Como medir concentração de mercado de mídia de forma automatizada: uma experiência em Python[1][2]

Maurício ALVES[3]
Universidade Federal de Pernambuco, Recife, PE

Assíria FLORÊNCIO[4]
Juliano DOMINGUES[5]
Universidade Católica de Pernambuco, Recife, PE

RESUMO

Analisar quantitativamente mercados de mídia é necessário para validar observações empíricas. O artigo apresenta o resultado do desenvolvimento de uma ferramenta automatizada para calcular concentração midiática. São considerados o Índice de Noam (MOCDI) e o Índice Hill (HI) — desenvolvidos especificamente para o mercado de mídia; e a Relação de Concentração (CR) e o Índice Herfindahl-Hirschman (HHI) — para análise de mercados em geral. Os parâmetros de avaliação dos cálculos correspondem à classificação descrita por Domingues (2019). A ferramenta visa democratizar a análise de dados quantitativos nas pesquisas em comunicação.

PALAVRAS-CHAVE: concentração midiática; democracia; análise quantitativa; automação; análise de dados.

[1] Trabalho apresentado no IJ08 - Estudos Interdisciplinares da Comunicação, XIX Encontro dos Grupos de Pesquisas em Comunicação, evento componente do 47º Congresso Brasileiro de Ciências da Comunicação.

[2] Trabalho resultado de Projeto de Pesquisa Nº 467070-JOR-052-2022/1, cadastrado na Universidade Católica de Pernambuco (Unicap).

[3] Estudante de Graduação do 8º semestre do Curso de Sistemas de Informação do Centro de Informática (CIn) da UFPE. E-mail: masj@cin.ufpe.br.

[4] Graduada em Jornalismo pela Escola de Comunicação da Unicap, bolsista PIBIC/CNPq, Projeto de Pesquisa Nº 467070-JOR-052-2022/1. E-mail: assiria.florencio@gmail.com.

[5] Professor-pesquisador da Universidade Católica de Pernambuco (Unicap) e da Universidade de Pernambuco (UPE). Bolsista de produtividade do CNPq (PQ-2). E-mail: juliano.domingues@unicap.br.

Introdução

A análise de dados quantitativos tem ganhado espaço nas ciências humanas. A expertise deixou de ser restrita à estatísticos, matemáticos e cientistas da computação exatamente por ser o ponto de neutralidade buscado para validar evidências empíricas.

Como nunca antes na história da humanidade foi produzida tanta informação quanto atualmente, bancos de dados sobre diversos assuntos podem ser acessados e utilizados com o intuito de que observações qualitativas possam ser comprovadas quantitativamente.

Quando se fala de concentração midiática, apesar de existirem equações que determinem quantitativamente tal nível, quando bases de dados extensas precisam ser estudadas, calcular todas elas uma a uma pode retornar resultados equivocados.

Para preencher esta lacuna e auxiliar aqueles sem tanta afinidade com equações mas que querem desenvolver uma investigação quantitativa de mercado de mídia, um script para cálculo automático em Python foi desenvolvido.

Objetivos

Este artigo tem como objetivo geral apresentar um script desenvolvido para calcular equações de concentração midiática e classificá-las de acordo com padrões descritos por Domingues (2019).

Para isto, fez-se necessário, mais especificamente:

- entender a importância da análise quantitativa da concentração midiática;
- selecionar as equações e desenvolver as funções para o cálculo de concentração midiática; e,
- implementar, no script, a capacidade de leitura e geração de arquivos.

Metodologia

Para alcançar o objetivo geral proposto, que intentou o desenvolvimento de uma ferramenta automatizada para o cálculo de concentração midiática, foram selecionadas

as equações. Estas foram: Índice de Noam (MOCDI) e Índice Hill (HI) — desenvolvidos especificamente para o mercado de mídia; e, Relação de Concentração (CR) e Índice Herfindahl-Hirschman (HHI) — para análise de mercados em geral.

O sistema, criado com o objetivo de calcular as equações de concentração midiática, processa os dados contidos em um arquivo formato CSV e dá como saída um outro arquivo, agora formato TXT. O documento apresenta o resultado dos cálculos e sua devida classificação, isto é, se é baixo, moderado, alto ou altamente concentrado, tal qual detalha Domingues (2019). Abaixo, é descrito o programa.

