Apoena Revista Eletrônica

ESTUDO SOBRE A LEI DE ZIPF E O EFEITO PARETO NAS LINGUAGENS DECOMPUTAÇÃO

Victor Silva Marins de Morais
Arthur Calasans Assis Bispo [1]
José Vicente Cardoso Santos
Gilson Amorim Carvalho [2]

RESUMO

Hodiernamente o avanço das linguagens formais demanda de conhecimentos específicos das suas leis de formação de maneira que se pode considerar a atualização de distintas ferramentas e funções de cada versão das linguagens de programação como linguagem isomorfa para a implementação de adaptações de alfabetos e dialetos para versões mais novas e com diferentes dicionários. Além disso, as linguagens de programação são propensas a paralelos evolutivos e a efeitos similares a técnicas aplicadas em vocabulários comuns, devido às suas propriedades e organização no geral. Com isso, o objetivo geral é evidenciar o padrão de correlação preconizado na Lei de Zipf e no Efeito Pareto; e, osobjetivos específicos são: a) utilizar exemplos das especificações das teorias; b) analisar seus efeitos; c) estabelecer correlações com as principais linguagens de nível assembly (linguagem de máquina), de programação de alto nível e médio nível. d) verificar o scripting; e) teorizar tais situações envolvendo estes experimentos. Para consolidar estes objetivos, adota-se uma metodologia de cunho documental, analítico e histórico, de forma qualitativa onde conclui-seque os níveis de expressão da propriedade nos diferentes escopos de relacionamento possuem similaridades devido à aplicabilidade do conceito em ambientes capazes de ser aleatórios, mas com estruturas e individualidades definidas em cada tipo.

PALAVRAS-CHAVE: Lei de Zipf; Linguagens dePrograma̧ão; Efeito Pareto.

[1] Graduandos do curso Ciências da Computação – Unijorge – Centro Universitário Jorge Amado
[2] professores do curso Ciências da Computação – Unijorge – Centro Universitário Jorge Amado

1. A LEI DE ZIPF E O EFEITO PARETO

Criado em 1935, pelo linguista americano George Kingsley Zipf, este tipo de propriedade pode ser definido em uma simples função: Dada uma frequência "n" de aparição de um termo (análogo a uma moda), "r" como seu ranking na organização em rol, é possível defini-la como $f(r, n) \cong 1/r^n$, e uma das principais características desta expressão reside em sua representação gráfica, com um formato de curva descendente, conhecido como a distribuição de Lei de Potência, muito presente na física para expressar, segundo Stephanie Glen(2015, l. 1)"[...] que a mudança relativa em uma quantidade corresponde a uma mudança relativa proporcional à outra [...]" e sendo este um dos meios mais evidentes de representar o padrão, que pôde ser observado em praticamente todas as linguagens funcionais do mundo, e por isto, se tornou um dos principais mecanismos de conexão entre a estatística e a linguística. Linguagens como Inglês, Português, Alemão, Russo e Japonês já identificaram o princípio em suas estruturas e, até os dias atuais, aspectos como suas ramificações, efeitos, sistematizaçõese principalmente a razão para este tipo de ocorrência são meticulosamente observados e estudados, sem uma clara resposta adquirida (SHUIYUAN et al, 2016).

Um destes aspectos, com uma similar maneira de ser observado no mundo, é o Efeito Pareto, criado pelo consultor Joseph Muran, em homenagem ao economista Vilfredo Pareto. A sua principal regra consiste na propriedade da relação "80/20" sendo expressa nos diversos contextos das várias áreas de atuação do mundo, especialmente nas áreas da matemática, ciência e engenharia de software (BEEVER, Gavin, 2017). Esta relação percebida na maioria das secções de atuação da sociedade, variando desde a matemática econômica, onde 20% dos fundos, se bem utilizados, podem maximizar até 80% dos lucros, até na agronomia, onde, segundo Beever (2017, l. 18), "80% da produção é perdida devido a 2 a cada 10 pragas, então programas de extermínio focam nestas e não em todas as pragas". Uma relação pode ser formada entre ambos os conceitos nas situações onde

os elementos com maior frequência acabam sendo os que causam a maioria das consequências, tornando estes 20% um aspecto mais valioso e estrutural dosmecanismos onde eles residem, e se tornando um tópico de análise muito utilizado para optimização, gerando melhorias em escopos de trabalho e, consequentemente, trazendo mais resultados nas áreas de atuação nas quais ele venha a ser aplicado (BEEVER,2017, l.38).

