Renato Druzian**

**SUMARIZAÇÃO DE NOTÍCIAS**

Trabalho de conclusão de curso apresentado ao Centro Universitário Facens como exigência parcial para obtenção do diploma de graduação em Engenharia da Computação.

Orientador: Prof. Johannes von Lotcher

Sorocaba, XX de XXXXXXXXXX de 2020

BANCA EXAMINADORA

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. Esp/Me/Dr e Nome

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. Esp/Me/Dr e Nome

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. Esp/Me/Dr e Nome

**AGRADECIMENTOS**

Frase de um autor que representa o espírito desse trabalho. Frase entre aspas duplas.

Nome do autor

**RESUMO**

**Palavras-Chave**:

**ABSTRACT**

**Key-words**:

**LISTA DE ILUSTRAÇÕES**

[Figura 1. Funcionamento do método](#_Toc40185531) *[CboW](#_Toc40185531)* [19](#_Toc40185531)

[Figura 2. Funcionamento do método](#_Toc40185532) *[Skip Gram](#_Toc40185532)* [2](#_Toc40185532)0

[Figura 3. Representação de grafo direcionado 23](#_Toc40185532)

[Figura 4. Exemplo de multiplicação da coluna com peso de matriz 24](#_Toc40185532)

[Figura 5. Exemplo de resultado com](#_Toc40185532) *[PageRank](#_Toc40185532)* [25](#_Toc40185532)

[Figura 6. Etapas de criação de sumário com](#_Toc40185532) *[PageText](#_Toc40185532)* [26](#_Toc40185532)

[Figura 7. Arquitetura da metodologia](#_Toc40185532) *[Sequence to Sequence](#_Toc40185532)* [28](#_Toc40185532)

[Figura 8. Tipos de Redes Neurais Recorrentes 30](#_Toc40185532)

[Figura 9. Implementação do Web Scraping 33](#_Toc40185532)

**LISTA DE EQUAÇÕES**

**LISTA DE TABELAS**

Tabela 1. Exemplo de vetor *Bag of Words* 17

[Tabela 2. Grafo direcionado transformado em matriz 25](#_Toc41147217)

Tabela 3. Grafo não direcionado transformado em matriz .................... 26

**LISTA DE SIGLAS**

*BoW* *Bag of Words*

*CBoW* *Continous Bag of Word*

*GloVe* *Global Vectors for Word Representations*

*IA Inteligência Artificial*

*Json JavaScript Object Notation*

*[NLP Neuro-linguistic programming](https://en.wikipedia.org/wiki/Recurrent_neural_network)*

*[RNN](https://en.wikipedia.org/wiki/Recurrent_neural_network)* *[Recurrent Neural Network](https://en.wikipedia.org/wiki/Recurrent_neural_network)*

*Seq2eq*  *Sequence-to-sequence*

**SUMÁRIO**

[1 INTRODUÇÃO 13](#_Toc41145484)

[2 REPRESENTAÇÃO DE TEXTO 15](#_Toc41145485)

[2.1](#_Toc41145486) *[Bag of Words](#_Toc41145486)* [15](#_Toc41145486)

[2.2](#_Toc41145487) *[Word Embeddings](#_Toc41145487)* [16](#_Toc41145487)

[2.3](#_Toc41145488) *[Skip Gram](#_Toc41145488)* [18](#_Toc41145488)

[2.4](#_Toc41145489) *[Cbow](#_Toc41145489)* [19](#_Toc41145489)

[2.5](#_Toc41145490) *[GloVe](#_Toc41145490)* [20](#_Toc41145490)

[3 ABORDAGENS DE SUMARIZAÇÃO 21](#_Toc41145491)

[3.1 Sumarização Extrativa 2](#_Toc41145492)1

[3.2 Sumarização Abstrata 2](#_Toc41145492)7

[3.2.1](#_Toc41145492) *[Sequence-to-Sequence](#_Toc41145492)* [28](#_Toc41145492)

[3.2.2 Trabalhos na Literatura 3](#_Toc41145492)0

[4 RESULTADOS 32](#_Toc41145493)

[4.1 Web Scraping 32](#_Toc41145492)

[4.2 Sumarização Extrativa com](#_Toc41145492) *[TextRank](#_Toc41145492)* [34](#_Toc41145492)

[4.3 Sumarização Extrativa com](#_Toc41145492) *[Sequence to Sequence](#_Toc41145492)* [00](#_Toc41145492)

[5 CONCLUSÃO 00](#_Toc41145495)

REFERÊNCIAS.......................................................................................00

# INTRODUÇÃO

O mundo está cada vez mais repleto de informações não-estruturadas, principalmente texto. Mídias sociais, como Twitter e Facebook, tiveram alto crescimento nos últimos anos e influenciam diariamente com opiniões e notícias.

