ENGENHO DE BUSCA

CRAWLER , CLASSIFICADOR E EXTRATOR

EQUIPE: JEFFSON SIMÕES E LUIZ REIS

BIBLIOTECAS

- PANDAS
- RE
- URLLIB
- TIME
- NLTK
- glob
- BEAUTIFULSOUP
- REQUESTS
- SELENIUM
- REPPY
- OPERATOR
- sklearn
- Numpy

PROBLEMAS

- **SITE DINÂMICO**
  - CARREGAMENTO DE SITE COM SCRIPT.
- **SOLUÇÃO:**
  - USAR SELENIUM PARA SIMULAR WEBCLIENT

    E ASSIM CONSEGUIR O HTML COM AS INFORMAÇÕES.

```
def CrawlerDinâmico(url):
    #C:\\Users\\Usuário\\Downloads\\chromedriver76\\chromedriver.exe
    driver = webdriver.Chrome("C:\\Users\\Usuário\\Downloads\\chromedriver76\\chromedriver.exe")
    driver.get(url)
    res = driver.execute_script("return document.documentElement.outerHTML")
    driver.quit
    return res
```

PROBLEMAS

- **URLS INCOMPLETAS**
  - URLS : ABSOLUTAS E RELATIVAS

- **SOLUÇÃO:**
  - USAMOS URLLIB PARA TRATARMOS OS CASOS RELATIVOS.

```
# Correção de href relativos para absolutos.
href = urllib.parse.urljoin(url.geturl(),href)
```

PROBLEMAS

- **ROBOTS.TXT**
  - PARTES DO SITE QUE NÃO PODEMOS CRAWLEAR.

- **SOLUÇÃO:**
  - USAMOS REPPY QUE POSSUI UMA FUNÇÃO QUE RETORNA TRUE PARA CRAWLEAR AQUELA PARTE DO SITE.

```
#Função para Verificar se podemos baixar a page ou não
def verificarRobotTxt(url):
    robots = Robots.fetch(url + '/robots.txt')
    return robots.allowed(url, '*')
```

PROBLEMAS

- **REQUISIÇÕES:**
  - MUITAS REQUISIÇÕES FEITAS AO MESMO SITE PODEM PREJUDICA-LO.

- **SOLUÇÃO:**
  - USAMOS A BIBLIOTECA TIME PARA DAR UM DELAY ENTRE AS REQUISIÇÕES FEITAS..

```
#esperar 500 milisegundos para visitar o prox link.
time.sleep(.5)
```

CRAWLER - CONDIÇÕES

- **LINK VISITADO:**
  - VERIFICA SE A PÁGINA FOI VISITADA. SE FOI, PROCURAR O PRÓXIMO LINK DA LISTA QUE NÃO FOI VISITADO.
- **NAVEGANDO NO DOMÍNIO:**
  - VERIFICAR SE O HOSTNAME DE CADA PÁGINA É IGUAL AO DO DOMÍNIO, SE NÃO, PROCURO O PRÓXIMO LINK DA LISTA.
- **HOSTNAME VAZIO:**
  - VERIFICAR SE HOSTNAME EXISTE , SE NÃO , PROCURO O PRÓXIMO LINK QUE TENHA.
- **ROBOTS.TXT:**
  - VERIFICO SE É POSSÍVEL CRAWLEAR A PÁGINA DE ACORDO COM AS REGRAS DO ROBOT.TXT , SE NÃO , PROCURO O PRÓXIMO LINK.
- **REQUISIÇÃO FUNCIONA:**
  - VERIFICAR SE A RESPOSTA DA REQUISIÇÃO FUNCIONA.(RETURN RESPONSE 200).