É necessário salientar, no entanto, que por serem propriedades com principal foco no ramo da estatística, isto acaba por gerar os mesmos problemas os quais possam ocorrer neste campo, ou em similares. Assim como os conceitos de probabilidade, ambas as leis apenas podem funcionar com precisão desejada em grandes escopos, onde quanto mais iterações possam aparecer, melhor o resultado se torna e, portanto, não se trata de uma regra de ouro, nem de uma lei escrita em pedra. Desvios podem acontecer, e para evitá-los, o padrão precisa estar extremamente evidente e nítido, antes de estabelecer qualquer predição (QOCHUK, SHETTY, 2020. l.177).

A partir disso, o objetivo principal do projeto consiste na análise dos chamados “corpora”, definidos por Shetty e Qochuk (2020, l.40) como conjuntos de elementos de linguagem os quais servem de amostra para experimentos, para identificação e ponderação das características essenciais destas propriedades em suas estruturas, através da exploração de conceitos conhecidos sobre o assunto para fundamentar o escopo de análise, em conjunto com a observação do moderno contexto linguístico da computação, dispondo de corpus distintos para cada tipo, como principal objeto de teste, e culminando na investigação das relações entre ambos, dispondo de uma metodologia baseada na investigação dos conceitos e tais relações para uma melhor compreensão e limpeza de código em diferentes tipos de sistemas de computação.

2. ANÁLISES

Ao utilizar estes conceitos como um auxílio na análise das linguagens formais, é possível notar que as estruturas formadas pela junção de um ou mais termos, ministrada por regras definidas, estão diretamente ligadas à importância deste termo, onde autômatos determinísticos com mais estados e funcionalidades derivadas podem possuiruma maior taxa de uso. Além disso, a definição de uma gramática específica para um certo alfabeto, por exemplo, pode obter grande influência no escopo do código analisado. É viável deduzir, em uma situação hipotética, que estados iniciais e suas transições possuem termos mais utilizados do que uma ocorrência de estado final, e diferentes maneiras de interpretar comandos abstratos necessitam de um nível de abstração o qual aproxima ambos os conceitos.

Outro possível fator poderia ser a utilização de inferências e atribuição de valores aos diferentes termos, em um ambiente onde a eficiência é priorizada e a análisesintática dos termos gera um conjunto de regras as quais influenciam o fluxo do texto a ser convertido em comandos, obedecendo a estrutura axiomática expressa em conceitos como a Forma Normal de Chomsky (SIPSER, 2005, p. 131), e estabelecendo distintos métodos de expressão, ao trabalhar com as limitações e possibilidades envolvidas no universo de aplicação de cada linguagem.

Além disso, a boa utilização deste fluxo acaba por gerar sistemas de linguagem mais efetivos, com um menor tamanho e com um uso saudável de estruturas e repetições para cumprir tarefas complexas, representando um conteúdo com um nível de precisão sofisticado para todas as funcionalidades, também permitindo a adaptação destas para novos contextos que possam surgir.

2.2. AS LINGUAGENS DE COMPUTAÇÃO

No quesito de precisão, a computação tem tido uma constante e exponencialmente rápida evolução, por sua influência nas tecnologias de conveniência, facilitando os estudos e fornecendo melhores resultados. Tendo suas origens baseadas no projeto de Ada Lovelace em conjunto com Charles Babbage, o conceito de “programação” e as linguagens as quais viriam a surgir a partir disto puderam auxiliar a humanidade a

áreas de atuação (PACKARD, 2018, l.10).

Um contexto inteiramente novo, o dos softwares foi, criado a partir da organizaçãode conjuntos de informação de maneiras a obedecer comandos, gerando especificações, vistas na evolução da relação a qual o ser humano tem com a máquina, através dos níveis de interação, tendo o nível de "assembly", comumente conhecido como linguagem de baixo nível, representado por sistemas como o FORTRAN e o COBOL, o agora popularmente utilizado alto nível, pois está presente em todos os tipos de Computadores Pessoais, e possui uma miríade de alternativas para obter tais comandos, como o C e o C#, Java, Python, através do advento da programação orientada a objetos, além de uma distinta capacidade de distinção e variedade, como a criação e evolução doSwift pela Apple, (PACKARD, 2005, l.92), como estratégia competitiva, e até outros tipos de linguagens derivados, como as Linguagens de Scripting (HTML5, JavaScript e derivadas), evidenciando ainda mais um dos principais fatores diferenciais destas línguas de outros dialetos comuns, este sendo o da customização. Para todos os tipos de funcionalidade, um comando específico existe com o objetivo de realizá-la, e isso permite a expressão de conceitos abstratos de maneira rápida, conveniente e, principalmente, precisa.