Sites de notícias são fontes provedoras de informações muitas vezes confiáveis, mas o volume de notícias nem sempre é possível de ser acompanhado por uma pessoa ocupada. (Rino & Pardo, 2003) ”...viajar pelas páginas de notícias a fim de apreender o que é essencial exige tempo, capacidade de identificar o que é relevante, no grande volume de informações disponível, e capacidade de mentalizar, de forma coerente, o conteúdo essencial...”.

Máquinas começaram a ser empregadas para realizar tarefas que antes eram das pessoas, como secretarias que resumem notícias financeiras para os patrões ou agentes de *home brokers* geram insights para investidores, que possibilitou a diminuição do tempo de muitos processos.

Em IA, uma das técnicas mais recentes para tratar de sumarização de texto são redes neurais recorrentes. Este método possui aplicações para solucionar algumas análises de sentimento, entidades nomeadas e sumarização de texto.

Primeiramente, para a sumarização de texto, há a coleta das notícias que para utilização como entrada na rede neural, nessa coleta são captados os títulos e os textos das notícias onde o título fica sendo nosso parâmetro de comparação para a saída da rede neural.

Posteriormente, na etapa de teste, que consiste em dividir os dados coletados em treino e teste, esses dois grupos podem ter tamanho que for necessário. A divisão mais comum é 70% para treino e 30% para teste, a rede aprenderá com o grupo de treino e o resultado obtido pela mesma será analisado com o grupo de teste.

A rede neural aprenderá lendo o texto da notícia e fará uma ligação com o título, já na etapa de teste lerá as notícias do grupo que ela não conhece os dados e tentará gerar um título, ao término será realizado uma análise dessa saída com o título original, com isso será feito a análise, de acordo com (Ferneda, 2006) ”As redes neurais artificiais se diferenciam pela sua arquitetura e pela forma como os pesos associados às conexões são ajustados durante o processo de aprendizado”.

Entretanto, deve-se observar que se a maioria das notícias na parte de treinamento possuírem títulos sensacionalistas ou que não condizem com as reais informações apresentadas no texto, será gerado resultados não confiáveis, podendo com isso gerar um algoritmo enviesado, ou seja, que pensa de forma muito parecida com a fonte das notícias.

Para evitar esse problema e garantir melhor assertividade, é necessário um grande volume de dados e de várias fontes, evitando assim criar algum viés na rede neural.

Por fim, a área científica poderá beneficiar-se para melhor inserção de títulos em artigos, assim como resenhas ou textos gerais em instituições acadêmicas, pois a sumarização possibilita que mais textos sejam lidos em menor tempo, sem que haja perda no sentido para que a mensagem seja transmitida.

# REPRESENTAÇÃO DE TEXTO

Texto e palavras como o ser humano compreende, um computador não consegue entender, para isso, é necessário realizar algumas tratativas para que o computador possa entender palavras e textos.

Realizar apenas o pré processamento do texto, não quer dizer que o computador consiga entender ainda, pois pode-se utilizar sinônimos para manter o mesmo contexto da frase ou texto (Kryscinski et. al, 2019) (Luo, Xu, & Guo, 2014).

Outras problematizações encontradas são na análise semântica, de contexto e léxica, esses são relacionados, por exemplo, que robôs humanoides não entenderam sarcasmo.

O campo da inteligência artificial, que estuda a forma como a comunicação e trabalho são realizados para que máquinas possam comunicar-se com as pessoas, é chamado de Processamento de Linguagem Natural, que por muitas vezes é usando na aprendizagem profunda com o uso de redes neurais. Alguns exemplos de uso do PLN são assistentes pessoais e *chatbots*.

## Bag of Words

Este método é o mais simples, mas com a possibilidade de resultados satisfatórios dependendo do vocabulário. Vocabulário é um conjunto de palavras, onde cada uma delas representa uma amostra de palavra contida no texto, e com esse vocabulário é que realiza-se a análise de outros textos.

