Faculdade IMPACTA TECNOLOGIA

Pós Graduação em BI com Big Data

Web Mining

Pré-processamento e modelo vetorial

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

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

Tokenização

- Tokenização é o processo de identificar sequências de caracteres como tokens
  - Token é uma sequência de caracteres em um documento que forma uma unidade semântica (palavra ou termo)
- A abordagem mais simples é separar por espaços
  - Cada termo separado por um espaço é um token
- Neste processo também são definidos quais caracteres serão excluídos e caracteres maiúsculos são trocados por minúsculos
  - Pontuação, numerais, caracteres especiais,...

Tokenização

- 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

Tokenização

- 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

Tokenização

- Por exemplo – remover caracteres indesejados

```
<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

Tokenização

- Por exemplo – tokenização

```
<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

Tokenização

- Por exemplo – colocar todas as letras em minúsculo

```
<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

Tokenização

- Por exemplo – resultado final

jordânia e israel abrem passagem na fronteira israel e jordânia abriram ontem a primeira passagem de fronteira entre os dois países depois de anos de estado formal de guerra.

Fonte: Folha de São Paulo - Adaptado

Processamento Linguístico

- Na fase do processamento linguístico são realizadas as tarefas:
  - Remover palavras indesejadas (stopwords)
  - Encontrar a forma base dos termos (steamming e lemmatização)
  - Extrair a contagem de termos do texto

Faculdade IMPACTA TECNOLOGIA

Stopwords

- Stopwords são palavras comuns que aparecem nos textos, independente do domínio de conhecimento, e devem ser removidas durante a preparação dos dados
  - Tem pouco valor na tarefa de mineração
  - Usualmente são aproximadamente metade das palavras de um texto.
  - Reduzem a complexidade do problema
  - Normalmente são pronomes, preposições, conjunções, artigos

Faculdade IMPACTA TECNOLOGIA

Stopwords

- Usualmente listas de stopwords são utilizadas para identificar quais palavras devem ser removidas
  - A lista pode variar de acordo com a aplicação e com o problema
  - Existem listas genéricas para cada idioma, que são um bom início para remover stopwords
  - O Snowball (http://snowball.tartarus.org) tem uma lista de stopwords para português do Brasil

Stopwords

- Outra estratégia é remover as palavras mais frequentes do conjunto
  - Ver sobre Lei de Zipf
- As palavras que restarem após a remoção das stopwords serão os tokens que representam o texto

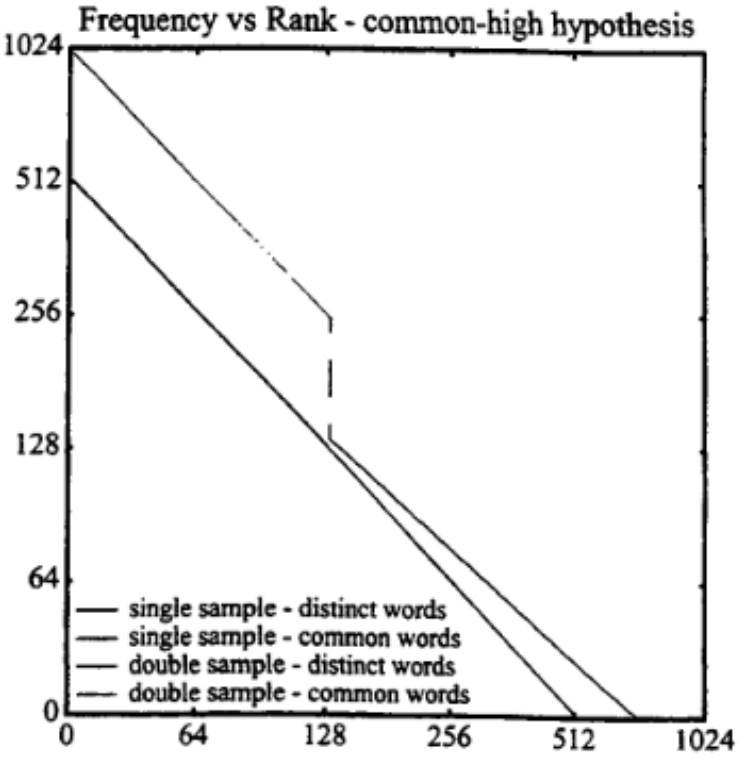


Figure 12. *Model with high freq. common vocabulary*

Powers DMW. Applications and explanations of Zipf's law. Proceedings of the Joint Conferences on New Methods in Language Processing and Computational Natural Language Learning [Internet].

Stopwords

- Exemplo: Dada a seguinte lista de stopwords e o mesmo texto dos exemplos anteriores, quais serão os tokens que representarão o texto?

Stopwords: de na a entre os e dois para se

jordânia e israel abrem passagem na fronteira israel e jordânia abriram ontem a primeira passagem de fronteira entre os dois países depois de anos de estado formal de guerra

Fonte: Folha de São Paulo - Adaptado

Stopwords

- Exemplo: Dada a seguinte lista de stopwords e o mesmo texto dos exemplos anteriores, quais serão os tokens que representarão o texto?

Stopwords: de na a entre os e dois para se

jordânia e **israel abrem passagem** na **fronteira israel** e **jordânia abriram ontem** a **primeira passagem** de **fronteira** entre os dois **países depois** de **anos** de **estado formal** de **guerra**

  - 18 termos

Fonte: Folha de São Paulo - Adaptado

Stemming e lemmatização

- Em um texto uma palavra é escrita de diferentes formas
  - Abrir, abrem, abriram
  - Tecnologia, tecnologias
  - Pedra, pedreira, pedregulho
- Stemming e lemmatização reduzem as flexões (gênero, número e grau) e as derivações de uma palavra para uma base comum, a fim de reduzir a quantidade de caracteres e a complexidade do problema
  - Abrir, abrem, abriram → abrir
  - Tecnologia, tecnologias → tecnologia
  - Pedra, pedreira, pedregulho → pedra

Stemming

