Faculdade IMPACTA TECNOLOGIA

Pós Graduação em BI com Big Data

Web Mining

Crawler e Scraping

Prof. MSc. Fernando Sousa

Bibliografia

- Bibliografia básica para esta aula
  - Weiss SM, Indurkhya N, Zhang T. Fundamentals of Predictive Text Mining. 2010 edition. London ; New York: Springer; 2010. 226 p. - Cap. 2
  - Manning CD, Raghavan P, Schütze H. Introduction to Information Retrieval. 1 edition. New York: Cambridge University Press; 2008. 506 p. – Cap. 2 e 6
  - Liu B. Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data. 1st ed. 2007. Corr. 2nd printing edition. Berlin ; New York: Springer; 2009. 552 p – Cap 6.

Bibliografia

- Bibliografia complementar
  - Salton G, McGill MJ. Introduction to Modern Information Retrieval. New York, NY, USA: McGraw-Hill, Inc.; 1986.
  - Sousa FS. Análise Comparativa de Métodos de Recuperação de Informação para Categorização de Conteúdos Web Relacionados à Saúde [Dissertação de Mestrado]. [São Paulo]: Universidade Federal de São Paulo (UNIFESP); 2011.
  - Intro to Computer Science & Programming Course [Internet]. [cited 2015 Jul 4]. Disponível em: https://www.udacity.com/course/cs101

Faculdade IMPACTA TECNOLOGIA

Introdução

- Realizar tarefas de Web Mining requer o conhecimento de técnicas básicas de Text Mining
- Um documento da web nada mais é que um texto, que pode conter algumas outras características
  - Marcadores
  - Links
  - Meta dados
  - Semântica
- Na fase de preparação dos dados, ou pré-processamento, técnicas de Text Mining são utilizadas para que os documentos sejam computacionalmente compreensíveis

Faculdade IMPACTA TECNOLOGIA

Introdução

Faculdade IMPACTA TECNOLOGIA

Introdução

Pré-processamento - Preparação dos Dados

- Textos são dados não estruturados
  - Não estão representados de maneira organizada
  - Não estão em formato de tabela
  - Não contém atributos e valores explícitos
- Antes de qualquer tarefa de Web Mining, como classificação ou recuperação de informação, os textos devem ser transformados em dados estruturados numéricos
  - **Extração de características**
  - Gerar atributos/características em forma de tabela de dados

Pré-processamento - Preparação dos Dados

- A maioria das aplicações trata um texto como um conjunto de palavras
  - Estratégia conhecida como “bag of words”
  - As palavras, ou termos, existentes no texto definirão suas características ou atributos
- Os atributos extraídos de um texto podem ser:
  - A ocorrência das palavras em um texto
  - A função gramatical de uma palavra
  - O significado da palavra

Pré-processamento - Preparação dos Dados

- A extração de características de um texto segue algum passos básicos
  - Definição e coleta dos documentos
  - Tokenização
  - Processamento linguístico
  - Geração do vetor de características

Coleta dos documentos

- Um conjunto de documentos pode vir de diversas fontes
  - Conjuntos de arquivos em um computador
  - Campo textual em uma base de dados
  - E-mails
  - Páginas da web
  - Palavras inseridas em buscadores
  - Páginas pessoais em redes sociais
  - Logs de acesso às páginas web
  - Tweets
  - Enciclopédias eletrônicas
  - Opiniões sobre um produto ou serviço

Coleta dos documentos

- Algumas bases de dados (como a web) podem ser enormes. Seleciona-se então um subconjunto da base de dados (amostra)
  - Período de tempo
  - Tamanho
  - Mais acessados
- Critérios de seleção e quais documentos irão compor a amostra dependem da aplicação