No campo da Ciência da Computação, um dos principais intuitos desta manipulação da informação é o de expressar, e, portanto, conceitos como a matemática podem ser concretizados a partir das expressões, e suas propriedades aplicadas em um meio capaz de armazenar os comandos e replicá-los a um esforço consideravelmente menor, por quanto tempo for permitido. Este tipo de auxílio aumentou a níveis insanos a capacidade de conhecimento a qual pode ser adquirido e moldado nas formas desejadase beneficiar a comunidade de computação presente no mundo a evoluir para novos patamares e explorar novas áreas de tecnologia (SEBESTA, 2011. p. 53) O que antigamente era utilizado para expressar contas básicas e sistematizações, agora é capaz de simular espaços completamente virtuais, expressar funções dos mais diversos tipos, evoluir para contextos além do binário, através da computação quântica e, no geral, contribuir em todos os tipos de atividades, sendo, segundo Packard (2018, l.93), "uma integral parte do funcionamento do mundo por gerações por vir, mesmo possuindo uma origem relativamente antiga". A análise de padrões utilizando este tipo de tecnologia

e em níveis ainda mais complexos.

3. INVESTIGAÇÃO DO CONTEÚDO

Este tipo de relação acabou por revelar diversas particularidades tanto no contexto no qual a lei é aplicada, quanto no processo em si, as quais podem ser evidenciadas em casos ponderando a necessidade do uso dos símbolos, se um algoritmo baseado na distribuição Zipfiana tomasse alguma biblioteca estrutural como seu corpus de análise. Seria pertinente analisar a validade léxica da ocorrência de termos sem símbolos, termos compostos por uma palavra e seu respectivo símbolo, e dos símbolos isolados em si, por possuírem valor semântico e legítimas funções sintáticas (como o ponto poder transformar palavras comuns em palavras compostas, tecnicamente falando, ou como os parênteses podem incluir palavras dentro deles, podendo ter três ou mais tipos de análises em uma sequência de caracteres só), tornando a definição da semântica na computação em si um universo sobreposto em si mesmo e, consequentemente, mais complexa de ser analisada.

Outro tema a ser considerado é o de suas propriedades em relação à complexidade de seus corpora. Por ser um sistema análogo ao da Lei de Potência, e também adaptável ao de elementos matemáticos, em especial o de probabilidade, como evidenciado pela Distribuição Zeta (uma versão mais complexa da Lei de Zipf relacionada ao assunto, e que utiliza o conceito de “densidade de probabilidade”) (GLEN, 2016), algumas de suas propriedades também são evidenciadas, especialmente a de que quanto maior o corpus, mais preciso é o resultado e o padrão é mais fácil de ser percebido, devido ao maior número de iterações existentes. Além destes pontos, existe um fator de imprevisibilidade no meio da computação, pois o elemento das variáveis apresenta-se como um termo o qual pode conter qualquer sequência de caracteres, com qualquer tamanho, a critério apenas de seu criador, o que teoricamente poderia tornar a percepção da lei em experimentos impossível em um contexto no qual constantes atualizações mantém uma contínua reformulação da estrutura trabalhada.

É possível notar, no entanto, que aplicações de modelos como o de G. A. Miller, destacado no artigo do American Journal of Psychology por Davis Howes (1968, p. 269, l. 12), onde:

> *[...]Para dar vida a seu argumento, Miller pede para imaginarmos o proverbial macaco apertando as teclas de uma máquina datilógrafa aleatoriamente. Uma palavra é definida por qualquer sequência de caracteres ligados por espaços. Miller mostra então, por um argumento direto baseado nos cálculos das probabilidades que a distribuição de frequência de palavras seguirá a equação de Mandelbrot para a lei de Zipf, onde ‘ρ(ω) = b{r(ω) + c}*

E através deste modelo, conhecido como “Random-Monkey” model, é possível formular a possibilidade de sistemas completamente caóticos serem, com sucesso, utilizados para a implementação da lei, e consequentemente, ambientes estrutural ou mesmo relativamente organizados, com elementos aleatórios, devem possuir resultados mais concretos, e com menores chances de ocorrerem desvios.

Os níveis em que estes elementos podem estar expressos, pois o tipo de relação entre indivíduo e máquina permite com que diferentes estados de abstração sejam explorados, e o seguinte experimento pode revelar alguns destes aspectos.

Com isso, o objetivo principal deste artigo consiste na identificação das características da lei de Zipf na organização dos termos utilizados em códigos das diferentes linguagens de programação no mundo, com os objetivos específicos sendo a utilização de diferentes corpus compostos por trechos funcionais de cada linguagem como amostra, possibilitando a dedução de propriedades do efeito Pareto em sua estrutura, e verificando o comportamento, sejam semelhanças e diferenças, no nível de assembly, médio e alto, englobando também as linguagens de formatação (subcategoria das linguagens de computação), para gerar uma visão completa do universo trabalhado,e evidenciar pontos de otimização de sintaxe.

A metodologia utilizada consiste em uma exposição de cunho documental, a qual dispõe de exemplos históricos, organizados de maneira qualitativa para melhor definição do escopo observado nos posteriores experimentos analíticos compostos pela coleta de dados, seguida pelo processamento destes e finalizando com uma subsequente disposição gráfica, para uma nítida consolidação das relações entre os conceitos e estabelecimento de suas funcionalidades em meios como o de otimização e perícia contextual em programas e algoritmos.