- Stemming é o processo de cortar o final das palavras (sufixos) a fim de encontrar a raiz (stem).
  - É uma abordagem simples, e nem sempre pode funcionar
  - Parte do princípio de que na média funciona bem
  - Abrir, abrem, abriram → abr
  - Tecnologia, tecnologias → tecnolog
  - Pedra, pedreira, pedregulho → pedr

Stemming

- Principais problemas:
  - Não considera a existência de prefixos
    - Empedrar → empedr
    - Raiz correta: pedr
  - O simples corte do final da palavra não considera questões gramaticais e morfológicas
    - Somos → som
    - Fomos → fom
    - Era → e
    - Raiz correto → ser (verbo)

Faculdade IMPACTA TECNOLOGIA

Lemmatização

- Na lemmatização a transformação é mais elaborada, utilizando vocabulários e análise morfológica das palavras
- A lemmatização retorna a base ou a forma do dicionário da palavra
  - Ser, somos, fomos, era → ser

Faculdade IMPACTA TECNOLOGIA

Stemming e Lemmatização

- A implementação de um stemmer ou lemmatizador é bem complexa, composta de inúmeras regras
- Existem stemmers e lemmatizadores largamente utilizados e descritos na literatura científica que são confiáveis
- É uma opção rápida para inserir na sua solução de text mining
  - Desenvolver um stemmer ou lemmatizador levaria um bom tempo...

Faculdade IMPACTA TECNOLOGIA

Stemming e Lemmatização

- Opções de stemmers e lemmatizadores:
  - Porter Stemmer
    - http://www.tartarus.org/~martin/PorterStemmer
  - PTStemmer – para português
    - https://code.google.com/p/ptstemmer
  - SnowBall – para português
    - http://snowball.tartarus.org/algorithms/portuguese/stemmer.html
  - Plugins para Weka
    - http://weka.wikispaces.com/Stemmers

Stemming e Lemmatização

- Exemplo: aplicando Stemmer ao texto que estamos trabalhando, um resultado seria:

jordânia israel abrem passagem fronteira israel jordânia abriram ontem primeira passagem fronteira países depois anos estado formal guerra

jordân israel abrem passag fronteir israel jordân abrir ontem primeir passag fronteir país depo anos estad formal guerr

  - Repare que o Stemmer aplicado não identificou que abrem e abriram (abrir) são o mesmo verbo

- Este será o conjunto de palavras (“bag of words”) que representa o texto

Fonte: Folha de São Paulo - Adaptado

Vetor de características

- Com os termos encontrados e processados, o próximo passo é construir o conjunto de dados (dataset)
- Cada registro ou exemplo do dataset é composto por atributos ou características
  - Vetor de características
- Um registro pode ter um atributo especial que indica a categoria do texto

Vetor de características

Sexo	Batimento cardíaco	Peso	Código da doença
M	175	65	3
F	141	72	1
...	...	...	...
F	161	59	2

Baseado em: Weiss, Sholom M., Nitin Indurkhya, and Tong Zhang. Fundamentals of predictive text mining. Vol. 41. Springer, 2010. p2.

Vetor de características

- Nas tarefas de Text e Web Mining cada exemplo do dataset representa de um documento
- Os termos extraídos do texto serão as características do exemplo
- O valor de cada característica será um número que representa uma contagem de ocorrência do termo no texto
  - Existem algumas estratégias de contagem

Vetor de características

	$Termo_1$	$Termo_2$	...	$Termo_n$	Categoria
$Texto_1$	$f_{1,1}$	$f_{1,2}$	...	$f_{1,n}$	C_1
$Texto_2$	$f_{2,1}$	$f_{2,2}$	...	$f_{2,n}$	C_1
...	...	...	...	...	...
$Texto_m$	$f_{m,1}$	$f_{m,2}$	...	$f_{m,n}$	C_2

Vetor de características

- Exemplo: No texto que estamos trabalhando cada palavra presente no texto será um atributo do *dataset* ou vetor de características

jordân israel abrem passag fronteir israel jordân abrir ontem primeir passag fronteir país depo anos estad formal guerr

jordân	israel	abrem	passag	fronteir	abrir	ontem	primeir	país	depo	anos	estad	formal	guerr	Categoria

Fonte: Folha de São Paulo – Adaptado

Vetor de características

- No exemplo anterior criamos o vetor baseado em apenas um texto
- Em uma aplicação real há centenas ou milhares de textos dentro de um corpus (conjunto de textos que será analisado)
- Os atributos do vetor de características deverá conter todos os termos encontrados em todo o corpus

Vetor de características

- Exemplo: Considere que o texto abaixo também faça parte do seu corpus, pertencendo a categoria “Economia”

EUA e Japão se armam para guerra comercial