Coleta dos documentos

- Algumas bases de dados públicas:
  - Temario
    - http://www.nilc.icmc.usp.br/nilc/tools/TeMario.zip
  - Revista Fapesp
    - http://www.nilc.icmc.usp.br/nilc/tools/Fapesp%20Corpora.htm
  - CETENFolha
    - http://www.linguateca.pt/CETENFolha/
  - Outras
    - http://www.nilc.icmc.usp.br/nilc/tools/corpora.htm

Coleta dos documentos

- Relação de páginas web:
  - Alexa – Páginas mais visitadas no brasil
    - http://www.alexa.com/topsites/countries/BR
  - DMOZ – Relação de páginas existentes na web
    - https://dmoztools.net/

Coleta dos documentos

- Kaggle:
  - Tweets – Democratas x republicanos
    - https://www.kaggle.com/kapastor/democratvsrepublicantweets
  - Opiniões sobre hotéis
    - https://www.kaggle.com/jiashenliu/515k-hotel-reviews-data-in-europe#Hotel_Reviews.csv
  - Opiniões de e-commerce de roupas
    - https://www.kaggle.com/nicapotato/womens-ecommerce-clothing-reviews
  - Stackoverflow
    - https://www.kaggle.com/stackoverflow/stackoverflow

Coleta dos documentos

- Kaggle:
  - Boatos de whatsapp
    - https://www.kaggle.com/rogeriochaves/boatos-de-whatsapp-boatosorg
  - Análise de sentimento em tweets
    - https://www.kaggle.com/crowdflower/twitter-airline-sentiment#Tweets.csv
    - https://www.kaggle.com/crowdflower/twitter-airline-sentiment#Tweets.csv
    - https://www.kaggle.com/welkin10/airline-sentiment

Faculdade IMPACTA TECNOLOGIA

Crawler

- Uma estratégia para coletar textos da web é utilizar um *crawler* (“rastreador”)
- Um *crawler* é um programa que percorre automaticamente a estrutura de links da Web e salva as páginas dos links em um repositório local
- Um *crawler* pode salvar todas as páginas que encontra na busca (*universal crawler*) ou apenas aquelas relacionadas a um tópico (*topic crawler*)

Faculdade IMPACTA TECNOLOGIA

Crawler

- Apesar de conceitualmente simples, a construção de um *crawler* não é tão simples
  - Deve ser eficiente
  - Deve lidar com bloqueios de segurança
- Existem softwares, como o RapidMiner, que implementam um *crawler*, facilitando a criação de uma base de dados
- Para páginas web que não tem bloqueios de segurança (login ou captcha, por exemplo), um crawler pode ser facilmente implementado com uma linguagem de programação, através de requisições HTTP GET

Crawler

- Vídeos sobre *Web Crawler* e Buscadores
  - https://www.youtube.com/watch?t=52&v=XFZxSogAwXo
    - Web Crawler
  - https://www.youtube.com/watch?t=10&v=VIbvK4OJ1Os
    - Como encontrar links em uma página
  - https://www.youtube.com/watch?v=K_ArZ2B8aiQ
    - Coletando links
  - https://www.youtube.com/watch?t=83&v=9nkR2LLPiYo
    - Como funcionam os buscadores
  - https://www.youtube.com/watch?v=35DgBfhtf9A
    - O que é mais importante para construir um buscador

Intro to Computer Science & Programming Course [Internet]. [cited 2015 Jul 4]. Available from: https://www.udacity.com/course/cs101

Faculdade IMPACTA TECNOLOGIA

Crawler

- O resultado de um crawler é um conjunto de conteúdo de páginas web, normalmente HTML
- Após a coleta, o armazenamento das páginas web pode ser feito de várias maneiras:
  - Arquivo de texto
  - Base de dados relacional
  - Planilhas eletrônicas
  - Base de dados não relacional