Uma análise é iniciada com a criação do vocabulário a partir da primeira amostra, então, cria-se um vetor com 0 e 1, que indica se uma palavra do vocabulário está presente no texto analisado ou não, porém dessa forma é perdido a posição dela na frase e o seu contexto também é alterado (Brownlee, 2017).

Para ficar mais claro, abaixo um exemplo com duas frases:

1. A FACENS é nota máxima no Enade
2. A FACENS conseguiu obter nota 5 no Enade

Com essas duas frases pode-se criar o vocabulário, o conjunto fica montado da seguinte forma {A, FACENS, nota, conseguiu, obter, máxima, Enade, 5, é, no}. Agora com o vocabulário monta-se o vetor de 1s e 0s, que fica da seguinte forma:

Tabela 1 – Exemplo de vetor *Bag of Words*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | FACENS | nota | conseguiu | obter | máxima | Enade | 5 | é | no |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |

Fonte: Elaborado pelos autores.

Com essa tabela é possível visualizar como seria uma saída do método *BoW*, onde a primeira linha é representada pelo vocabulário, e cada linha a mais na tabela representa uma frase.

Pode-se observar mais um problema nesse método, quanto maior for o vocabulário criado, maior será a tabela criada, uma vez que o total de linhas sempre será a quantidade de palavras no vocabulário.

## Word Embeddings

Este método é amplamente utilizado nos dias atuais, e consiste em converter cada palavra em um vetor e com isso agrupa palavras similares próximas umas das outras.

Algumas das vantagens deste método são análise semântica e de contexto de uma determinada palavra em um texto, outra forma de análise é a verificação do radical, prefixo e sufixo de uma palavra, com isso é possível encontrar sinónimos para uma palavra.

Para este tipo de processamento são usadas redes neurais para identificar e compreender qual o contexto que a palavra está inserida, e com essa possibilidade é possível resolver problemas como recuperação de informação, classificação de documentos e respostas a perguntas Bolukbasi et al (2016), Kryscinski et al. (2019), Huang et al (2012).

Como todos os métodos anteriores apresentados, esse também possui desvantagem, para que o modelo possa aprender melhor são necessários uma grande quantidade de textos, com isso é possível analisar muitas palavras em diversos contextos e em diferentes posições semântica, quanto maior o vocabulário melhor será o resultado do modelo e maior será o tempo de treino do modelo.

## Skip Gram

Conforme estudo de Chen et al. o *Skip Gram* é um método baseado em vetores, ou seja, cada palavra é transformada em um vetor e com a finalidade de prever quais são as combinações de palavras disponíveis para uso antes e depois da palavra, dado a mesma como entrada.

Assim, é possível analisar o contexto em que a palavra está inserida para encontrar essas combinações. Outro uso deste método é, dado dois vetores e realizar a soma deles, gerar na saída um terceiro vetor com relevância em relação a soma dos dois vetores, como no exemplo abaixo:

Nesse exemplo, realiza-se a conexão das palavras Brasil e Capital pode-se notar que a saída dele é Brasília.

Outro ponto a ser observado é que, ao treinar um modelo com *Skip Gram*, encontra-se uma combinação de palavras que tenha sentido no contexto.

Dado uma sequência de palavras de treino, , ,…,, o *Skip Gram* tem o objetivo de aumentar a probabilidade de uma palavra fazer sentido no contexto de acordo com a Equação 1.

(1)

Onde tem-se que é o tamanho do vocabulário para treino, quanto maior for , maiores são os exemplos de treino, maior o tempo de treinamento e, consequentemente, melhor o resultado do modelo.

A base da fórmula do *Skip Gram* é definida por , que também é escrita como a Equação 2:

(2)

São as representações dos vetores de entradas e saídas dadas por e da palavra , e é o número de palavras no vocabulário, ou seja, o número total de palavras presentes no conjunto de treino.

## Cbow

Como (Luo, Xu, & Guo) disse em seu trabalho, *CBoW* é mais uma técnica de representação de texto que utiliza vetores para simbolizar palavras, esta técnica também faz uso de redes neurais recorrentes para avaliar e prever uma palavra em um determinado contexto.