- Aplicando o pré-processamento anterior (tokenização, remoção de stopwords e stemming), são extraídos os seguintes termos:

eua japã armam guerr comercial

Fonte: Folha de São Paulo – Adaptado

Vetor de características

- O vetor de características com os dois textos ficará da seguinte forma

jordân israel abrem passag fronteir israel jordân abrir ontem primeir passag fronteir país depo anos estad formal guerr

eua japã armam guerr comercial

jordân	israel	abrem	passag	fronteir	abrir	ontem	primeir	país	depo	anos	estad	formal	guerr	eua	japã	armam	comercial	Categoria

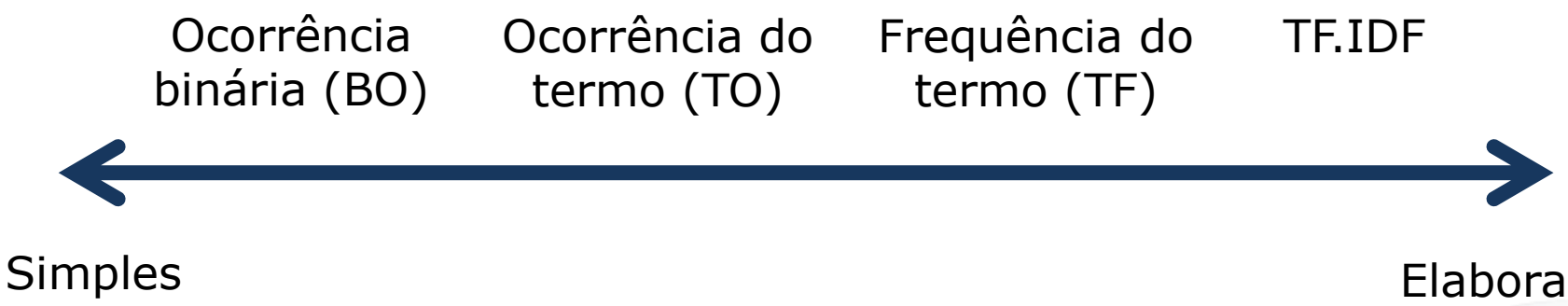
Fonte: Folha de São Paulo – Adaptado

Contagem dos termos

- Agora que sabemos quais são as características do *dataset*, deve-se atribuir valores para estas características, para cada texto do corpus
- A abordagem mais simples, mas que tem ótimos resultados, é verificar a ocorrência e realizar a contagem dos termos no corpus

Contagem dos termos

- Existem algumas técnicas, das mais simples até as mais elaboradas
  - Não existe a melhor. Sempre depende da aplicação

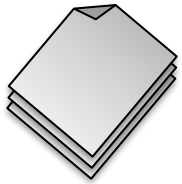


Ocorrência binária (BO)

- Ocorrência binária (*Binary Occurrence* - BO) considera apenas a ocorrência ou não de um termo no texto
- O valor de cada atributo do vetor de características pode assumir somente 2 valores
  - 1 (presença do termo no texto); ou
  - 0 (ausência do termo no texto)

Ocorrência binária (BO)

- Exercício: crie o vetor de características dos textos abaixo, utilizando BO.
  - Lembre que valor de cada atributo é ocorrência binária de um termo no texto



	$Termo_1$	$Termo_2$	...	$Termo_n$	Categoria
$Texto_1$	$f_{1,1}$	$f_{1,2}$	...	$f_{1,n}$	C_1
$Texto_2$	$f_{2,1}$	$f_{2,2}$	...	$f_{2,n}$	C_1
...	...	...	...	...	...
$Texto_m$	$f_{m,1}$	$f_{m,2}$	...	$f_{m,n}$	C_2

jordân israel abrem passag fronteir israel jordân abrir ontem primeir passag fronteir país depo anos estad formal guerr

eua japã armam guerr comercial

Fonte: Folha de São Paulo – Adaptado

Ocorrência binária (BO)

- Exemplo: vetor de características com BO

jordân israel abrem passag fronteir israel jordân abrir ontem primeir passag fronteir país depo anos estad formal guerr

eua japã armam guerr comercial

Documento	jordân	israel	abrem	passag	fronteir	abrir	ontem	primeir	país	depo	anos	estad	formal	guerr	eua	japã	armam	comercial
D1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
D2	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

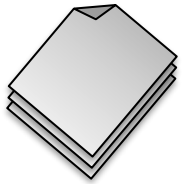
Fonte: Folha de São Paulo – Adaptado

Ocorrência do termo (TO)

- Ocorrência do termo (*Term Occurrence* - TO) conta quantas vezes um termo aparece em um documento
- O valor de cada atributo do vetor de características será a quantidade de vezes que o termo aparece

Ocorrência do termo (TO)

- Exercício: crie o vetor de características dos textos abaixo, utilizando TO.
  - Lembre que valor de cada atributo é ocorrência binária de um termo no texto



	$Termo_1$	$Termo_2$	...	$Termo_n$	Categoria
$Texto_1$	$f_{1,1}$	$f_{1,2}$	...	$f_{1,n}$	C_1
$Texto_2$	$f_{2,1}$	$f_{2,2}$	...	$f_{2,n}$	C_1
...	...	...	...	...	...
$Texto_m$	$f_{m,1}$	$f_{m,2}$	...	$f_{m,n}$	C_2

jordân israel abrem passag fronteir israel jordân abrir ontem primeir passag fronteir país depo anos estad formal guerr

eua japã armam guerr comercial

Fonte: Folha de São Paulo – Adaptado

Ocorrência do termo (TO)

- Exemplo: vetor de características com TO

jordân israel abrem passag fronteir israel jordân abrir ontem primeir passag fronteir país depo anos estad formal guerr

eua japã armam guerr comercial

Documento	jordân	israel	abrem	passag	fronteir	abrir	ontem	primeir	país	depo	anos	estad	formal	guerr	eua	japã	armam	comercial
D1	2	2	1	2	2	1	1	1	1	1	1	1	1	1	0	0	0	0
D2	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Fonte: Folha de São Paulo – Adaptado

Frequência do termo (TF)

- Frequência do termo (*Term Frequency* - TF) considera quantas vezes um termo ocorre no texto em relação ao total de palavras do texto
  - O valor de cada atributo do vetor de características será um número decimal, dado por:

$$tf_{t,d} = \frac{to_{t,d}}{n_d}$$

  - $tf_{t,d}$: Frequência do termo t no documento d
  - $to_{t,d}$: Ocorrência do termo termo t no documento d
  - n_d: Quantidade de termos no documento d

Frequência do termo (TF)