CRAWLER - BASELINE

- **BUSCA EM LARGURA**
  - USANDO OS LINKS DAS PÁGINAS.

```
for link in soup.findAll('a'):
    href = link.get('href')
```

Baseline - Tempo

Média Tempo-Download

```
print('Média Tempo-Download em Segundos: ',timeData['Tempo'].mean())
print('Média Tempo-Download em Minutos: ',timeData['Tempo'].mean()/60)
print('Média Tempo-Download em Horas: ',(timeData['Tempo'].mean()/60)/60)
```

```
Média Tempo-Download em Segundos:  4619.0
Média Tempo-Download em Minutos:  76.98333333333333
Média Tempo-Download em Horas:  1.2830555555555556
```

Tempo de Download - csv

```
# Tempo de download dos anteriores - tempo em horas
timeData.to_csv('CrawlerBaselineTime.csv',index_label=False)
timeData
```

	Dominio	Tempo
0	https://www.capefeargames.com/	1.100833
1	https://www.mtgotraders.com/	1.451667
2	https://abugames.com/	4.820278
3	https://www.cardkingdom.com/	0.579722
4	http://www.mtgmintcard.com/	1.284444
5	https://scryfall.com/	0.563333
6	http://www.starcitygames.com/	0.421944
7	https://www.wizardscupboard.com/	1.315278
8	https://www.tcgplayer.com/	0.010000

CRAWLER - HEURÍSTICA

- **BUSCA ORDENADA**
  - FILTRADA PELAS TAGS E ORDENADA POR UM SCORE.

CRAWLER - HEURÍSTICA

- **FILTRO**

```
# Encontrar todos os links da pagina(url) filtrando pelas tags li ul tr td e div.
for lista in soup.findAll(re.compile(r'li|ul|td|tr|div')):
    for link in lista.findAll('a',href = True):
```

CRAWLER - HEURÍSTICA

PADRÕES

```
def Padroes():

    regex = r'(.+)(magic_singles-)(.+)(\/)(.+)(\/)(\d+)'
    CapeFearGamesMagicCard = re.compile(regex)

    regex2 = r'(.+)(store\/)([A-Z]{3}|[A-Z]{1}[0-9]{2})(.+)'
    MtGoTradersMagicCard = re.compile(regex2)

    regex3 = r'(.)*(p-)(\d+)(\.html)'
    WizardCupBoardMagicCard = re.compile(regex3)

    regex4 = '(.)*(magic-the-gathering\/)(singles\/)(product-detail\/)(.+)'
    AbuGamesMagicCard = re.compile(regex4)

    regex5 = r'(.+)(mtg\/)(.+)'
    CardKingdomMagicCard = re.compile(regex5)

    regex6 = r'(.+)(magic_the_gathering\/)(product\/)(\d+)(\/\d+)*'
    StarCityMagicCard = re.compile(regex6)

    regex7 = r'(.+)(card)(.+)(\d+)(.+)'
    ScryfallMagicCard = re.compile(regex7)

    regex8 = r'(.+)(mtg\/)(singles\/)(.+\/)(.+)'
    MtgMIntCard = re.compile(regex8)
```

WORDS

```
def Words():
    goods = ['magic','single','set','catalog','gathering','card','store']
    bad = ['pokemom','yugioh']
    words = [goods,bad]
    return words
```

CRAWLER - HEURÍSTICA

- **SCORE**

```
def Score(url,siteInicial,texto):
    score = 0

    regexMoney = re.compile(r'(\$)(\d+)(.)(\d{2})')

    # Dentro do Dominio +1
    if siteInicial in url:
        score = score + 1

    # url contem alguma palavra da lista de palavraes boas que aparecem no link relevante +2
    for palavra in Words()[0]:
        #casefold ignorar sensitive-case
        if palavra.casefold() in url.casefold():
            score = score + 2

    # url fullmatch com link relevante +5
    for padrao in Padroes():
        if padrao.fullmatch(url):
            score = score + 5

    #CArdpage tem texto sobre valor da carta. +10
    if regexMoney.match(texto):
        score = score + 10

     # url contem alguma palavra da lista de palavras ruins(pokemom, yugioh ...) zera o score.
    for palavra in Words()[1]:
        if palavra.casefold() in url.casefold():
            score = 0
            return score
    return score
```

Heurística- Tempo

Média Tempo-Download Heuristica

```
]: print('Média Tempo-Download em Segundos: ',dataTimeHeuristica['Tempo-Download'].mean())
   print('Média Tempo-Download em Minutos: ',dataTimeHeuristica['Tempo-Download'].mean()/60)
   print('Média Tempo-Download em Horas: ',(dataTimeHeuristica['Tempo-Download'].mean()/60)/60)
```

```
Média Tempo-Download em Segundos:  3627.285714285714
Média Tempo-Download em Minutos:  60.4547619047619
Média Tempo-Download em Horas:  1.007579365079365
```