Faculdade IMPACTA TECNOLOGIA

Crawler

- Algoritmo base de um crawler:
  - Escolha um link da web para iniciar o crawler
  - Adicione em uma lista de links para capturar
  - Enquanto a lista de links não estiver vazia:
    - Pegue um link da lista
    - Se o link ainda não foi capturado:
      - Capture (GET) o conteúdo HTML deste link
      - Armazene o conteúdo capturado
      - Procure os links existentes no conteúdo deste link
      - Armazene os links encontrados na lista para capturar
      - Armazene o link na lista de capturados

Crawler com Python

- Vamos primeiro aprender as tarefas básicas do crawler (captura e processamento) para depois montar o algoritmo
- Bibliotecas:
  - urllib, beautifulsoup
  - Já incluídas no Anaconda
- Importações necessárias:

```
from urllib.request import urlopen
from urllib.parse import urljoin
from bs4 import BeautifulSoup
```

  - urlopen: faz a requisição GET do link e retorna em um objeto
  - urljoin: mesclar dois endereços web
  - BeatifulSoup: fazer scraping na página web (encontrar elementos)

Crawler com Python

- Capturar o conteúdo de uma página web via GET

```
# fazer GET da página web
init = "https://pt.wikipedia.org/wiki/Cerveja"
get = urlopen(init)
# recuperar conteúdo HTML
html = get.read()
```

  - urlopen(link): faz a requisição GET do link e retorna em um objeto
  - read(): retorna o conteúdo HTML da requisição

Crawler com Python

- Encontrar todos os links da página (marcador <a>) com BeatifulSoup
  - Documentação do BeatifulSoup: https://www.crummy.com/software/BeautifulSoup/bs4/doc

```
# criar objeto do BS
bsObj = BeautifulSoup(html, "html.parser")
# encontrar todos os links <a>
links_a = bsObj('a')
```

  - BeatifulSoup(html, parser): cria um objeto com o conteúdo da página (html), segundo uma regra (parser).
  - Com este objeto é possível enviar parâmetros para procurar elementos no conteúdo da página

Crawler com Python

- O que determina o endereço dentro de um link HTML (<a>) é o atributo href
- Alguns dos links retornados não tem este atributo e outros tem o valor de href começando com “#”
  - # indica que é uma referência para a mesma página. Não faz sentido colocar estes links para coleta
- É possível fazer este filtro diretamente no objeto criado
  - Retornar todos os links da página com o atributo href e que este não comece com #
- Para isso, crie uma função que faz essa verificação
  - Esta função será enviada como parâmetro do objeto BeatifulSoup

Crawler com Python

```
# utilizar beatiful soup para pegar apenas os links válidos
# função para verificar se o link é válido (não vazio e não
inicia com #)
def links_validos(href):
    return href and href[0] != '#'
# função é enviada como valor do parâmetro href
validos = bsObj('a', href=links_validos)
```

Crawler com Python - Scraping

- A busca pode ser feita com os seguintes parâmetros

  `bsObj(name, attrs, recursive, string, limit, **kwargs)`

  - name: nome do marcador HTML
    - Exemplo: bsObj(‘a’)
  - attrs: dicionário com atributos do marcador que deseja filtrar
    - Exemplo: links com a classe ‘mw-redirect’

```
filtro_classe = bsObj('a', attrs={'class':'mw-redirect'})
```

- Exemplo: links com a classe ‘mw-redirect’ e title ‘Pilsen’

```
filtro_classe_titulo =
    bsObj('a', attrs={'class':'mw-redirect', 'title’:‘Pilsen'})
```

Crawler com Python - Scraping

```
bsObj(name, attrs, recursive, string, limit, **kwargs)
```

- recursive: se procura nos elementos filhos (True, padrão) ou somente na raiz do objeto (False)
  - Exemplo: bsObj('a', recursive= False) -> não retorna nada
- string: filtrar pelo texto dentro do marcador
  - Exemplo: links cujo texto que aparece na tela seja Pilsen

```
filtro_string = bsObj('a', string='Pilsen'})
```

- limit: número inteiro. Limita a quantidade de resultados

Crawler com Python - Scraping

```
bsObj(name, attrs, recursive, string, limit, **kwargs)
```