- Por exemplo: a frequência dos termos israel e guerr no texto é seguir será:

jordân israel abrem passag fronteir israel jordân abrir ontem primeir passag fronteir país depo anos estad formal guerr

israel

n_{d1}= 18

$to_{israel,d1}$ = 2

$tf_{israel,d1}$ = 2/18 = 0,11

guerr

n_{d1}= 18

$to_{guerra,d1}$ = 1

$tf_{guerra,d1}$ = 1/18 = 0,05

Fonte: Folha de São Paulo – Adaptado

Frequência do termo (TF)

- Os vetores de características para os dois textos:

jordân israel abrem passag fronteir israel jordân abrir ontem primeir passag fronteir país depo anos estad formal guerr

eua japã armam guerr comercial

Documento	jordân	israel	abrem	passag	fronteir	abrir	ontem	primeir	...	formal	guerr	eua	japã	armam	comercial
D1	0,11	0,11	0,05	0,11	0,11	0,05	0,05	0,05	...	0,05	0,05	0	0	0	0
D2	0	0	0	0	0	0	0	0	...	0	0,2	0,2	0,2	0,2	0,2

Fonte: Folha de São Paulo – Adaptado

TF.IDF

- No método de TF.IDF a frequência do termo é calculada como no método anterior, entretanto é ponderado pelo inverso da frequência deste mesmo termo nos documentos (*inverse document frequency* - IDF)
  - Frequência nos documentos é em quantos documentos o termo aparece
  - IDF pondera o TF atribuindo a importância do termo para diferenciar um documento dos outros
    - Se o termo aparece em muitos documentos, ele não é importante para diferenciar, e TF.IDF diminuirá
    - Se o termo aparece em poucos documentos, ele tem uma importância maior, e TF.IDF aumentará

TF.IDF

- O IDF é calculado da seguinte forma:

$$idf_t = \log \frac{N}{df_t}$$

  - idf_t: inverso da frequência de documentos do termo t
  - N: Quantidade de documentos no corpus
  - df_t: frequência de documentos do termo t
- Portanto, TF.IDF será calculado por:

$$tf.idf_{t,d} = tf_{t,d}\ x\ idf_t$$

TF.IDF

- Por exemplo: o TF.IDF de israel e guerr no texto é seguir será:

jordân israel abrem passag fronteir israel jordân abrir ontem primeir passag fronteir país depo anos estad formal guerr

israel

$tf_{israel,d1}$ = 0,11

N= 2

df_{israel} = 1

idf_{israel} = log(2/1) = 0,3

$tf.idf_{israel,d1}$ = 0,11 * 0,3 = 0,033

guerr

$tf_{guerr,d1}$ = 0,05

N= 2

df_{guerr} = 2

idf_{guerr} = log(2/2) = 0

$tf.idf_{guerr,d1}$ = 0,05 * 0 = 0

Fonte: Folha de São Paulo – Adaptado

TF.IDF

- Observe que o atributo guerr terá o valor 0 para D1
  - O termo guerr aparece nos dois textos, então não tem relevância para diferenciá-los

israel

$tf_{israel,d1}$ = 0,11

N= 2

df_{israel} = 1

idf_{israel} = log(2/1) = 0,3

$tf.idf_{israel,d1}$ = 0,11 * 0,3 = 0,03

guerr

$tf_{guerr,d1}$ = 0,05

N= 2

df_{guerr} = 2

idf_{guerr} = log(2/2) = 0

$tf.idf_{guerr,d1}$ = 0,05 * 0 = 0

TF.IDF

- Os vetores de características com TF.IDF para os dois textos:

jordân israel abrem passag fronteir israel jordân abrir ontem primeir passag fronteir país depo anos estad formal guerr

eua japã armam guerr comercial

Documento	jordân	israel	abrem	passag	fronteir	abrir	ontem	primeir	...	formal	guerr	eua	japã	armam	comercial
D1	0,03	0,03	0,01	0,03	0,03	0,01	0,01	0,01	...	0,01	0	0	0	0	0
D2	0	0	0	0	0	0	0	0	...	0	0	0,06	0,06	0,06	0,06

Fonte: Folha de São Paulo – Adaptado

Faculdade IMPACTA TECNOLOGIA

TF.IDF

- Relação entre os termos e o TF.IDF
  - Maior quando um termo ocorre muitas vezes em poucos documentos
  - Menor quando o termo ocorre poucas vezes em um documento ou ocorre em muitos documentos
  - Menor ainda quando ocorre em quase todos os documentos

Vetor de características

- Com o vetor de características pronto, temos em mãos os textos representados de forma estruturada, na forma de conjunto de palavras (“bag of words”)
- Este s vetores de características e conjuntos de palavras é a base para as aplicações de text mining e web content mining
  - Classificação
  - Recuperação de informação
  - Buscadores
  - Clusters

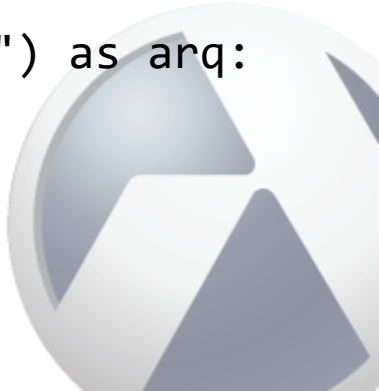
Vetor de características utilizando Python

- Utilizando TfidfVectorizer do sklearn vamos fazer todos os processos anteriores:
  - Tokenização
  - Remover stopwords
  - Aplicar stemmer
  - Gerar o vetor de características

Vetor de características utilizando Python

- Na forma mais simples, o modulo TfidfVectorizer faz a divisão dos tokens e a contagem dos termos em um mesmo processo
- Com ele é também é possível remover stopwords e aplicar o stemmer
- Primeiro vamos carregar o arquivos da seguinte forma:

```
# ler textos dos arquivos
lista_conteudos = []
for arquivo in os.listdir(os.getcwd()+'/wikis_text'):
    with open('wikis_text/'+arquivo, 'r', encoding="utf8") as arq:
        lista_conteudos.append(arq.read())
```



Vetor de características utilizando Python

- A forma mais simples de utilizar o TfidfVectorizer é a seguinte:

```
contagem = TfidfVectorizer()
vetor = contagem.fit_transform(lista_conteudos)
#nomes das características
print(contagem.get_feature_names())
# vetor de características
print(vetor.toarray())
```

- TfidfVectorizer() cria um objeto com os parâmetros d vetorização
- fit_transform(lista_conteudos) aplica a vetorização na lista de textos enviada como parâmetro

Vetor de características utilizando Python

- Na configuração padrão:
  - Um token será qualquer sequência de 1 ou mais caracteres (letras, números ou _)
  - Transforma tudo em letras minúsculas
  - Calcula a frequência por tf.idf
  - Não remove stop words nem aplica stemmer

Vetor de características utilizando Python

- Contagem dos termos por BO
- Para contar a ocorrência dor termos por BO, os seguintes parâmetros deve ser alterados:
  - binary=True, norm=None, use_idf=False

```
# extrair ocorrência binária
bo = TfidfVectorizer(binary=True, norm=None, use_idf=False)

vetor = bo.fit_transform(lista_conteudos)
#nomes das características
print(bo.get_feature_names())
# vetor de características
print(vetor.toarray())
```

Vetor de características utilizando Python

- Contagem dos termos por TO
- Para contar a ocorrência dor termos por TO, os seguintes parâmetros deve ser alterados:
  - binary=False, norm=None, use_idf=False

```
# extrair com ocorrência dos termos
to = TfidfVectorizer(binary=False, norm=None, use_idf=False)

vetor = to.fit_transform(lista_conteudos)
#nomes das características
print(to.get_feature_names())
# vetor de características
print(vetor.toarray())
```

Vetor de características utilizando Python

- Contagem dos termos por TF
- Para contar a ocorrência dor termos por TF, os seguintes parâmetros deve ser alterados:
  - binary=False, norm='l1', use_idf=False

```
# extrair com tf
tf = TfidfVectorizer(binary=False, norm='l1', use_idf=False)

vetor = tf.fit_transform(lista_conteudos)
#nomes das características
print(tf.get_feature_names())
# vetor de características
print(vetor.toarray())
```

Vetor de características utilizando Python

- norm define o tipo de normalização dos elementos do vetor
  - l1: divide cada elemento do vetor pela soma dos elementos
  - l2: divide cada elemento do vetor pela raiz da soma dos quadrados de cada elemento (euclidiana)
  - None: não usa normalização (TO)

Vetor de características utilizando Python

- Contagem dos termos por TF.IDF
- Para contar a ocorrência dor termos por TF.IDF, os seguintes parâmetros deve ser alterados:
  - binary=False, norm='l1', use_idf=True

```
# extrair com tf.idf
tfidf = TfidfVectorizer(binary=False, norm='l1', use_idf=True)

vetor = tfidf.fit_transform(lista_conteudos)
#nomes das características
print(tfidf.get_feature_names())
# vetor de características
print(vetor.toarray())
```

Vetor de características utilizando Python

- Remover termos irrelevantes
- É possível definir como será o padrão do token utilizando expressão regular
- A configuração padrão utiliza sequências de letras, número e underline
- O padrão pode ser definido para atender as características do problema
- Vamos configurar o parâmetro token_pattern para utilizar somente os palavras com 3 ou mias letras

Vetor de características utilizando Python