De uma forma bem simples pode-se dizer que o *CBoW* faz o caminho inverso do *Skip Gram,* onde o último quer, através de uma palavra prever toda a frase, enquanto o *CBoW* possui a frase mas com uma palavra faltando, e o mesmo pretende prever essa palavra faltante.

O uso de redes neurais recorrentes se enquadra, pois as palavra antes são representadas como sendo o passado e as palavra após a lacuna como sendo o futuro, e como lacuna a palavra que deseja-se prever.

Abaixo há duas imagens que mostram de maneira mais clara a diferença entre eles na Figura 1 e Figura 2.

Figura 1 - Funcionamento do método *CBoW*

Uma imagem contendo objeto, relógio

Descrição gerada automaticamente

Fonte: Disponível em: https://arxiv.org. Acesso em 12 mai 2020.

Figura 2 - Funcionamento do método *Skip Gram*

Uma imagem contendo objeto, relógio

Descrição gerada automaticamente

Fonte: Disponível em: https://arxiv.org. Acesso em 12 mai 2020.

## GloVe

Conforme estudos realizados por (Pennington, Socher, & Manning, 2014) e (Misra) o modelo *GloVe*, ou *Global Vectors for Word Representation*, preserva o contexto global e cria uma matriz de co-ocorrência para estimar se uma palavra vai co-ocorrer junto com outra.

Este modelo é utilizado em problemas de sumarização de textos, para responder perguntas, verificação de fraudes, recuperação de informação.

Quanto maior for o volume de dados ou dicionário criado, melhor será sua saída. Por outro lado, não se pode esquecer que quanto maior o volume de dados para treinar o modelo, maior será o tempo que a Rede Neural necessita para finalizar o treinamento.

# ABORDAGENS DE SUMARIZAÇÃO

Para a empresa Sciforce [(2019](https://medium.com/sciforce/towards-automatic-text-summarization-extractive-methods-e8439cd54715?source=post_page-----e8439cd54715----------------------)) sumarizar consiste em um resumo que seja conciso e fluente, e manter o significado geral com as informações-chave do texto. Assim, espera-se que o computador tenha melhor desempenho na tarefa do que um humano por conta da imparcialidade.

Sem perceber, a sumarização de textos está cada vez mais presente nas mais diversas áreas de estudo. No meio acadêmico, os cientistas tendem a ler mais resumos de artigos antes mesmo de ler todo o artigo, uma vez que não encontrem o que procuram, procura-se outras fontes.

Para os vestibulandos, ler os resumos dos livros, ou criar os próprios resumos das disciplinas para estudo, e assimilar melhor o conteúdo apenas com pontos mais importantes dos livros ou o que foi ensinado em sala de aula.

Ao entrar em algum site de notícia, a manchete é lida primeiro e seu rápido resumo, e se houver mais interesse em saber detalhes sobre aquela reportagem, entrar na página para conferir na íntegra a notícia.

Para realizar a tarefa de sumarizar um texto, existem arquiteturas, métodos, classes e algoritmos a serem explorados. Neste capítulo será explicado mais a fundo sobre como funcionam e são implementados.

## Sumarização Extrativa

De acordo com (Pardo, & Rino, 2018) a sumarização extrativa é baseada em métodos estatísticos e/ou empíricos, em que é necessário ter o título do texto para realizar comparações entre as palavras e frases do texto com seu título para obter valores comparativos. É nesse processo, a vetorização de palavras e a técnica de similaridade de cosseno auxiliam a encontrar a proximidade das palavras ou frase do título.

Com a similaridade de cosseno, gera-se valores entre 0 e 1, onde valores próximos de 0 possuem baixa similaridade, e próximos à 1 alta, dessa forma, é possível realizar as comparações e encontrar as frases mais relevantes.

O estudo da empresa (Sciforce, 2019) relata a utilização de frases relevantes no texto para produzir um sumário com subcojunto baseado em frases obtidas do texto original.

Sumarizadores extrativos partem de três princípios de tarefas independentes:

1) Construção de uma representação intermediária do texto de entrada: Há duas formas de abordagem. Representar o texto em formato de tópico para ser interpretado e representado de maneira intermediária, formada com a entrada de dados, e saída depois do processamento. Ou através de indicadores de importância, baseado em uma lista de características de cada frase.