Na primeira parte, o código desenvolvido importa as bibliotecas que são utilizadas ao longo do script.

```
import csv
import math
import os
from collections import defaultdict
from typing import List, Tuple, Optional, Dict
```

Em seguida, o programa lê as informações do arquivo de entrada e retorna, em estrutura formato dicionário, os dados agrupados por localidade, emissoras e respectiva porcentagem.

```
    @staticmethod
    def read_input_file() -> Optional[Dict[str, List[Tuple[str, float]]]]:
        input_file = None
        for file_name in ('input.csv', 'input.txt'):
            if os.path.isfile(file_name):
               input_file = file_name
                break

        if input_file is None:
            print("Input file not found. Make sure there is an 'input.csv'
or 'input.txt' file.")
            return None

        locations = defaultdict(list)
        try:
            with open(input_file) as file:
```

```
                reader = csv.reader(file)
                for row in reader:
                    try:
                        local, broadcaster, percentage = row
                        if local.lower() == 'local' or percentage.lower()
== 'no data':
                            continue
                        locations[local.upper()].append([broadcaster,
float(percentage)])
                    except ValueError:
                        print(f"Error processing row: {row}. Skipping.")
        except (IOError, csv.Error) as e:
            print(f"Error reading the input file: {e}")
            return None

        for local in locations:
            locations[local] = sorted(locations[local], key=lambda x: x[1],
reverse=True)

        return locations
```

Outra parte do código diz respeito aos cálculos de mensuração de concentração de mercado de mídia. Com base nos dados extraídos do arquivo de entrada, são executados os cálculos e, posteriormente, classificados.

```
       CR3_LIMITS = [(35, 'Low Concentration'), (55, 'Moderate
Concentration'), (float('inf'), 'High Concentration')]
    CR4_LIMITS = [(35, 'No Concentration'), (50, 'Low Concentration'), (65,
'Moderate Concentration'),
                  (75, 'High Concentration'), (float('inf'), 'Very High
Concentration')]
class MediaConcentrationAnalysis:
    CR8_LIMITS = [(45, 'No Concentration'), (70, 'Low Concentration'), (85,
'Moderate Concentration'),
                  (90, 'High Concentration'), (float('inf'), 'Very High
Concentration')]
    HHI_LIMITS = [(1000, 'Not Concentrated'), (1800, 'Moderately
Concentrated'), (float('inf'), 'Highly Concentrated')]
    MOCDI_LIMITS = [(300, 'Not Concentrated'), (500, 'Moderately
Concentrated'), (float('inf'), 'Highly Concentrated')]

    @staticmethod
    def classify(value: float, limits: List[Tuple[float, str]]) -> str:
        for limit, classification in limits:
            if value <= limit:
```

```
                return classification
        return 'Unknown'

    @classmethod
    def CR_CLASSIFICATION(cls, cr_value: float, cr_n: int) -> str:
        if cr_n == 3:
            return cls.classify(cr_value, cls.CR3_LIMITS)
        elif cr_n == 4:
            return cls.classify(cr_value, cls.CR4_LIMITS)
        elif cr_n == 8:
            return cls.classify(cr_value, cls.CR8_LIMITS)
        return 'Unknown'

    @classmethod
    def CR_VALUE(cls, broadcasters: List[Tuple[str, float]], cr_n: int) ->
float:

        return sum(percentage for _, percentage in broadcasters[:cr_n])

    @staticmethod
    def CR(broadcasters: List[Tuple[str, float]]) ->
Optional[List[Tuple[int, float, str]]]:
        qtd_broadcasters = len(broadcasters)
        if qtd_broadcasters < 3:
            return None

        cr_values = []
        for cr_n in (3, 4, 8):
            if cr_n <= qtd_broadcasters:
                cr_value =
MediaConcentrationAnalysis.CR_VALUE(broadcasters, cr_n)
                cr_values.append((cr_n, cr_value,
MediaConcentrationAnalysis.CR_CLASSIFICATION(cr_value, cr_n)))

        return cr_values

    @classmethod
    def HHI_CLASSIFICATION(cls, hhi_value: float) -> str:
        return cls.classify(hhi_value, cls.HHI_LIMITS)

    @staticmethod
    def HHI(broadcasters: List[Tuple[str, float]]) -> Tuple[float, str]:
        hhi_value = sum(percentage ** 2 for _, percentage in broadcasters)
        return hhi_value,
MediaConcentrationAnalysis.HHI_CLASSIFICATION(hhi_value)

    @classmethod
    def MOCDI_CLASSIFICATION(cls, mocdi_value: float) -> str:
```

```
        return cls.classify(mocdi_value, cls.MOCDI_LIMITS)

    @staticmethod
    def MOCDI(broadcasters: List[Tuple[str, float]], hhi_value:
Optional[float] = None) -> Tuple[float, str]:
        if hhi_value is None:
            hhi_value, _ = MediaConcentrationAnalysis.HHI(broadcasters)
        mocdi_value = hhi_value / math.sqrt(len(broadcasters))
        return mocdi_value,
MediaConcentrationAnalysis.MOCDI_CLASSIFICATION(mocdi_value)

    @staticmethod
    def HI(broadcasters: List[Tuple[str, float]]) -> float:
        return sum(math.sqrt(percentage) for _, percentage in broadcasters)
```