Tempo-Download Heurística-Horas

```
# Convertendo para horas
dataTimeHeuristica['Tempo-Download'] = dataTimeHeuristica['Tempo-Download'].apply(lambda x: (x / 60)/60)
dataTimeHeuristica
```

	Tempo-Download	Site
0	0.589444	https://www.cardkingdom.com/
1	0.825833	http://www.starcitygames.com/
2	0.464167	https://scryfall.com/
3	1.659444	https://www.caperfeargames.com/
4	1.175000	https://www.wizardcupboard.com/
5	1.006944	https://www.mtgotraders.com/
6	1.332222	http://www.mtgmintcard.com/

Estratégias: Baseline x Heurística

Harvest Ratio

Baseline

	Ratio	Relevantes	Total de Paginas
0	0.341582	2733	8001

Heurística

	Ratio	Relevantes	Total de Paginas
0	0.697757	4885	7001

Classificador

Extração das Features

```
def encode_url_to_file(url):
    return re.sub("\/", "-", url)
def get_html_text(url):
    page = requests.get(url)
    with open("./train-set/" + encode_url_to_file(url) + ".html", 'w' , encoding='utf-8') as outfile:
        outfile.write(page.text)
    soup = BeautifulSoup(page.content)
    return soup.get_text()

def get_text_from_link(link):
    html = open(link, "r")
    soup = BeautifulSoup(html)
    return soup.get_text()
```

```
en_stopwords = set(stopwords.words("english"))

def extract_features(text):
    data = re.sub("[^$0-9a-zA-Z]", " ", text).lower()
    words = data.split()
    return [w for w in words]#if not w in en_stopwords]
```

Treinamento

```
def train(clf, data, target):
    pipe_clf = Pipeline([
        ('vect', CountVectorizer(analyzer = "word",
                                 tokenizer = None,
                                 preprocessor = None,
                                 stop_words = None)),
        ('tfidf', TfidfTransformer()),
        ('clf', clf),
    ])
    parameters = {
        'vect__ngram_range': [(1, 1), (1, 2)],
        'vect__max_features': [500, 1000, 5000],
        'tfidf__use_idf': (True, False),
        #'clf__alpha': (1e-2, 1e-3),
    }

    scoring = {'AUC': 'roc_auc', 'Accuracy': make_scorer(accuracy_score)}
    gs_clf = GridSearchCV(pipe_clf, parameters, cv=10, iid=False, n_jobs=-1, scoring=scoring, refit='AUC', return_trai
    fit = gs_clf.fit(data, target)
    print(fit.cv_results_)
    return fit
```

Classificadores

```
clfs = [SVC(), MultinomialNB(), SGDClassifier(), LogisticRegression(), tree.DecisionTreeClassifier(),
        MLPClassifier(max_iter = 100)]
```

- **best:**
  - **SGDClassifier**
- **média:**
  - **70 á 81 %**

Tempo e Score

Classificador	Tempo Total	Score
SVC	18.84	0.75
MultinomialNB	17.85	0.7428
SGDClassifier	17.60	0.8142
LogisticRegression	18.45	0.7785
DecisionTree	22.69	0.7714
MLP	198.44	0.7714

Extratores

- Um para cada site!

Extratores

```python
def extract_from_any_link(link, soup):
    try:
        if 'capefeargames' in link:
            return extract_capefeargames(soup)
        if 'cardkingdom' in link:
            return extract_cardkingdom (soup)
        if 'mtgotraders' in link:
            return extract_mtgotraders(soup)
        if 'starcitygames' in link:
            return extract_starcitygames(soup)
        if 'wizardscupboard' in link:
            return extract_wizardscupboard(soup)
        if 'abugames' in link:
            return extractor_abugames(soup)
        if 'scryfall' in link:
            return extract_scryfall(soup)
        if 'mtgmintcards' in link:
            return extract_mtgmintcards(soup)
    except:
        return None
    return None
```

```python
def extract_capefeargames(soup):
    root_div = soup.find("div", class_="product-info")
    title = root_div.find('h1').text
    price_span = root_div.find('span', class_='price')
    price = re.sub('[\s+]', '', price_span.text) if price_span != None else ''
    infos_box = root_div.find_all('div', class_='info-box')
    description = infos_box[1].find('p').text
    extra_infos = infos_box[2].find('p')
    infos = {}
    key = ""
    value = ""
    for child in extra_infos.children:
        if child.name == None:
            continue
        if key == "":
            key = child.text
        else:
            value = value + child.text + " "
        if child.name == "br":
            infos[key] = value.strip()
            key = ""
            value = ""
    return {"name": title, "desc": description, "price": price, "infos": infos}
```