2) Pontuação das sentenças com base na representação: Após a geração de representação intermediária, cada frase recebe um coeficiente entre 0 e 1, que representa maior importância da frase quanto mais próximo e 1 e menor relevância quando mais próximo de 0.

3) Seleção de um resumo composto por uma série de frases: Selecionar um certo número de frases mais importantes para produzir o resumo. Atentar-se para a coerência e minimizar a redundância durante a produção.

Para coletar dados de sites de notícias, a sumarização extrativa possui algoritmos capazes organizar as páginas da *web* em grafos. São conjuntos representados por vértices ou nós, com letras, e arestas, com setas, na Figura, onde o nó da ponta da seta é a saída e o da origem é a entrada representando uma página *web* conectada a outra como, por exemplo, um *link* de  para (Towards Data Science, 2019).

Figura 3 - Representação de grafo direcionado



Fonte: Disponível em: https://towardsdatascience.com. Acesso em 2 set 2020

O algoritmo *PageRank* utiliza grafos, como da Figura 3, chamado de grafo direcionado ou dirigido por causa do fluxo de informações entre os nós serem em uma única direção, para atribuir pesos à páginas da *web e* classificá-las em importância.

Após a construção completa do grafo, o cálculos para atribuir os pesos é dado pela Equação 3.

 (3)

Onde  é o peso de cada página na *web* representado por , um fator de saída de links por , um conjunto de *links* de entrada por , um conjunto e a quantidade de *links* de saída por .

Em termos práticos, para calcular o peso do nó , em destaque na Figura 3, apenas analisando os nós e arestas de  para representá-los em uma matriz e, para simplificar o cálculo, atribuir peso 1 quando houver um *link* entre os nós, de acordo com a ponta da seta, e 0 caso não tenha, descrito na Tabela 2.

Tabela 2 - Grafo direcionado transformado em matriz



Fonte: Disponível em:https://towardsdatascience.com/. Acesso em 2 set 2020

É possível observar um *link* em três casos, ,  e .

Em seguida, cada coluna dessa matriz é normalizada de acordo com  da Equação 3, cada linha é somada e multiplicada com peso de todos os nós. Por enquanto, sem levar em consideração o fator de saída , como mostra a Figura 4, o peso de  é igual a 1,5 após uma iteração, ou seja, realizar o processo uma vez.

Figura 4 - Exemplo de multiplicação da coluna com peso de matriz



Fonte: Disponível em:https://towardsdatascience.com/. Acesso em 2 set 2020

Já se o fator de saída  for igual à 0,85 e 10 iterações forem realizadas, o peso de , de acordo com o *PageText*, é 0,34125. Quanto maior for a quantidade de iterações e o tamanho na matriz, mais custoso e demorado é o processo de cálculo. A Figura 5 mostra o valor de cada nó em cada iteração.

Figura 5 - Exemplo de resultado com *PageRank*



Fonte: Disponível em:https://towardsdatascience.com/. Acesso em 3 set 2020

Apenas uma iteração era necessária para atribuir o peso de , porém em casos mais complexos, as iterações podem variar de valor.

Caso o grafo seja não direcionado, ou seja, sem uma única direção entre um nó e outro, a matriz formada que corresponde ao direcionado fica de acordo com a Tabela 3 e, o cálculo do peso segue as mesmas etapas anteriores.

Tabela 3 - Grafo não direcionado transformado em matriz



Fonte: Disponível em:https://towardsdatascience.com/. Acesso em 2 set 2020.

(Joshi, 2018) Baseado no *PageRank*, o algoritmo *PageText* segue a mesma premissa de classificação, entretanto, pesos são atribuídos à frases de um texto para formar um resumo, também chamado de sumário. Essa análise é feita com quantidade de recomendações da página *web*.

Geralmente, algumas palavras são retiradas dessa análise por não apresentarem relevância significativa no texto, chamadas de *stop words*, como artigos e preposições.

As etapas para criar o sumário partem de 1) uma combinação de textos, 2) separar cada texto em frases individuais, 3) representar as frases em vetores, 4) calcular a similaridade de cada vetor de frase e armazenar em uma matriz, 5) converter a matriz em um grafo não direcionado com as frases sendo nós e as arestas a recomendação e, por fim, 6) as melhores frases classificadas compõem o sumário. A Figura 6 ilustra cada etapa utilizada pelo *PageText*.