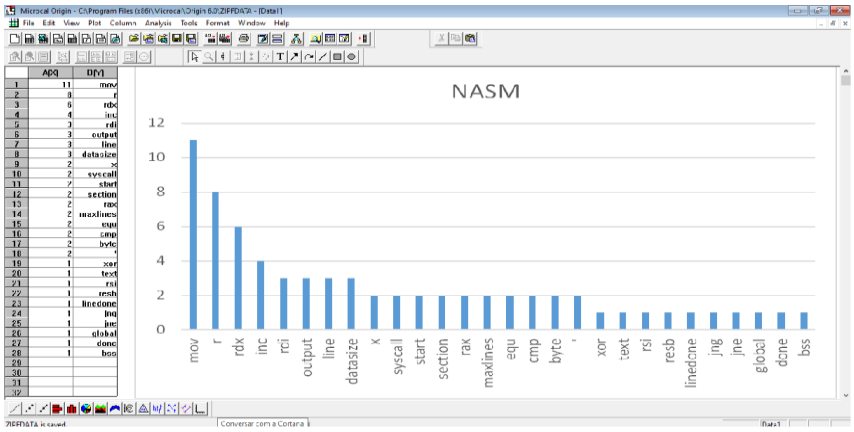
A ferramenta utilizada para analisar a frequência das palavras, ou realizar o "word-counting", é denominada "Word Frequency Counter", desenvolvida por David Bruce, Anna Reynolds e Richard Brown (2002, l.1), e está disponível no site "WriteWords".

A primeira iteração, representada pela linguagem de baixo-nível denominada como NASM, está representada no seguinte fragmento de código (com comentários removidos), desenvolvido por Ray Toal, membro da Loyola Marymount University (2022, l.1), na Califórnia, como parte do tutorial de programação em NASM:

```
        global    start
        section  .text
start:
        mov       rdx, output
        mov       r8, 1
        mov       r9, 0
line:
        mov       byte [rdx], '*'
        inc       rdx
        inc       r9
        cmp       r9, r8
        jne       line
lineDone:
        mov       byte [rdx], 10
        inc       rdx
        inc       r8
        mov       r9, 0
        cmp       r8, maxlines
        jng       line
done:
        mov       rax, 0x02000004
        mov       rdi, 1
        mov       rsi, output
        mov       rdx, dataSize
        syscall
        mov       rax, 0x02000001
        xor       rdi, rdi
        syscall
```

```
        section .bss
maxlines  equ     8
dataSize  equ     44
output:   resb    dataSize
```

Dispondo de 34 linhas, este script possui como função o desenho de um triângulo através da organização da saída de asteriscos na tela do console, e utilizando o word-counter, os resultados são dispostos no seguinte experimento:



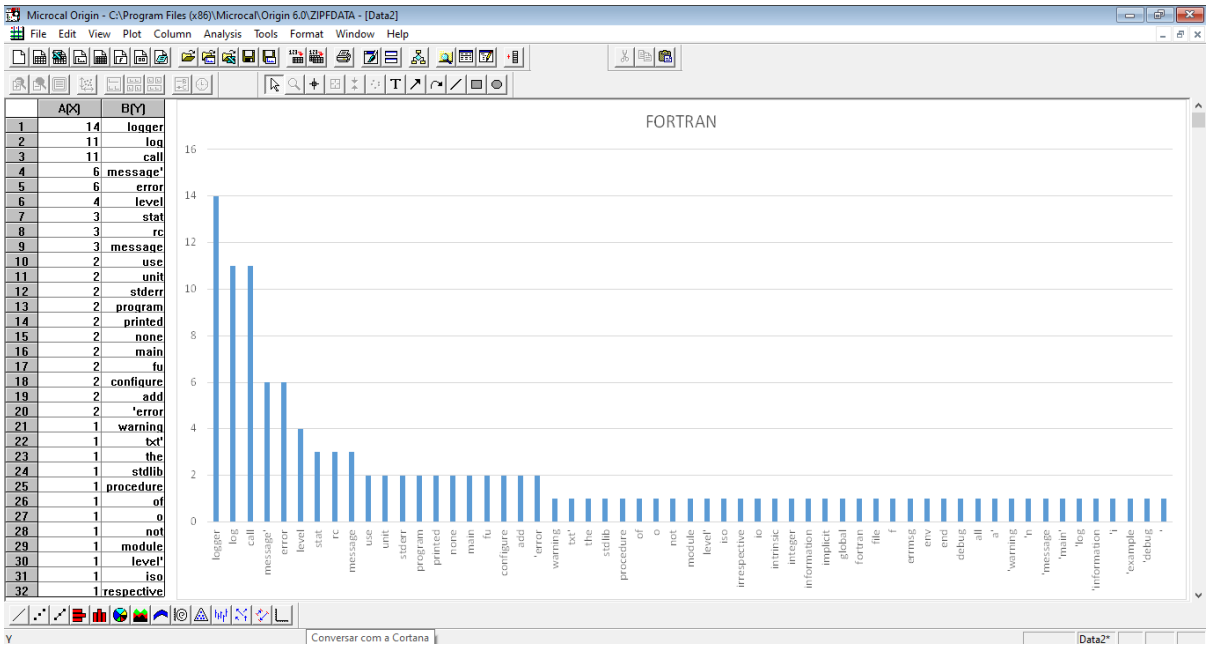
Algumas das características mais no gráfico com maiores peculiaridades, consistem em sua organização aparentemente discreta, sendo menos segmentada no começo, mas com quedas precisamente divididas e que apresentam grupos com crescente quantidade de termos com o mesmo número de aparições, evidenciando uma suposta quantização da maneira como o corpus se comporta, o que seria condizente com a natureza do ambiente onde tal sistema seria aplicado. É possível, notar que as segmentações possuem números que se aproximam do deduzido pela Lei de Zipf (a tendência descendente, com uma notável diferença entre a frequência da mais usadas, mas com um agrupamento mais sucinto, levando-os a uma menor "presença", para a das menos usadas proporcionalmente, com as últimas mais frequentes, mas com menos influência na atuação do código, mesmo que acabem possuindo muito mais espaço e compondo a maior parte do gráfico).