Saída da Extração


```
{'name': 'Core Set 2020: Agonizing Syphon', 'desc': 'Agonizing Syphon deals 3 damage to any target and you gain 3 lif
e.', 'price': '$0.25', 'infos': {'Edition': 'Core Set 2020', 'Type': 'Sorcery', 'Cast': '3b', 'Rarity': 'C'}}
{'name': 'Core Set 2020: Air Elemental', 'desc': 'Flying', 'price': '$0.25', 'infos': {'Edition': 'Core Set 2020', 'T
ype': 'Creature - Elemental', 'Cast': '3uu', 'Rarity': 'U', 'Pow/Tuf': '4/4'}}
{'name': 'Modern Horizons: Alpine Guide', 'desc': 'When Alpine Guide enters the battlefield, you may search your libr
ary for a Mountain card, put that card onto the battlefield tapped, then shuffle your library.\nAlpine Guide attacks
each combat if able.\nWhen Alpine Guide leaves the battlefield, sacrifice a Mountain.', 'price': '$0.25', 'infos':
{'Edition': 'Modern Horizons', 'Type': 'Snow Creature - Human Scout', 'Cast': '2r', 'Rarity': 'U', 'Pow/Tuf': '3/3'}}
{'name': 'Modern Horizons: Altar of Dementia', 'desc': "Sacrifice a creature: Target player puts a number of cards eq
ual to the sacrificed creature's power from the top of their library into their graveyard. \n Pro Tip!\n\nAltar of De
mentia is a longtime Commander favorite, especially among players who love to fill up their graveyards. It recently e
ntered the Modern format thanks to a reprint in Modern Horizons.", 'price': '$1.79', 'infos': {'Edition': 'Modern Hor
izons', 'Type': 'Artifact', 'Cast': '2', 'Rarity': 'R'}}
{'name': 'Modern Horizons: Angel Token', 'desc': 'Flying, vigilance', 'price': '$0.25', 'infos': {'Edition': 'Modern
Horizons', 'Type': 'Token Creature - Angel', 'Cast': '', 'Rarity': 'S', 'Pow/Tuf': '4/4'}}
{'name': 'Ultimate Masters: Aethersnipe', 'desc': "When Aethersniper enters the battlefield, return target nonland pe
rmanent to its owner's hand.\nEvoke  (You may cast this spell for its evoke cost. If you do, it's sacrificed when it
enters the battlefield.)", 'price': '$0.25', 'infos': {'Edition': 'Ultimate Masters', 'Type': 'Creature - Elemental',
'Cast': '5u', 'Rarity': 'C', 'Pow/Tuf': '4/4'}}
{'name': 'Ultimate Masters: All Is Dust', 'desc': 'Each player sacrifices all colored permanents they control.', 'pri
ce': '$9.99', 'infos': {'Edition': 'Ultimate Masters', 'Type': 'Tribal Sorcery - Eldrazi', 'Cast': '7', 'Rarity':
'R'}}
{'name': 'Commander 2015: Blasted Landscape', 'desc': ': Add  to your mana pool.\nCycling  (, Discard this card: Draw
a card.)', 'price': '$0.99', 'infos': {'Edition': 'Commander 2015', 'Type': 'Land', 'Cast': '', 'Rarity': 'U'}}
{'name': 'Global Series: Jiang Yanggu & Mu Yanling: Island', 'desc': 'No Text', 'price': '$0.35', 'infos': {'Editio
```

OBRIGADO!

DUVIDAS?