Figura 6 - Etapas para criação de sumário com *PageText*



Fonte: Disponível em:<https://www.analyticsvidhya.com/>. Acesso em 2 set 2020

Uma implementação possível é com a linguagem de programação Python, que oferece diversas bibliotecas e pacotes para facilitar algumas etapas.

*Numpy*:auxilia em cálculos numéricos e pode ser aplicada para resolver tarefas como modelos de *Machine Learning*, cálculos matemáticos como

integração e diferenciação, funções de álgebra linear, processamento de imagem e computação gráfica.

*Pandas*: Realiza análises de dados ou estruturas de dados, chamados de *datasets*, de uma forma simples e com alta performance.

*NLTK*: Auxilia em processar dados em Linguagem Natural, ao dividir as frases em *tokens* e separá-las em grupos diferentes como, por exemplo, palavras, números, datas e ponto final e, assim, gerar vetores, e também a remoção de *stopwords*.

*KeyedVectors*: Para carregar modelos pré-treinados de maneira mais rápida como, por exemplo, solicitações para carregar páginas na *web*.

*Punctuation*: Disponibiliza dos os conjuntos relacionados a pontuação de texto, como uma constante de variável do tipo *string*.

*Cosine\_similarity*: Calcula a similaridade de cosseno entre as amostras de um ponto à outro, com base no produto escalar normalizado .

*NetworkX*: Criação e manipulação para estudar estruturas, funções e redes mais elaboradas.

*Json*: Representa estrutura de dados em formato de texto com a sintaxe do objeto *JavaScript*.

Com essas ferramentas e métodos, é possível construir uma estrutura de dados capaz de extrair e sumarizar textos de páginas *web*. No próximo capítulo, será mostrado o algoritmo e resultado para realizar essa tarefa.

## Sumarização Abstrata

No estudo feito por Bing et. al (2016) sumarização abstrata refere-se a tarefa de gerar títulos e resumos, capturando as ideias principais. Chamada de abstrata devido ao potencial de utilizar palavras que não estão presentes no documento original para gerar o título ou resumo.

(NewTechDojo, 2018) A categorização é dividida em três: Aprendizagem Supervisionada baseada em prever dados conhecidos de entrada e saída na etapa de treinamento no processo de aprendizagem, Aprendizagem Não Supervisionada em que apenas os dados de entrada são conhecidos e não há como saber qual será o resultado da saída e Reforço ou Semi-Supervisionado que é uma mescla dos dois anteriores e mais comum em aplicações do mundo real.

## 3.2.1 Sequence to Sequence

(Nikolov, Pfeiffer & Hahnloser, 2018) *Sequence to Sequence* é uma categorização de modelo utilizada em Redes Neurais para processar o texto e gerar na saída o texto totalmente processado e legível para uma pessoa.

Regularmente utilizada em estudos em áreas da *Neuro-linguistic programming* (NLP*)*, principalmente em traduções. *Encoder-Decoder* é uma arquitetura da rede *Seq2Seq* com diversas variações que um grande volume de dados de entrada são ordenados em sequência e codificados para que métodos e algoritmos sejam capazes de classificá-los em algum contexto e, posteriormente, decodificados pela saída, com a tradução ou sumarização legível do texto.

A figura 7, ilustra como arquitetura *Encoder-Decoder* no modelo *Seq2Seq* funciona na sumarização de texto.

Figura 7 – Arquitetura da metodologia *Sequence to Sequence*

**

Fonte: Disponível em: https://github.com/. Acesso em 2 set 2020

Na publicação de Trask (2015) Redes Neurais Recorrentes é um modelo que propagam-se como matriz que definir uma relação entre suas camadas em cada etapa de tempo. Somente a camada oculta baseasse nos dados de entrada. O fluxo de informações e camadas mais simples é descrito abaixo.



Basicamente, a camada oculta tem o efeito de uma memória para armazenar informações da camada anterior.

Em situações mais elaboradas, com apenas uma entrada, é possível criar camadas ocultas a partir da camada de entrada, como no exemplo:



Já se houver mais de uma entrada, criar uma camada de saída com a oculta. Este processo muda o efeito de memória que a camada oculta possui para guardar as informações relevantes recebidas da anterior, já que a combinação entre os dados da camada de entrada com a oculta são de tempos diferentes, como no exemplo:



A Figura 8 ilustra com mais detalhes como funciona cada tipo *RNN* representando cada bloco como um vetor, onde vermelho é a camada de entrada, verde oculta e azul saída.