- **kwargs: filtrar por qualquer atributo html. O atributo é o nome do parâmetro enviado
  - Exemplo: links com a classe ‘mw-redirect’

```
filtro_classe = bsObj('a’, class_='mw-redirect')
```

  - Exemplo: links com a classe ‘mw-redirect’ e title ‘Pilsen’

```
filtro_classe_titulo =
          bsObj('a’, class_='mw-redirect’, title=‘Pilsen’)
```

Crawler com Python - Scraping

- Veja que o uso é similar ao do parâmetro attrs
  - ao invés de ser um dicionário de atributos, os atributos são passados como parâmetro diretamente na função
- A regra geral é que o nome do parâmetro seja o mesmo nome do atributo HTML com as seguinte exceções:
  - Atributos com hífen devem não podem der passados como parâmetro, apenas utilizando attrs
  - O atributo name não pode ser passado como parâmetro, uma vez que é o nome do primeiro parâmetro que pode ser enviado, e identifica o marcador HTML. Para filtrar pelo atributo name deve-se utilizar attr
  - O atributo class pode ser enviado como parâmetro, mas utilizando o parâmetro class_, já que class é uma palavra reservada no python

Crawler com Python - Scraping

- Todos os exemplo anteriores buscavam links com algum filtro.
- Entretanto, todos os filtros podem ser utilizados com qualquer marcador HTML, ou até mesmo sozinhos

```
# procurar por elementos do tipo span com classe mw-headline
filtro_span = bsObj('span', class_='mw-headline’)

# procurar por qualquer elemento que contenha o texto cerveja
# retorna apenas o texto, e não o elemento
import re
filtro_cerveja = bsObj(string=re.compile('cerveja',re.IGNORECASE))
```

  - re: biblioteca de expressão regular do Python

Crawler com Python

- Voltando aos links, por enquanto temos uma lista de elementos HTML <a>. O que precisa ser feito:
  - Extrair o atributo href de cada um dos links
  - Montar o endereço completo do link

```
# retornar apenas atributo href (onde está o link)
# e transformar no endereço completo utilizando urljoin
enderecos = [urljoin(init, l.get('href')) for l in links_a]
```

Crawler utilizando Python

- urljoin mescla dois links:
  - Se o segundo parâmetro for de outro endereço, o retorno será o link do segundo parâmetro
  - Se o segundo parâmetro for o caminho relativo do link, junta o caminho relativo com o link do primeiro parâmetro
- Exemplos:
  - urljoin("https://pt.wikipedia.org/wiki/Cerveja", "/wiki/Bebida") --> https://pt.wikipedia.org/wiki/Bebida
  - urljoin("https://pt.wikipedia.org/wiki/Cerveja", "http://google.com") --> http://google.com

Crawler utilizando Python - Algoritmo

- Conhecendo os comandos básicos para fazer um crawler, vamos implementá-lo completo em python
- Vamos fazer isso dentro de uma função crawler, que recebe 2 parâmetros: o link inicial do crawler e a quantidade máxima de páginas

```
def crawler(init, max = 10):
    # implementação do crawler
```

Crawler utilizando Python - Algoritmo

- Passo 1: escolher o link de início do crawler (parâmetro da função)
- Passo 2: colocar o link em uma lista para buscar
  - Vamos utilizar o tipo set do Python, um tipo de dado que não tem elementos repetidos
  - Adicionalmente, vamos criar um set para os links que já foram buscados e um contador para controlar a quantidade de páginas já capturadas

```
def crawler(init, max = 10):
    buscar = set([init])
    buscados = set()
    contador = 0
```

Crawler utilizando Python - Algoritmo

- Passo 3: Enquanto houver links na lista ou atingir a quantidade máxima de links esperada

```
def crawler(init, max = 10):
    buscar = set([init])
    buscados = set()
    contador = 0
    while buscar and contador <= max:
        # executar crawler
```

