

Recuperação de Informação 2022.1

Jefferson Rodrigues

Ciência da computação
jrm@cin.ufpe.br

Igor Eduardo

Ciência da computação
iem@cin.ufpe.br

Victor Vasconcelos

Ciência da computação
vvb2@cin.ufpe.br

Domínio: Receitas de Bolo

Sites Escolhidos:

<https://www.tudogostoso.com.br/>

<https://www.comidaereceitas.com.br/>

<https://receitaculinaria.com.br/>

<https://receitas.globo.com/>

<https://vovopalmirinha.com.br/>

<https://www.receiteria.com.br/>

<https://www.receitasnestle.com.br/>

<https://cybercook.com.br/>

<https://www.panelinha.com.br/>

<https://br.recepedia.com/>

Crawler

Victor Borges

Ciência da computação

ivvb2@cin.ufpe.br

Crawler:

```
def decodeSite(url):
    if url == "#primary":
        return ""
    try:
        ua = UserAgent()
        header = {'User-Agent': str(ua.chrome)}
        retorno = ""
        for t in range(tentativasSite):
            result = req.get(url, headers=header)
            if result.status_code == 200:
                retorno = result.content.decode()
                break
        return retorno
    except:
        return ""

def getSoup(url):
    return BeautifulSoup(decodeSite(url), "html.parser")

def salvarArquivo(url, dominio, texto):
    guid = str(uuid.uuid4())
    path = os.path.join(config['PastaRaiz'], "Crawler", dominio)

    bancoSite.append((guid, url))
    f = open(os.path.join(path, guid + ".txt"), "w", encoding='utf8')
    f.write(str(texto.encode('utf-8')))
    f.close()
```

O Crawler decodifica a página até encontrar o status 200.

Depois do decoder, ele vai para o salvar arquivo onde gera o Guid e Path para salvar o arquivo na pasta do seu domínio.

Crawling:

```
def navegacao2(paginapai,pagina):  
    if paginapai == pagina or pagina == "#primary" or len(bancoSite) > 100 or existeUrl(pagina):  
        return  
  
    paginaText = decodeSite(pagina)  
    salvarArquivo(pagina,dominio,paginaText)  
  
    print(str(len(bancoSite)) + " - " + paginapai + " - " + pagina)  
  
    soup1 = BeautifulSoup(paginaText, "html.parser")  
    listAncora1 = soup1.find_all('a', href=True)  
  
    for site in listAncora1:  
        url = site["href"]  
        if pegarDominio(url) == pegarDominio(paginapai):  
            navegacao2(pagina,url)
```

Aqui o Crawler pega as âncoras que foram salvas e faz a recursão baixando os sites Positivos.

Crawler:

Dificuldades:

- Links que levam para fora do site (aplicativo, redes sociais, loja, etc) - Sites com alta “indisposição” (verificação de robôs) a serem minerados
- Links com prefixo
- Links com redirecionadores
- Especificidades de links de determinadas páginas
- Parâmetros causando comportamento obsessivo na busca heurística

Classificador

Igor Eduardo

Ciência da computação

iem@cin.ufpe.br

Treino e teste de rótulos

- Usamos **10 links positivos e 10 negativos rotulados previamente** de cada site para avaliar os classificadores;
- Um total de 200 HTMLs;
- As **páginas selecionadas como exemplos positivos** são paginas que são **claramente uma receita de bolo**.
- As páginas selecionadas como **exemplos negativos** são paginas que **não são uma receita de bolo** , porém ainda podem ser outros tipos de receitas ou paginas dos sites escolhidos que não são receitas como dicas de culinaria, blog, loja, etc.

TF-IDF:

O TF-IDF significa frequência de termos - documentos inversos de frequência.

TF-IDF é na verdade a combinação de duas métricas a tf e a idf com o intuito de se obter um peso frequentemente usado nas áreas de recuperação de informações, mineração de texto, processamento de linguagem natural, sistemas de recomendação , etc. E esse peso é uma medida estatística normalmente usada para avaliar a importância de uma palavra para um documento a partir de uma coleção de documentos, chamada de corpus.

TF:

O TF:

Frequência de termo, que mede com que frequência um termo ocorre em um documento.

Como cada documento tem um comprimento diferente , é possível que um termo apareça muito mais vezes em documentos longos do que em documentos mais curtos.