```
# token pattern: termos com letras, com 3 ou mias caracteres
tfidf = TfidfVectorizer(binary=False,norm='l1', use_idf=True,
                token_pattern=r'\b[a-zA-Z]{3,}\b')
vetor = tfidf.fit_transform(lista_conteudos)
#nomes das características
print(tfidf.get_feature_names())
# vetor de características
print(vetor.toarray())
```

Vetor de características utilizando Python

- Remover stopwords
- A lista de stopwords é enviada pelo par6ametro stop_words
- Podemos manter a lista de stopwords em um arquivo, ler este arquivo e transformar em uma lista de palavras

```
# ler arquivo com stopwords e transforma em uma lista
stopwords = []
with open('stopwords.txt','r', encoding='utf-8') as s:
    stopwords = s.read().split('\n')

tfidf = TfidfVectorizer(binary=False,norm='l1', use_idf=True,
                token_pattern=r'\b[a-zA-Z]{3,}\b',
                stop_words=stopwords)
vetor = tfidf.fit_transform(lista_conteudos)
```

Vetor de características utilizando Python

- Stemmer
- Sklearn não tem algoritmo de stemmer
- O stemmer está na biblioteca nltk
  - Tem stemmer para português
  - nltk está no Anaconda
- Primeiro faça a importação e o download do pacote de stemmer

```
import nltk
nltk.download('rslp')
```

Vetor de características utilizando Python

- O módulo do nltk que faz o stemmer é nltk.stem.RSLPStemmer
- Crie uma função que recebe um texto e retorna o texto com o stemmer aplicado