Figura 8 - Tipos de Redes Neurais Recorrentes



Fonte: Disponível em: http://karpathy.github.io/. Acesso em 30 ago 2020

[(Karpathy](http://karpathy.github.io/), 2015) O tipo de *RNN* *one-to-one* é o mais clássico com uma única camada de entrada, saída e oculta de dados. No *one-to-many* uma camada de entrada gera várias de saídas, e cada camada oculta é processada em um período de tempo com uma saída.

Outros modelos possuem várias camadas entradas como o *many-to-one,* queproduz uma sequência de dados na saída após várias leituras na entrada, utilizado para análises de sentimentos.

Já o *many-to-many* pode funcionar duas maneiras. A primeira que antes de começar a gerar a sequência de saída de imediato, processa alguns períodos de tempo diferentes das saídas, utilizado em casos de tradução de máquina. Já na segunda, o processamento da sequência de entrada e saída pertencem ao mesmo período de tempo.

## 3.2.2 Trabalhos na Literatura

No artigo de (Xiang et. al, 2016) foi proposto aplicar uma *Encoder-Decoder* atencional para melhorar o desempenho da sumarização abstrata com dois *datasets* diferentes. O objetivo era melhorar a forma como uma soma abstrata é feita e solucionar dois problemas específicos. Os resultados foram promissores nos dois *datasets*, e um novo foi cogitado para construir um modelo mais robusto em trabalhos futuros.

Já no artigo de (Hara, Adelson, & Arora, 2017) misturar dois modelos de redes neurais, o *QRNN* e *RNN,* para diminuir perdas ao longo do tempo. Entretanto, apenas o *RNN* isolado teve resultados significativos ao produzir

manchetes. Os títulos gerados com a mescla dos modelos apesar de ter apresentado uma sequência de aprendizagem, não obteve bons resultados na tentativa de diminuir perdas com baixo custo, considerado 'falta de *IA*', e todos os modelos apresentaram potencial para melhoria.

E com os estudos de (Ding, Soricut, & Zhang, 2018) duas arquiteturas de modelos foram criadas, SHAPED e Mix-SHAPED, com a papel de adaptar o estilo para gerar uma sequência de texto. Ambos os modelos tiveram melhor desempenho na etapa de treinamento se comparados aos que não tinham adaptação, ou seja, a saída gera uma média de estilo e, também, a modelos com estilo único.

# RESULTADOS

Para implementar métodos e algoritmos na sumarização de notícias, algumas ferramentas foram utilizadas durante o decorrer do projeto para contribuir na organização e evitar possíveis problemas.

O Google Collaboratory foi usado para compartilhar códigos de testes. Já o Anaconda Navigator versão 1.9.12 para criar e padronizar os ambientes, minimizando o tempo de seção e Internet instável e a linguagem de programação Python versão 3.7 com a importação de bibliotecas auxiliares.

Os códigos feitos foram armazenados na plataforma GitHub para centralizar as versões do projeto e evitar possíveis perdas de conteúdo.

## Web Scraping

Para a coleta de dados, um robô para *Web Scraping* em Python foi implementado com a função de extrair e estruturar dados *web* em forma de vetores de palavras e frases para serem analisados. Neste caso, foram coletadas, aproximadamente, 28000 noticias de 3000 páginas *web* do site G1.

Cada notícia foi coletada a partir da página principal do site e a busca pela categoria. Com isso, os dados são salvos em um arquivo *Json* capaz de reconhecer a gramática *PT-BR* e estruturar a divisão de cada notícia em título, categoria e texto.

Figura 9 - Implementação do *Web Scraping*



Fonte: Elaborado pelos autores

## Sumarização Extrativa com TextRank

Para este método, com os textos carregados em um arquivo *Json,* o *dataset* é dividido em dois vetores, notícias e títulos, com o objetivo de facilitar o pré-processamento e todo o texto é convertido para letra minúscula.

Em sequência, as frases são divididas baseadas no ponto final e a tokenização realizada através do NLTK.