Por fim, é criado automaticamente o arquivo de saída em formato TXT com os resultados da análise.

```
    @staticmethod
    def write_output(locations: Dict[str, List[Tuple[str, float]]]):
        try:
            with open('output.txt', 'w', newline='') as file:
                for local, values in locations.items():
                    cr_info = MediaConcentrationAnalysis.CR(values)
                    hhi_value, hhi_classification =
MediaConcentrationAnalysis.HHI(values)
                    mocdi_value, mocdi_classification =
MediaConcentrationAnalysis.MOCDI(values, hhi_value)
                    hi_value = MediaConcentrationAnalysis.HI(values)

                    file.write(f'{local}\n')
                    print(f'{local}')
                    if cr_info:
                        for cr_n, cr_value, cr_classification in cr_info:
                            file.write(f'    CR{cr_n}: {cr_value:.2f} -
{cr_classification}\n')
                            print(f'    CR{cr_n}: {cr_value:.2f} -
{cr_classification}')
                    file.write(f'    HHI: {hhi_value:.2f} -
{hhi_classification}\n')
                    print(f'    HHI: {hhi_value:.2f} -
{hhi_classification}')
                    file.write(f'    MOCDI: {mocdi_value:.2f} -
{mocdi_classification}\n')
                    print(f'    MOCDI: {mocdi_value:.2f} -
```

```
{mocdi_classification}')
                    file.write(f'    HI: {hi_value:.2f}\n')
                    print(f'    HI: {hi_value:.2f}')
        except IOError as e:
            print(f"Error writing the output file: {e}")
```

Contribuições

A pesquisa pretende contribuir para (1) democratizar a análise de dados quantitativos nas ciências humanas, mais especificamente nas pesquisas em comunicação; (2) estimular a utilização de bancos de dados extensos para validar observações qualitativas; e, por fim, (3) aprimorar a precisão na mensuração da concentração midiática.

Neste contexto, pesquisadores com expertises diferentes poderão analisar quantitativamente quaisquer mercados de mídia, não sendo mais limitados pela afinidade com estatística ou programação. A pesquisa visou também a homogeneização da comunidade acadêmica uma vez que, além de tornar a análise quantitativa mais acessível, incentiva a interdisciplinaridade.

REFERÊNCIAS

Alves, Maurício. **Análise de Concentração**. 2024. Disponível em: https://github.com/1-mauricio/media-concentration-analysis. Acesso em: 27 junho 2024.

Domingues, Juliano. Técnicas para medir concentração de mercado de mídia: modo de usar. In: Elen Geraldes (Brasília). Faculdade de Comunicação da Universidade de Brasília (org.). **Resistências e Inovações**: políticas de comunicação em tempos de crise. 2. ed. Brasília: Fac Livros, 2019. Cap. 3. p. 122-138.