Crawler utilizando Python - Algoritmo

- Passo 4: remover e retornar um elemento da lista para buscar

```
def crawler(init, max = 10):
    buscar = set([init])
    buscados = set()
    contador = 0
    while buscar and contador <= max:
        link = buscar.pop()
```

Crawler utilizando Python - Algoritmo

- Passo 5: verificar se o link ainda não foi capturado

```
def crawler(init, max = 10):
    buscar = set([init])
    buscados = set()
    contador = 0
    while buscar and contador <= max:
        link = buscar.pop()
        if link not in buscados:
           # executar crawler
```

Crawler utilizando Python - Algoritmo

- Passo 6: capturar conteúdo HTML

```
def crawler(init, max = 10):
    buscar = set([init])
    buscados = set()
    contador = 0
    while buscar and contador <= max:
        link = buscar.pop()
        if link not in buscados:
           conteudo = urlopen(link).read().decode('utf-8')
```

  - A função decode foi utilizada para transformar o retorno do GET em uma string codificada com UTF-8

Crawler utilizando Python - Algoritmo

- Passo 7: Armazenar conteúdo capturado (por exemplo em arquivo)

```
def crawler(init, max = 10):
    buscar = set([init])
    buscados = set()
    contador = 0
    while buscar and contador <= max:
        link = buscar.pop()
        if link not in buscados:
            print(link)
             conteudo = urlopen(link).read().decode('utf-8’)
             contador += 1
             salva_arquivo('wikis/wiki'+str(contador)+'.txt', conteudo)


def salva_arquivo(nome, conteudo):
    with open(nome,'w', encoding="utf8") as file:
        file.write(conteudo)
```

Crawler utilizando Python - Algoritmo

- Passo 8: Procurar os links existentes no conteúdo do link atual

```
def crawler(init, max = 10):
    buscar = set([init])
    buscados = set()
    contador = 0
    while buscar and contador <= max:
        link = buscar.pop()
        if link not in buscados:
            print(link)
                conteudo = urlopen(link).read().decode('utf-8’)
                contador += 1
                salva_arquivo('wikis/wiki'+str(contador)+'.txt', conteudo)
                novos_links = busca_links(conteudo, link)

def salva_arquivo(nome, conteudo):
    with open(nome,'w', encoding="utf8") as file:
        file.write(conteudo)
def busca_links(conteudo, link):
    links = BeautifulSoup(conteudo, "html.parser")('a', href=links_validos)
    # retorna o link absoluto
    links = [wiki for wiki in [urljoin(init, l.get('href')) for l in links] \
        if wiki.startswith('https://pt.wikipedia.org/wiki') ]
    return links
```

Crawler utilizando Python - Algoritmo

- Passo 9 e 10: Armazenar os novos links encontrados na lista para capturar e armazena o link na lista de capturados

```
def crawler(init, max = 10):
    buscar = set([init])
    buscados = set()
    contador = 0
    while buscar and contador <= max:
        link = buscar.pop()
        if link not in buscados:
            print(link)
                conteudo = urlopen(link).read().decode('utf-8’)
                contador += 1
                salva_arquivo('wikis/wiki'+str(contador)+'.txt', conteudo)
                novos_links = busca_links(conteudo, link)
                buscar.update(novos_links)
                buscados.add(link)
def salva_arquivo(nome, conteudo):
    with open(nome,'w', encoding="utf8") as file:
        file.write(conteudo)
def busca_links(conteudo, link):
    links = BeautifulSoup(conteudo, "html.parser")('a', href=links_validos)
    # retorna o link absoluto
    links = [wiki for wiki in [urljoin(init, l.get('href')) for l in links] \
        if wiki.startswith('https://pt.wikipedia.org/wiki') ]
    return links
```

Faculdade IMPACTA TECNOLOGIA

Crawler utilizando Python

- Para páginas web é necessário fazer um processamento do conteúdo para:
  - Remover marcadores HTML (<html>, <body>, , etc) e extrair somente o texto
  - Ignorar o conteúdo de determinados marcadores
  - Extrair o texto apenas de apenas um grupo de marcadores
- Marcadores HTML definem a estrutura do documento e não tem valor para identificar o conteúdo e o assunto do texto
- Entretanto, em outro tipo de abordagem, pode ter valor semântico
  - Em que parte do documento está o texto? Título? Corpo? Metadados?