```
# método para fazer o stemmer
def stemmer(doc):
    stemmer = nltk.stem.RSLPStemmer()
    return ' '.join([stemmer.stem(t) for t in nltk.word_tokenize(doc)])
```

  - word_tokenizer separa o texto em tokens

Vetor de características utilizando Python

- Este método deve ser enviado no parâmetro preprocessor do TfidfVectorizer

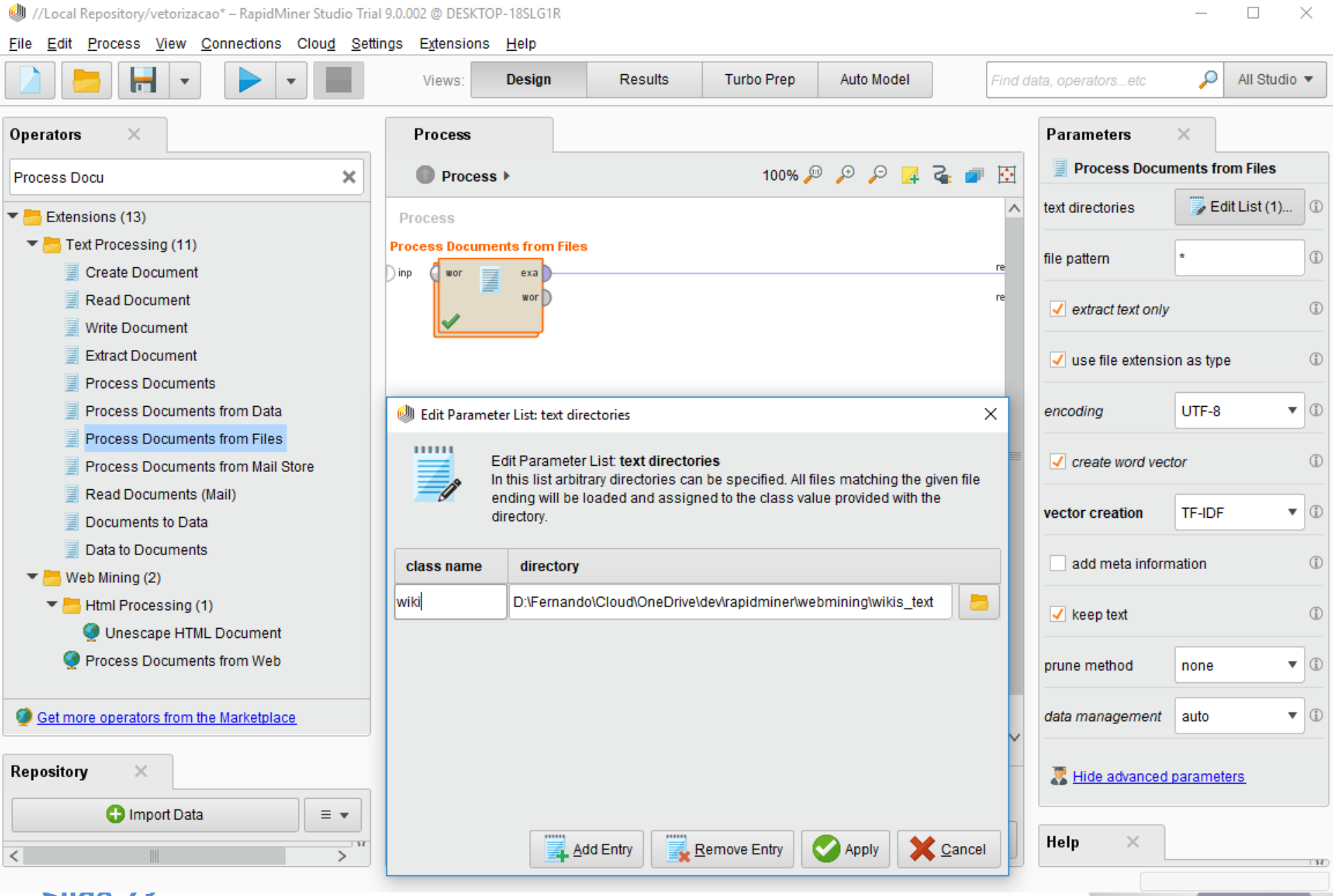
```
tfidf = TfidfVectorizer(binary=False,norm='l1', use_idf=True,
                        token_pattern=r'\b[a-zA-Z]{3,}\b’,
                        stop_words=stopwords,
                        preprocessor=stemmer)
vetor = tfidf.fit_transform(lista_conteudos)
```

Vetor de características utilizando RapidMiner

- Com o RapidMiner também é possível fazer o pré-processamento e gerar o vetor de características
- Procure o operador Process Documents from Files, na árvore Extensions → Text Processing
  - Este operador lê um conjunto de textos de arquivos e gera o vetor de características

Vetor de características utilizando RapidMiner

- Faça as seguintes configurações nos parâmetros do operador:
  - text directories: clique no botão Edit List e na janela que abrir selecione a pasta onde estão os arquivos que que processar.
    - O campo class name serve para definir uma categoria para um conjunto de arquivos, para quando estiver trabalhando com classificação. Coloque um nome qualquer por hora
  - encoding: UTF-8
  - create word vector: marcado
  - vector creation: forma de contagem dos termos. Selecione tf.idf
  - add meta information: marcado


//Local Repository/vetorizacao* – RapidMiner Studio Trial 9.0.002 @ DESKTOP-18SLG1R
File Edit Process View Connections Cloud Settings Extensions Help
Views: Design Results Turbo Prep Auto Model
Find data, operators...etc
All Studio
Operators
Process Docu
Extensions (13)
Text Processing (11)
Create Document
Read Document
Write Document
Extract Document
Process Documents
Process Documents from Data
Process Documents from Files
Process Documents from Mail Store
Read Documents (Mail)
Documents to Data
Data to Documents
Web Mining (2)
Html Processing (1)
Unescape HTML Document
Process Documents from Web
Get more operators from the Marketplace
Repository
Import Data
Process
Process
100%
Process
Process Documents from Files
inp
wor
exa
wor
Edit Parameter List: text directories
Edit Parameter List: text directories
In this list arbitrary directories can be specified. All files matching the given file ending will be loaded and assigned to the class value provided with the directory.
class name
directory
wiki
D:\Fernando\Cloud\OneDrive\dev\rapidminer\webmining\wikis_text
Add Entry
Remove Entry
Apply
Cancel
Parameters
Process Documents from Files
text directories
Edit List (1)...
file pattern
*
extract text only
use file extension as type
encoding
UTF-8
create word vector
vector creation
TF-IDF
add meta information
keep text
prune method
none
data management
auto
Hide advanced parameters
Help

Vetor de características utilizando RapidMiner

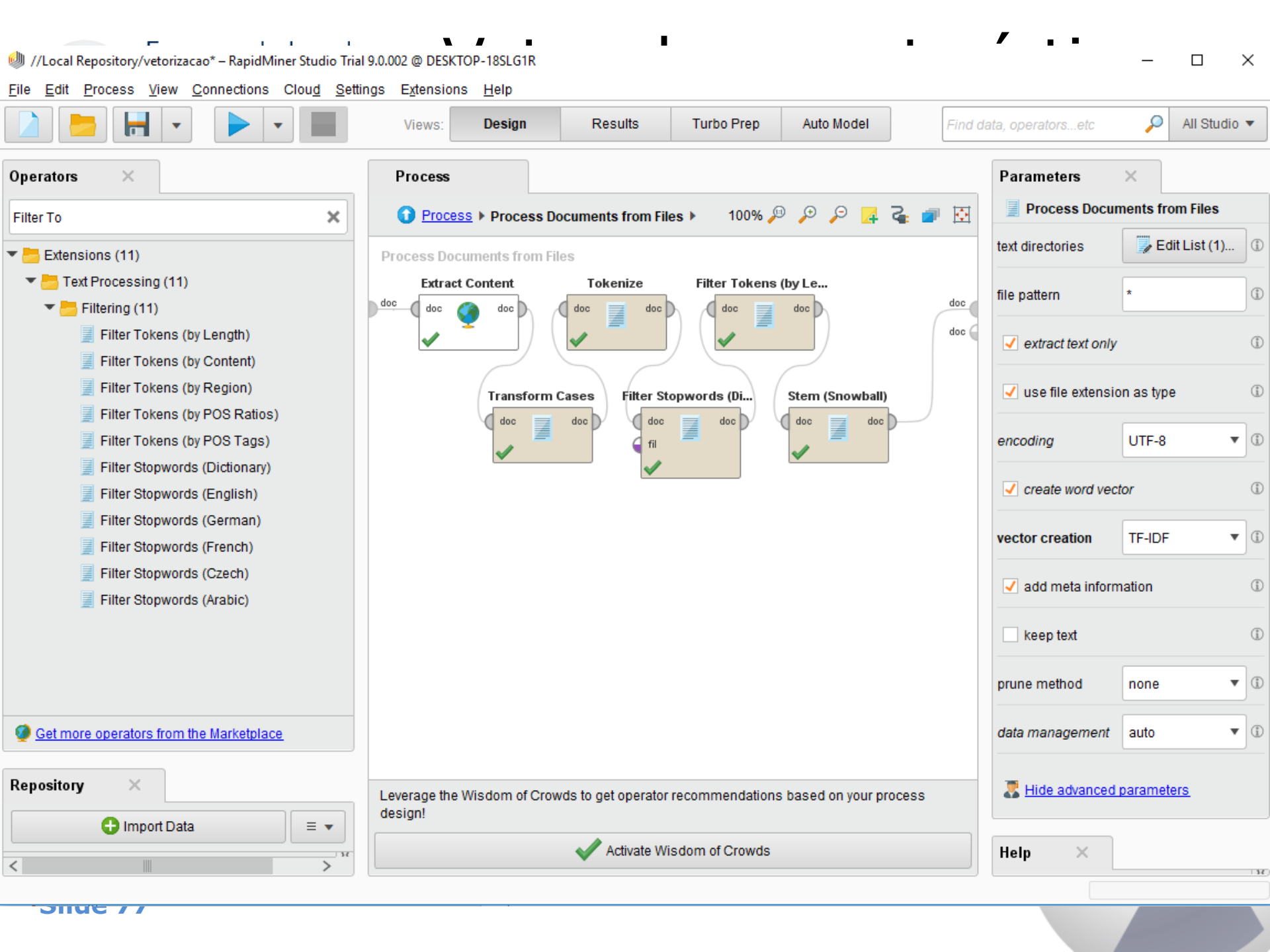
- Process Documents from Files tem um subprocesso
  - Clique duas vezes no operador para abri-lo
- Dentro desse subprocesso é onde será feito o pré-processamento
  - Extração do conteúdo HTML
  - Padronizar o texto para minúsculas
  - Tokenizar
  - Remover stopwords
  - Remover palavras pequenas e grandes
  - Aplicar stemmer

Vetor de características utilizando RapidMiner

- Procure os seguintes operadores:
  - Extract Content
    - Extrair conteúdo do HTML. Deixe a configuração padrão
  - Transform Cases
    - Transformar texto em minúsculo. Deixe a configuração padrão (lower case)
  - Tokenize
    - Separar os tokens. Deixe a configuração padrão (non letters)

Vetor de características utilizando RapidMiner

- Procure os seguintes operadores:
  - Filter Stopwords (Dictionary)
    - Remover stopwords dos tokens. As stopwords devem estar em um arquivo, uma por linha
    - Selecione o arquivo de stopwords no parâmetro file e coloque UTF-8 no parâmetro encoding
  - Filter Tokens (by Length)
    - Remove tokens menores e maiores que a configuração dos parâmetros min chars e max chars
  - Stem (Snowball)
    - Aplica o stemmer nos tokens. Escolha Portuguese no parâmetro language


//Local Repository/vetorizacao* – RapidMiner Studio Trial 9.0.002 @ DESKTOP-18SLG1R
File Edit Process View Connections Cloud Settings Extensions Help
Views: Design Results Turbo Prep Auto Model
Find data, operators...etc
All Studio
Operators
Filter To
Extensions (11)
Text Processing (11)
Filtering (11)
Filter Tokens (by Length)
Filter Tokens (by Content)
Filter Tokens (by Region)
Filter Tokens (by POS Ratios)
Filter Tokens (by POS Tags)
Filter Stopwords (Dictionary)
Filter Stopwords (English)
Filter Stopwords (German)
Filter Stopwords (French)
Filter Stopwords (Czech)
Filter Stopwords (Arabic)
Get more operators from the Marketplace
Repository
Import Data
Process
Process ▸ Process Documents from Files ▸
100%
Process Documents from Files
Extract Content
Tokenize
Filter Tokens (by Le...
Transform Cases
Filter Stopwords (Di...
Stem (Snowball)
doc
fil
Leverage the Wisdom of Crowds to get operator recommendations based on your process design!
Activate Wisdom of Crowds
Parameters
Process Documents from Files
text directories
Edit List (1)...
file pattern
*
extract text only
use file extension as type
encoding
UTF-8
create word vector
vector creation
TF-IDF
add meta information
keep text
prune method
none
data management
auto
Hide advanced parameters
Help

Exercício

- Faça o processo de vetorização utilizando python e depois com o RapidMiner para a base de notícias disponibilizada.
- São notícias de 4 temas: agricultura, ciência e tecnologia (CTI), cultura e esportes
- Extraia apenas o conteúdo referente as notícias, eliminando todo o resto do HTML. Em cada tema, o conteúdo de notícia está em:
  - Agricultura: class='visualiza-noticia'
  - CTI: id = 'layout-column_column-2'
  - Cultura: class='journal-content-article'
  - Esportes: class='item-page'

Exercício

- Dica: trabalhar com textos e html nem sempre é simples poder ocorrer problemas como:
  - Codificação de caracteres (acentos, caracteres de escape)
  - Palavras de dois marcadores HTML adjacentes podem ficar juntas

```
<h1>Dólar apresenta forte alta</h1>
<div>Na última semana a alta acumulada da moeda...</div>
```

```
Dólar apresenta forte altaNa última semana a alta
acumulada da moeda
```

  - Faça o processamento destes textos e tente identificar como corrigir estes problemas

Exercício

- Para a leitura de arquivos em pastas diferentes, utilize os seguinte código:

```
for pasta in os.listdir('noticias'):
    for arquivo in os.listdir('noticias/'+pasta):
        with open('noticias/'+pasta+'/'+arquivo, ‘r’) as arq:
            conteudo_html = arq.read()
            # completar com a extração do texto
```

Faculdade
IMPACTA
TECNOLOGIA