Por isso a frequência de um termo é frequentemente dividida pelo comprimento do documento, como forma de normalização.

supondo que um documento tenha 20 palavras e 5 delas é a palavra "açucar". o TF será calculado como: $TF: 5/20 = 0.25$

TF:

```
def pegaTFPalavrasDoc(path):
    dicTF = dict()
    f = open(path, mode="r", encoding="utf-8")
    soup = BeautifulSoup(f.read())

    for script in soup(["script", "style"]):
        script.extract()

    text = soup.get_text()
    lines = (line.strip() for line in text.splitlines())
    chunks = (phrase.strip() for line in lines for phrase in line.split(" "))
    text = '\n'.join(chunk for chunk in chunks if chunk)

    listText = text.split("\n")

    for linha in listText:
        for palavra in linha.split(" "):
            palavra = palavra.lower()
            if dicTF.get(palavra) is None:
                dicTF[palavra] = 1
            else:
                dicTF[palavra] = dicTF[palavra] + 1
    return dicTF
```

```
def qtdOcorrencia(listaDic,palavra):
    qtdOcorrencia = 0
    for dic in listaDic:
        if not(dic.get(palavra) is None):
            qtdOcorrencia = qtdOcorrencia + 1
    return qtdOcorrencia
```

IDF:

Precisamos ponderar os termos frequentes enquanto aumentamos os termos raros.

Devido a isso o IDF é calculado como logaritmo do numero dos documentos no corpus dividido pelo numero de documentos onde o termo especifico aparece.

suponha que temos **5** documentos no total e a palavra "açucar" aparece em **2** deles o IDF será calculado como: IDF: **$\log(5/2)=0.398$**

Portanto, o tf-idf da palavra açúcar será o produto dessas duas metricas.

$$0,25(tf) \times (idf) 0,398 = 0.0995$$

IDF:

```
def computaIDF(palavra, tipo):  
    if(tipo == "positivo"):  
        qtdOcorrenciaPositiva = qtdOcorrencia(listaTFDicpositivo, palavra)  
        if qtdOcorrenciaPositiva is None or qtdOcorrenciaPositiva == 0:  
            return 0  
        return math.log10(200 / qtdOcorrenciaPositiva)  
    else:  
        qtdOcorrenciaNegativa = qtdOcorrencia(listaTFDicNegativo, palavra)  
        if qtdOcorrenciaNegativa is None or qtdOcorrenciaNegativa == 0:  
            return 0  
        return math.log10(200 / qtdOcorrenciaNegativa)
```

Classificador:

Dificuldades:

- Muitos dos htmls baixados estavam muitos "sujos", o tratamento dos dados foram uma das dificuldades.

```
for key in dictPath:
    for file in os.listdir(dictPath[key]):
        textoDict = dict()
        fullpath = os.path.join(dictPath[key],file)
        f = open(fullpath, mode="r", encoding="utf-8")
        soup = BeautifulSoup(f.read())
        for script in soup(["script", "style"]):
            script.extract()

        text = soup.get_text()
        lines = (line.strip() for line in text.splitlines())
        chunks = (phrase.strip() for line in lines for phrase in line.split(" "))
        text = '\n'.join(chunk for chunk in chunks if chunk)

        listText = text.split("\n")

        for linha in listText:
            for palavra in linha.split(" "):
                palavra = palavra.lower()
                if textoDict.get(palavra) is None :
                    textoDict[palavra] = 1
                else:
                    textoDict[palavra] = textoDict[palavra] + 1

        if key == "positivo":
            listaTFDicpositivo.append(textoDict)
        else:
            listaTFDicNegativo.append(textoDict)
```