A amostra seguinte apresenta a linguagem estruturada FORTRAN como seu escopo de análise, em um conjunto de funções para, segundo Philipp Engel, criador site e arquivo de armazenamento "Programming in Modern Fortran" (2022, l.46), "prover classes básicas de logging para mostrar no console e no arquivo":

```
!         log.f90
program main
   use, intrinsic :: iso_fortran_env, stderr => error_unit
   use :: stdlib_logger, logger => global_logger
   implicit none
   integer :: fu, rc
```

```
    call logger%add_log_unit(stderr, stat=rc)
    call logger%add_log_file('log.txt', fu, stat=rc)

    call logger%log_error(message  = 'Example error log message', &
                          module   = 'N/A', &
                          procedure = 'MAIN', &
                          stat     = 0, &
                          errmsg   = 'error message')

    call logger%log_message('Message printed irrespective of the level')

    call logger%log_debug('Debug message')
    call logger%log_information('Information message')
    call logger%log_warning('Warning message')
    call logger%log_io_error('I/O error message')

    call logger%configure(level=NONE_LEVEL)
    call logger%log_error('Error message (not printed)')
end program main
```

É possível notar os resultados neste gráfico:



	A[X]	B[Y]
1	14	logger
2	11	log
3	11	call
4	6	message'
5	6	error
6	4	level
7	3	stat
8	3	rc
9	3	message
10	2	use
11	2	unit
12	2	stderr
13	2	program
14	2	printed
15	2	none
16	2	main
17	2	fu
18	2	configure
19	2	add
20	2	'error
21	1	warning
22	1	txt'
23	1	the
24	1	stdlib
25	1	procedure
26	1	of
27	1	o
28	1	not
29	1	module
30	1	level'
31	1	iso
32	1	respective

Nesta outra amostra, cujo exemplo é referente a uma linguagem de médio nível, é possível notar uma semelhança na organização dos termos, incluindo as propriedades definidas anteriormente, algo a ser notado também nas amostras seguintes, e uma característica principal de todos estes experimentos, pois a maioria dos termos utilizados para estrutura e formatação do código acabam por ser usados com mais