Crawler utilizando Python

- Por exemplo

```
<html>
    <head>
        Jordânia e Israel abrem passagem na fronteira
    <head>
    <body>
        Israel e Jordânia abriram ontem a primeira passagem de fronteira entre os dois países depois de 46 anos de estado formal de guerra.
    </body>
</html>
```


Fonte: Folha de São Paulo - Adaptado

Crawler utilizando Python

- Por exemplo – remover marcadores HTML

```
<html>
    <head>
        Jordânia e Israel abrem passagem na fronteira
    <head>
    <body>
        Israel e Jordânia abriram ontem a primeira passagem de fronteira entre os dois países depois de 46 anos de estado formal de guerra.
    </body>
</html>
```

Fonte: Folha de São Paulo - Adaptado

Crawler utilizando Python

- Utilizando beautifulsoup o texto de todo o documento é retornando utilizando o atributo text do objeto criado

```
# percorrer os arquivos da pasta onde estão os arquivos
import os
for i, arquivo in enumerate(os.listdir('wikis')):
    with open('wikis/'+arquivo, 'r', encoding="utf8") as arq:
        # le conteúdo do arquivo
        html = arq.read()
    bsObj = BeautifulSoup(html, "html.parser")
    texto = bsObj.text
    # salvar em um novo arquivo
    # mesmo método utilizado anteriormente
    salva_arquivo('wikis_text/wiki'+str(i+1)+'.txt', texto)
```

Crawler utilizando Python

- Veja que o arquivo gerado tem muito código
  - Provenientes de marcadores script e style
  - Devem ser removidos

```
for i, arquivo in enumerate(os.listdir(os.getcwd()+'/wikis')):
    with open('wikis/'+arquivo, 'r', encoding="utf8") as arq:
        # le conteúdo do arquivo
        html = arq.read()
    bsObj = BeautifulSoup(html, "html.parser")
    # remover tags de script e style
    for remove in bsObj(["script", "style"]): remove.extract()
    # remover espaços e quebras de linhas
    texto = bsObj.text
    # salvar em um novo arquivo
    salva_arquivo('wikis_text/wiki'+str(i+1)+'.txt', texto)
```