Extrator

Jefferson Rodrigues

Ciência da computação

jrm@cin.ufpe.br

Alguns Exemplos de Extratores Abaixo

Extrator:(comida e receitas)

```
16 path = "C:\\Users\\jefferson-pc\\Documents\\ProjetoReceitas\\Classificador\\comidasereceitas"
17
18 def criaCsvInformacoes(informacoes,path):
19     with open(path, mode="w", encoding="utf-8") as csvfile:
20         fieldnames = ["NomeReceita", "Ingredientes", "Passos"]
21         writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
22         writer.writeheader()
23         for informacao in informacoes:
24             writer.writerow({"NomeReceita": informacao[0], "Ingredientes": informacao[1], "Passos": informacao[2]})
25
```

```
26 for file in os.listdir(path):
27     fullPaht = os.path.join(path,file)
28     f = open(fullPaht, mode="r", encoding="utf-8")
29     soup = BeautifulSoup(f.read())
30
31     ingredientes = soup.find("p").find_next_siblings("ul")
32
33     textoIngredientes = " ".join(ingrediente.get_text() for ingrediente in ingredientes)
34
35     modoPreparo = soup.find("p",{"class":"no-print"}).find_next_siblings("ul")[0]
36
37     textoModoPreparo = " ".join(modoPreparo.get_text() for modoPreparo in modoPreparo)
38
39     nomeReceita = soup.find("h1").get_text()
40
41     informacoes = [(nomeReceita, textoIngredientes, textoModoPreparo)]
42
43     print(informacoes)
44     file= "informacoes"
45     criaCsvInformacoes(informacoes, "C:\\Users\\jefferson-pc\\Documents\\ProjetoReceitas\\Classificador\\comidasereceitas\\" + file + ".csv")
46
47     f.close()
```

Extrator:(cybercook)

```
16 path = "C:\\Users\\jefferson-pc\\Documents\\ProjetoReceitas\\Classificador\\cybercook"
17
18 def criaCsvInformacoes(informacoes,path):
19     with open(path, mode="w", encoding="utf-8") as csvfile:
20         fieldnames = ["NomeReceita", "Ingredientes", "Passos"]
21         writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
22         writer.writeheader()
23         for informacao in informacoes:
24             writer.writerow({"NomeReceita": informacao[0], "Ingredientes": informacao[1], "Passos": informacao[2]})
25
```

```
26 for file in os.listdir(path):
27     fullPaht = os.path.join(path,file)
28     f = open(fullPaht, mode="r", encoding="utf-8")
29     soup = BeautifulSoup(f.read())
30
31     ingredientes = soup.find("div",{"class":"mt-4"}).find_all("p")
32
33     textoIngredientes = " ".join(ingrediente.get_text() for ingrediente in ingredientes)
34
35     modoPreparo = soup.find("div",{"class":"mt-0"}).find_all("p")
36
37     textoModoPreparo = " ".join(modoPreparo.get_text() for modoPreparo in modoPreparo)
38
39     nomeReceita = soup.find("h1").get_text()
40
41     informacoes = [(nomeReceita, textoIngredientes, textoModoPreparo)]
42
```

Extrator:(receiteria)

```
16 path = "C:\\Users\\jefferson-pc\\Documents\\ProjetoReceitas\\Classificador\\Receiteria"
17
18
19 def criaCsvInformacoes(informacoes,path):
20     with open(path, mode="w", encoding="utf-8") as csvfile:
21         fieldnames = ["NomeReceita", "Ingredientes", "Passos"]
22         writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
23         writer.writeheader()
24         for informacao in informacoes:
25             writer.writerow({"NomeReceita": informacao[0], "Ingredientes": informacao[1], "Passos": informacao[2]})
26
```

```
27 for file in os.listdir(path):
28     fullPaht = os.path.join(path,file)
29     f = open(fullPaht, mode="r", encoding="utf-8")
30     soup = BeautifulSoup(f.read())
31
32
33     ingredientes = soup.find("h2").find_next_siblings("ul")[0].find_all("li")
34
35     textoIngredientes = " ".join(ingrediente.get_text() for ingrediente in ingredientes)
36     print(textoIngredientes)
37
38     passos = soup.find("h2").find_next_siblings("ol")[0].find_all("li")
39     textoPassos = " ".join(passo.get_text() for passo in passos)
40     print(textoPassos)
41
42     nomeReceita = soup.find("h1").get_text()
43
44     informacoes = [(nomeReceita, textoIngredientes, textoPassos)]
45
46     file= "informacoes"
47     criaCsvInformacoes(informacoes, "C:\\Users\\jefferson-pc\\Documents\\ProjetoReceitas\\Classificador\\Receiteria\\" + file + ".csv")
48
```

Extrator:

Dificuldades:

- Fazer um extrator pra cada site

▼ Extrator

comidasEreceitas.py

cybercook.py

extratorCompleto.py

ExtratorReceiteria.py

globo.py

panelinha.py

receitaculinaria.py

Fazer um extrator que funcione para todos os sites é difícil, pois cada um tem sua particularidade