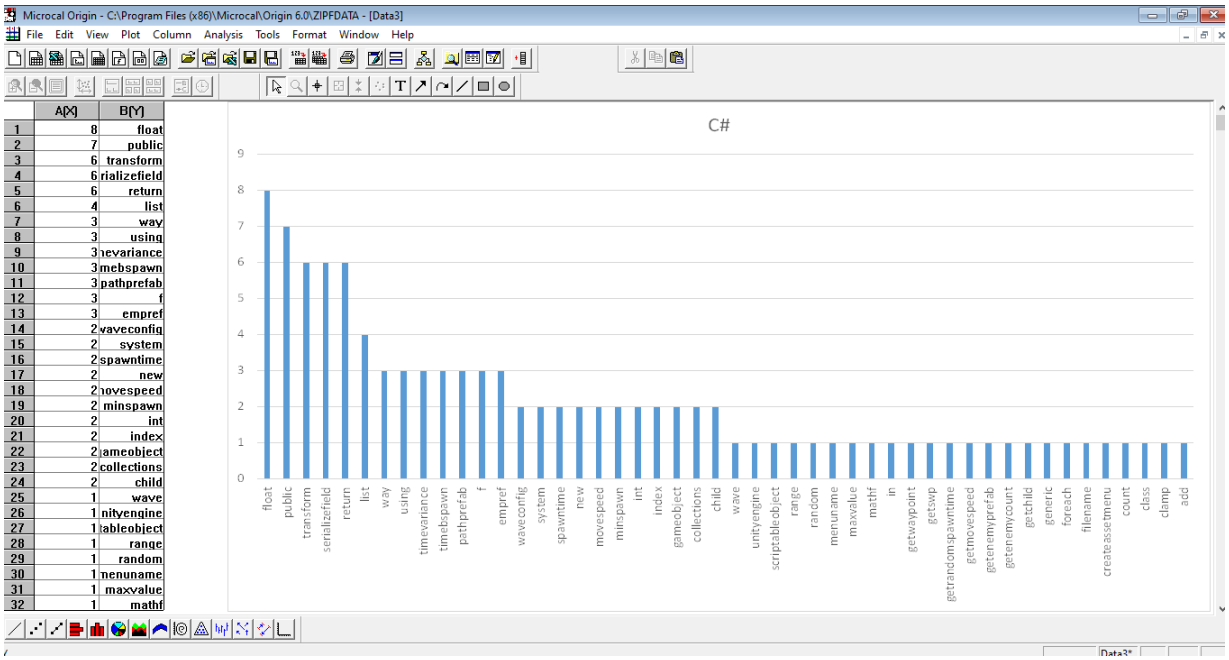
Pareto, e a uma queda considerável a partir dos percentuais que se aproximam dos 20% em cada caso.

A próxima iteração consiste no uso da linguagem de programação orientada a objetos da Microsoft, C#, em conjunto com a Engine de Desenvolvimento de Games conhecida como Unity, propriedade da developer “Unity Technologies”, como parte do curso de Desenvolvimento 2D de desenvolvimento, criado e gerenciado pela equipe GameDev.tv, composta por Ben Tristam, Rick Davidson e Gary Pettie (2022), e confeccionado por Victor Silva Marins de Morais, como atividade de fixação, baseado no modelo deste curso, na qual o seguinte script de 45 linhas foi construído:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

[CreateAssetMenu(menuName = "WaveConfig", fileName = "New Wave")]
public class WaveConfig : ScriptableObject
{
   [SerializeField] List<GameObject> emPref;
   [SerializeField] float timebspawn = 1f;
   [SerializeField] float timevariance = 0;
   [SerializeField] float minSpawn = 0.2f;
   [SerializeField] Transform pathPrefab;
   [SerializeField] float moveSpeed = 5f;

   public Transform GetSWP() {
      return pathPrefab.GetChild(0);
   }

   public List<Transform> GetWaypoint() {
      List<Transform> way = new List<Transform>();
      foreach(Transform child in pathPrefab)
      {
         way.Add(child);
      }
      return way;
   }
   public float GetMoveSpeed() {
      return moveSpeed;
   }

   public int GetEnemyCount(){
      return emPref.Count;
   }
   public GameObject GetEnemyPrefab(int index) {
```