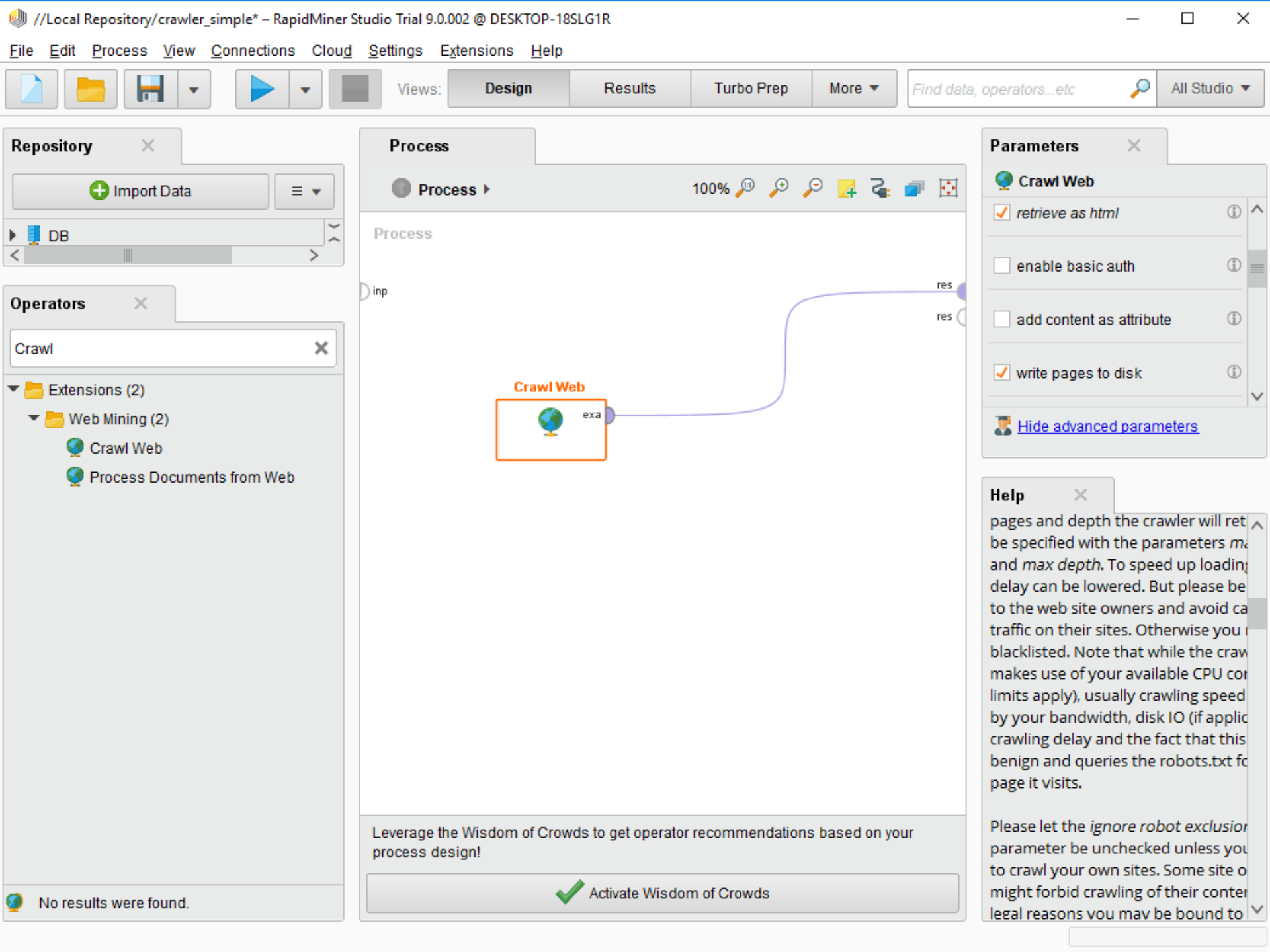
Crawler utilizando Python

- Também podemos ter interesse em extrair o texto só uma região da página
  - Por exemplo, div com o id='content'
  - É onde está o conteúdo principal da página

```
# percorrer os arquivos da pasta onde estão os arquivos
for i, arquivo in enumerate(os.listdir(os.getcwd()+'/wikis')):
    with open('wikis/'+arquivo, 'r', encoding="utf8") as arq:
        # le conteúdo do arquivo
        html = arq.read()
    bsObj = BeautifulSoup(html, "html.parser")
    # remover tags de script e style
    for remove in bsObj(["script", "style"]): remove.extract()
    # extrair o texto apenas da div com id content
    texto = bsObj('div', id='content')[0].text
    # salvar em um novo arquivo
    salva_arquivo('wikis_text/wiki'+str(i+1)+'.txt', texto)
```

Crawler utilizando Rapid Miner

- O RapidMiner tem diversos operadores, fazendo com que um mesmo problema possa ser resolvido com operadores diferentes
- Para fazer Crawler, o RapidMiner tem o operador Crawl Web
- Encontre o operador Crawl Web na árvore Extensions -> Web Mining e coloque na área do processo