O modelo *GloVe* pré treinado pelo Núcleo Interinstitucional de Linguística Computacional (NILC) da Universidade de São Paulo (USP) foi carregado com a biblioteca *Gensim* utilizando o *KeyedVector* e, posteriormente, a remoção das *stopwords* de acordo com a gramática da língua portuguesa. .

Assim, um dicionário com o método *Word Embbedings* foi criado com as palavras para que variáveis armazenem a média e vetores o valor escalar de cada titulo e notícia.

Por fim, a similaridade de cosseno foi aplicada de acordo com a média em uma matriz de 100 dimensões onde as linhas contem títulos e as colunas notícias, o maior e menor coeficiente encontrado armazenado em um arquivo de texto.

## Sumarização Abstrata com Redes Neurais Recorrentes

# CONCLUSÃO

**REFERÊNCIAS**

ADELSON, P., ARORA, S., & HARA, J. (2017). Abstractive text summarization with Quasi-Recurrent neural netwaorks. p. 7.

Bolukbasi, T., Chang, K.-W., Zour, J., Saligrama, v., & Kalai, A. (2016). Man is to computer programmer as woman is to homemaker? Debiasing embeddings. *30th Conference on Neural Information Processing Systems (NIPS 2016)*, (p. 9). Barcelona.

Brownlee, J. (2017). *Machine Learning Mastery*. Fonte: Machine Learning Mastery: https://machinelearningmastery.com/gentle-introduction-bag-words-model/

Chen, K., Corrado, G., Dean, J., Tomas, M., & Sutskever, I. (s.d.). Distribuited representations of words and phrases and their compositionality., (p. 9).

Duarte, G., Matsumoto, F. & Murakami, L. (2019) *Redes Neurais Recorrentes*. Fonte: Turing Talks: <https://medium.com/turing-talks/turing-talks-26-modelos-de-predi%C3%A7%C3%A3o-redes-neurais-recorrentes-439198e9ecf3>

Ding, N., Soricut, R., & Zang., Y. (2018). Shared-Private Encoder-Decoder for Text Style. Adaptation. Association for Computational Linguistics., (p. 9). New Orleans.

Ferneda, E. (2006). *Redes neurais e sua aplicação em sistemas de recuperação de.* Ribeirão Preto.

Huang, E. H., Socher, R., Manning, C. D., & Ng, A. Y. (2012). Improving word representations via global context and multiple word prototypes. *50th Annual Meeting of the Association for Computational Linguistics.* Jeju: 10.

Kapur, R (2016) Neural Networks & The Backpropagation Algorithm Explained Fonte: Medium: <https://ayearofai.com/rohan-lenny-1-neural-networks-the-backpropagation-algorithm-explained-abf4609d4f9d>

Kryscinski, W., Keshar, N. S., McCAnn, B., Xiong, C., & Socher, R. (2019). Neural text summarization: A critical evaluation., (p. 13).

Luo, Q., Xu, W., & Guo, J. (2014). A study on the CBOW model's overfitting and stability., (p. 4). Shangai.

[Meier](https://arxiv.org/search/cs?searchtype=author&query=Meier,+F), F., [Mujika](https://arxiv.org/search/cs?searchtype=author&query=Mujika,+A), A., & [Steger](https://arxiv.org/search/cs?searchtype=author&query=Steger,+A), A., (2018). Approximating Real-Time Recurrent Learning with Random Kronecker Factors., (p. 3).

NewTechDojo .(2018). *List of Machine Learning Algorithms*. Fonte: NewTechDojo: [https://www.newtechdojo.com/list-machine-learning-algorithms/#Reinforcement%20Learning](https://www.newtechdojo.com/list-machine-learning-algorithms/" \l "Reinforcement Learning)

Paulus, R., Xiong, C., & Socher, R. (2017). A deep reinforced model for abstractive summarization., (p. 12). Palo Alto.

Pardo, T, & Rino, L. (2018). A sumarização automática de textos principais características e metodologias.

Sciforce. (2019). *Towards Automatic Text Summarization Extractive Methods*. Fonte: Medium: <https://medium.com/sciforce/towards-automatic-text-summarization-extractive-methods-e8439cd54715>

Trask, A., (2015). *Anyone Can Learn To Code an LSTM-RNN in Python Part1 RNN*. Fonte: iamtrask: <https://iamtrask.github.io/2015/11/15/anyone-can-code-lstm/>