```
    }

    public float GetRandomSpawnTime(){
        float spawnTime = Random.Range(timebspawn - timevariance,
        timebspawn + timevariance);
        return Mathf.Clamp(spawnTime, minSpawn, float.MaxValue);
    }
}.
```

Este script representa um componente de configuração de aparecimento de objetos na tela, denominados como inimigos por outros sistemas da engine, agrupados em um filamento conhecido como “Wave”, onde uma sequência é criada na tela e, posteriormente, destruída, tendo propriedades como o intervalo de tempo das ondas definidas simultaneamente (PETTIE, 2022). Ao ser apresentado e contado, o código teve seus resultados dispostos na amostra abaixo:



Nesta amostra, percebe-se a maior presença das palavras agrupadas, devido ao uso mais aprofundado de contexto e estrutura mais flexível que linguagens de alto nível e, principalmente, orientadas a objetos tendem a apresentar.

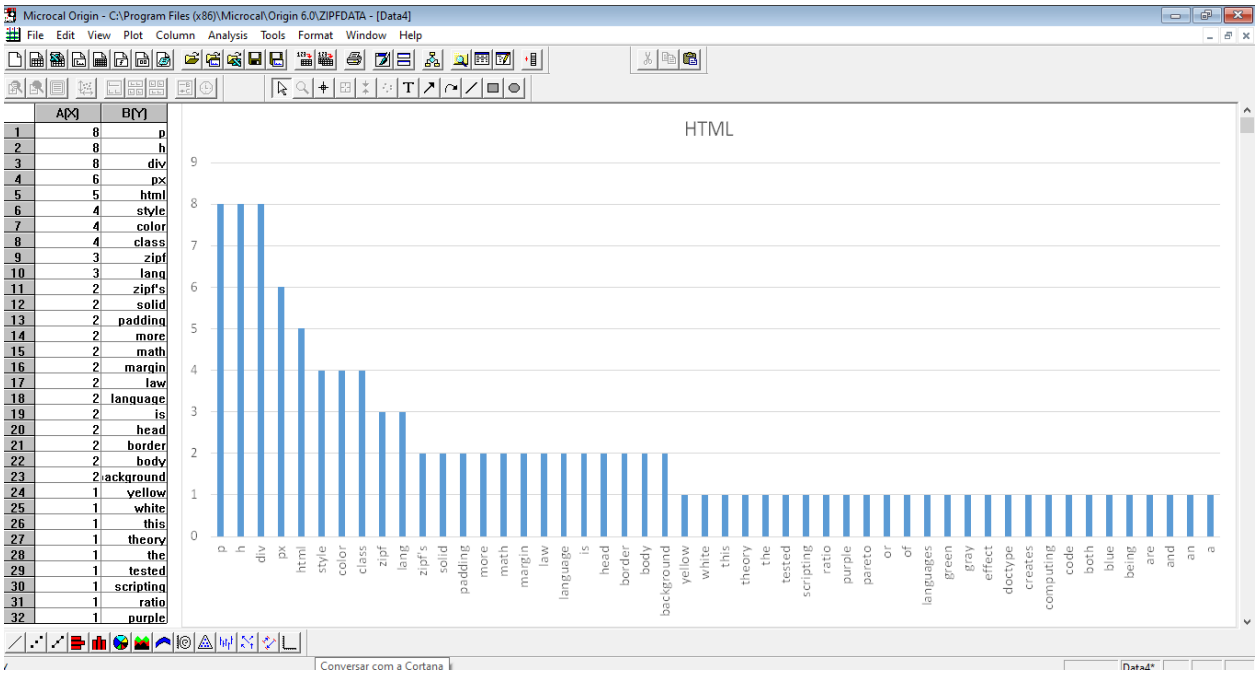
E, por fim, esta iteração utiliza a linguagem de scripting em front-end conhecida como HTML, com o exemplo escrito por Victor Silva Marins de Morais utilizando o TryIt Editor, desenvolvido pela W3Schools (2022, l1), e baseado no modelo de código escrito por esta mesma, através do seguinte script:

```
<!DOCTYPE html>
```

```
<head>
<style>
.zipf {
  background-color: green;
  color: white;
  border: 5px solid blue;
  margin: 40px;
  padding: 20px;
}
</style>
<style>.lang {
          background-color: yellow;
          color: purple;
          border: 5px solid gray;
          margin: 30px;
          padding: 10px;
          }
</style>
</head>
<body>

<div class="zipf">
  <h2>Zipf's Law</h2>
  <p>Zipf's Law is a theory of
  both math and language.</p>
</div>

<div class="zipf">
  <h2>80/20</h2>
  <p>The Pareto Effect creates an 80/20 ratio.</p>
</div>

<div class="lang">
  <h2>Html</h2>
  <p>This html code is being tested.</p>
</div>

<div class="lang">
   <h2>Scripting</h2>
   <p>Are computing languages more math or more language?</p>
</div>