//Local Repository/crawler_simple* – RapidMiner Studio Trial 9.0.002 @ DESKTOP-18SLG1R
File Edit Process View Connections Cloud Settings Extensions Help
Views: Design Results Turbo Prep More
Find data, operators...etc
All Studio
Repository
Import Data
DB
Operators
Crawl
Extensions (2)
Web Mining (2)
Crawl Web
Process Documents from Web
No results were found.
Process
Process
100%
Process
inp
res
res
Crawl Web
exa
Leverage the Wisdom of Crowds to get operator recommendations based on your process design!
Activate Wisdom of Crowds
Parameters
Crawl Web
retrieve as html
enable basic auth
add content as attribute
write pages to disk
Hide advanced parameters
Help
pages and depth the crawler will ret
be specified with the parameters ma
and max depth. To speed up loadin
delay can be lowered. But please be
to the web site owners and avoid ca
traffic on their sites. Otherwise you
blacklisted. Note that while the craw
makes use of your available CPU cor
limits apply), usually crawling speed
by your bandwidth, disk IO (if applic
crawling delay and the fact that this
benign and queries the robots.txt fo
page it visits.
Please let the ignore robot exclusio
parameter be unchecked unless you
to crawl your own sites. Some site o
might forbid crawling of their conte
legal reasons you may be bound to

Crawler utilizando Rapid Miner

- Parâmetros importantes:
  - url: link de onde começar o crawler
  - crawling rules: lista com regras para fazer o crawler:
    - Padrão de URL
    - Padrão de conteúdo
    - Padrão no link da página
  - max crawl depth: quantidade máxima de “filhos” para percorrer
  - retrieve as html: retornar como conteúdo HTML
  - max pages: quantidade máxima para coletar
  - write pages to disk: salvar a página coletada
  - Output dir: caminho de onde salvar as páginas

Crawler utilizando Rapid Miner

- Fizemos um Crawler que armazena as páginas web em arquivos
- Agora vamos ler os arquivos desta pasta e extrair o conteúdo textual
- Procure por Loop Files em Utility -> Process Control –> Loops e coloque na área do processo
- Nos parâmetros, selecione a pasta onde salvou as páginas coletadas

Crawler utilizando Rapid Miner

- Clique duas vezes no operador Loop Files inserido
  - Será aberto um subprocesso
  - Os operadores que colocar ai dentro serão aplicados para cada arquivo da pasta
- Procure pelo operador Read Document em Extensions -> Text Processing e coloque na área de processos
  - Configure o parâmetro *encoding* para UTF-8
  - Desmarque a opção *extract text only*
- Ligue a entrada fil do subprocesso na entrada fil de Read Document

Crawler utilizando Rapid Miner

- Procure pelo operador Extract Content em Extensions -> Web Mining -> Html Processing e coloque na área de processos
- Ligue a saída out de Read Document na entrada doc de Extract Content
- Ligue a saída de Extract Content na saída do subprocesso
- Volte ao Processo principal e ligue a saída do Loop Files na saída do processo

Faculdade IMPACTA TECNOLOGIA

Exercício

- Pesquise por sites que podem ter conteúdo para ajudar no desenvolvimento do seu TCC ou para o trabalho de web mining
- Construa um Crawler para recuperar dados relevantes do site e armazenar em arquivos de texto, utilizando Python
- Caso não tenha sites sobre seu TCC, construa um Crawler para extrair as avaliações do Kindle. Comece por este link: https://www.amazon.com.br/Kindle-preta-sens%C3%ADvel-toque-Gera%C3%A7%C3%A3o/product-reviews/B0186FEYKW/ref=cm_cr_getr_d_paging_btm_1?ie=UTF8&reviewerType=all_reviews&pageNumber=1
  - Veja que para encontrar mais avaliações basta mudar o número da página

Faculdade
IMPACTA
TECNOLOGIA