</body>
</html>.
```

A função desse script consiste na definição de classes para armazenamento de texto em objetos, dispondo-os em forma de mensagem na tela, com cada mensagem disposta

e, novamente, os termos foram contados, e os resultados dispostos nestes dados:



Já nesta última análise, a discrepância nas posições das frequências dos termos é notavelmente mais branda e possui algumas quebras ao serem gerados grupos de palavras com frequências similares nas palavras mais realizadas, algo não notado nos outros exemplos, e que mostra uma certa individualidade no modo de atuação desta linguagem.

4. CONCLUSÃO

A partir dos dados coletados e apresentados no experimento, em conjunto com as informações fornecidas pelo estudo literário, a principal resolução do assunto reside na capacidade de optimização destes tipos de linguagens, mesmo considerando o escopo relativamente reduzido, por se tratarem de fragmentos e argumentos existentes em um universo relativamente estruturado. Uma das principais características observáveis na expressão da Lei de Zipf em cada linguagem é a diferente taxa de usabilidade dos termos, onde os menos utilizados e que possuem menos efeito na construção de código acabam por possuírem a maior probabilidade de se tornarem obsoletos e serem removidos em alguma atualização, por serem mais instáveis. Vale ressaltar também a probabilidade de grandes violações de segurança ou outros tipos de quebra e exploração

utilizadas se aplicam, que são mais enraizadas na estrutura e, portanto, são as que devem possuir mais suporte preventivo.

Outra característica visualizada foi a existência de diferentes níveis de solidificação na distribuição das palavras de cada elemento, onde na linguagem mais orientada para a máquina, o resultado acabou sendo levemente mais rígido, pois as organizações possuem poucas nuances, com contextos criados de maneira mais rústica, e nem a utilização de referenciais metalinguísticos, e os atributos semelhantes entre cada uma residem no uso de termos similares e paralelos funcionais, pois o tipo de evolução baseado em iterações das classes permitiu que um tipo de paralelismo permanecesse tanto nas amostras quanto em seus respectivos materiais de origem.

Com as informações fornecidas nas análises realizadas, é possível concluir que a notabilidade na diferença modal entre os termos iniciais e os finais pode indicar uma possível tendência à padronização, e tornar mais manipulável e consistente a adaptação do conteúdo para formatos mais eficientes, pois, ao serem devidamente indicados os pontos de otimização em um contexto determinado, alterações podem ter mais impacto com menos esforço, gerando resultados mais constantes e aumentando significativamente a produtividade de trabalho no meio computacional, além de criar programas mais efetivos e com menores margens de erro, tornando a conveniência uma ferramenta útil e prioritária nas diferentes funcionalidades do mundo.

REFERÊNCIAS

QOCHUCK, Benjamin W. SHETTY, Ashvith. **Zipf's Law in NLP.** OpenGenusIQ, 2020. Disponível em: https://iq.opengenus.org/zipfs-law/. Acesso em: 30/09/2022.

YU, Shuiyuan. XU, Chunshan. LIU, Haitao. **Zipf's Law in 50 Languages:** It's structural pattern, linguistic intepretation, and cognitive motivation. Arxiv.org, 2016.

30/09/2022.

PACKARD, Hewlett. **Computer History:** A Timeline of Computer Programming Languages. HP.com, 2018. Disponível em: https://www.hp.com/us-en/shop/tech-takes/computer-history-programming-languages#:~:text=1883%3A%20The%20first%20programming%20language,just%20numerical%20values%20of%20things. Acesso em: 30/09/2022.

GLEN, Stephanie. **Zeta Distribution (Zipf Distribution).** Statistics How To, 2016. Disponível em: https://www.statisticshowto.com/zeta-distribution-zipf/. Acesso em: 30/09/2022.

BEEVER, Gavin. **Pareto's Law**. AgriFutures Australia, 2017. Disponível em: https://extensionaus.com.au/extension-practice/paretos-law/#:~:text=The%20value%20of%20the%20Pareto,plants%20in%20the%20farming%20system. Acesso em: 30/09/2022.

HOWES, Davis. **Zipf's Law and Miller's Random Monkey Model.** The American Journal of Psychology, 1968. p. 269-272. Disponível em: https://www.jstor.org/stable/1421275. Acesso em: 30/09/2022.

DAVIDSON, Rick et al. **Complete C# Unity Game Developer 2D.** Udemy, 2022. Disponível em:https://www.udemy.com/course/unitycourse/learn/lecture/1788536?start=0#questions. Acesso em: 17/10/2022.

ENGEL, Philipp. **Fortran Standard Library.** Programming in Modern Fortran, 2022. Disponível em: https://cyber.dabamos.de/programming/modernfortran/fortran-standard-library.html. Acesso em 17/10/2022.

TOAL, Ray. **NASM Tutorial.** Loyola Marymount University, 2022. Disponível em: https://cs.lmu.edu/~ray/notes/nasmtutorial/. Acesso em: 17/10/2022.

W3Schools. **HTML Classes – The Class Attribute.** Refsnes Data, 1999. Disponívelem: https://www.w3schools.com/html/html_classes.asp. Acesso em: 20/10/2022.

https://www.w3schools.com/html/tryit.asp?filename=tryhtml_classes_capitals. Acessoem: 20/10/2022.

BRUCE, David et al. **Word Frequency Counter.** WriteWords.co.uk, 2002. Disponívelem: http://www.writewords.org.uk/word_count.asp. Acesso em: 20/10/2022.

SIPSER, Michael. **Introdução à Teoria da Computação.** 2ª Edição. Cengage CTP, 2005. 479 p.

MENEZES, Paulo Blauth. **Linguagens Formais e Autômatos.** 6ª Edição. Porto Alegre.Bookman, 2011. 256 p.

RAMOS, Marcos Vinícius Midena. **Linguagens Formais e Autômatos.** Sem Editora, 2022. 384 p.

HOPCROFT, John E. et al. **Introdução à Teoria de Autômatos, Linguagens e Computação.** Tradução: Editora Campus. 2ª Edição. Santa Catarina. UESC, 2000. 177 p.

SEBESTA. Robert W. **Introdução à Teoria da Computação**. 9ª Edição. Porto Alegre. Bookman, 2011. 795 p.