

DATA MINING & MACHINE LEARNING (I)

Thiago Marzagão



Centro Universitário

transformando textos em dados

- ▶ Documento 1: “Não trabalho para ter clientes; tenho clientes para poder trabalhar.”
- ▶ Documento 2: “Não se pode forçar a inteligência a trabalhar.”

transformando textos em dados

- ▶ Como transformar textos em dados?

transformando textos em dados

- ▶ Como transformar textos em dados?
- ▶ R: Contar as ocorrências de cada palavra, em cada documento.

bag of words

- ▶ Documento 1: “Não trabalho para ter clientes; tenho clientes para poder trabalhar.”
- ▶ Documento 2: “Não se pode forçar a inteligência a trabalhar.”

Não	trabalho	para	ter	clientes	tenho	poder	trabalhar	se	pode	forçar	a	inteligência
1	1	2	1	2	1	1	1	0	0	0	0	0
1	0	0	0	0	0	0	1	1	1	1	2	1

transformando textos em dados

- ▶ Documento 1: “Não trabalho para ter clientes; tenho clientes para poder trabalhar.”
- ▶ Documento 2: “Não se pode forçar a inteligência a trabalhar.”

Não	trabalho	para	ter	clientes	tenho	poder	trabalhar	se	pode	forçar	a	inteligência
1	1	2	1	2	1	1	1	0	0	0	0	0
1	0	0	0	0	0	0	1	1	1	1	2	1

- ▶ O resultado é uma *matriz de termos-freqüências*. Dimensões: qtde. de documentos, qtde. de palavras (no exemplo acima: 2, 13).

transformando textos em dados

- ▶ Documento 1: “Não trabalho para ter clientes; tenho clientes para poder trabalhar.”
- ▶ Documento 2: “Não se pode forçar a inteligência a trabalhar.”

Não	trabalho	para	ter	clientes	tenho	poder	trabalhar	se	pode	forçar	a	inteligência
1	1	2	1	2	1	1	1	0	0	0	0	0
1	0	0	0	0	0	0	1	1	1	1	2	1

- ▶ O resultado é uma *matriz de termos-freqüências*. Dimensões: qtde. de documentos, qtde. de palavras (no exemplo acima: 2, 13).
- ▶ Ordem das palavras não importa (bag of words).

transformando textos em dados

- ▶ Documento 1: “Não trabalho para ter clientes; tenho clientes para poder trabalhar.”
- ▶ Documento 2: “Não se pode forçar a inteligência a trabalhar.”

Não	trabalho	para	ter	clientes	tenho	poder	trabalhar	se	pode	forçar	a	inteligência
1	1	2	1	2	1	1	1	0	0	0	0	0
1	0	0	0	0	0	0	1	1	1	1	2	1

- ▶ O resultado é uma *matriz de termos-freqüências*. Dimensões: qtde. de documentos, qtde. de palavras (no exemplo acima: 2, 13).
- ▶ Ordem das palavras não importa (bag of words).
- ▶ Cada documento é uma amostra. Cada palavra é um atributo.

transformando textos em dados

- ▶ Documento 1: “Não trabalho para ter clientes; tenho clientes para poder trabalhar.”
- ▶ Documento 2: “Não se pode forçar a inteligência a trabalhar.”

Não	trabalho	para	ter	clientes	tenho	poder	trabalhar	se	pode	forçar	a	inteligência
1	1	2	1	2	1	1	1	0	0	0	0	0
1	0	0	0	0	0	0	1	1	1	1	2	1

- ▶ O resultado é uma *matriz de termos-freqüências*. Dimensões: qtde. de documentos, qtde. de palavras (no exemplo acima: 2, 13).
- ▶ Ordem das palavras não importa (bag of words).
- ▶ Cada documento é uma amostra. Cada palavra é um atributo.
- ▶ Cada documento pode estar associado a um y , que pode ser categórico ou contínuo (“livro XYZ”, “2,57”, “português”, “autor XYZ”, etc).

transformando textos em dados

- ▶ Documento 1: “Não trabalho para ter clientes; tenho clientes para poder trabalhar.”
- ▶ Documento 2: “Não se pode forçar a inteligência a trabalhar.”

Não	trabalho	para	ter	clientes	tenho	poder	trabalhar	se	pode	forçar	a	inteligência
1	1	2	1	2	1	1	1	0	0	0	0	0
1	0	0	0	0	0	0	1	1	1	1	2	1

- ▶ O resultado é uma *matriz de termos-freqüências*. Dimensões: qtde. de documentos, qtde. de palavras (no exemplo acima: 2, 13).
- ▶ Ordem das palavras não importa (bag of words).
- ▶ Cada documento é uma amostra. Cada palavra é um atributo.
- ▶ Cada documento pode estar associado a um y , que pode ser categórico ou contínuo (“livro XYZ”, “2,57”, “português”, “autor XYZ”, etc).
- ▶ ...exatamente igual a todos os datasets (imaginários e reais) que discutimos até agora na disciplina

transformando textos em dados

- ▶ Documento 1: “Não trabalho para ter clientes; tenho clientes para poder trabalhar.”
- ▶ Documento 2: “Não se pode forçar a inteligência a trabalhar.”

Não	trabalho	para	ter	clientes	tenho	poder	trabalhar	se	pode	forçar	a	inteligência
1	1	2	1	2	1	1	1	0	0	0	0	0
1	0	0	0	0	0	0	1	1	1	1	2	1

- ▶ O resultado é uma *matriz de termos-freqüências*. Dimensões: qtde. de documentos, qtde. de palavras (no exemplo acima: 2, 13).
- ▶ Ordem das palavras não importa (bag of words).
- ▶ Cada documento é uma amostra. Cada palavra é um atributo.
- ▶ Cada documento pode estar associado a um y , que pode ser categórico ou contínuo (“livro XYZ”, “2,57”, “português”, “autor XYZ”, etc).
- ▶ ...exatamente igual a todos os datasets (imaginários e reais) que discutimos até agora na disciplina
- ▶ ...o que nos permite usar regressão, classificação, clusterização

- ▶ Identificar automaticamente o autor de um documento (classificação).

aplicações comuns

- ▶ Identificar automaticamente o autor de um documento (classificação).
- ▶ Agrupar um conjunto de documentos por autores ou tópicos (clusterização).

aplicações comuns

- ▶ Identificar automaticamente o autor de um documento (classificação).
- ▶ Agrupar um conjunto de documentos por autores ou tópicos (clusterização).
- ▶ Prever a “positividade” ou “negatividade” de um documento (regressão).

transformando textos em dados: textos curtos e longos

- ▶ É difícil comparar documentos de tamanhos muito diferentes.

transformando textos em dados: textos curtos e longos

- ▶ É difícil comparar documentos de tamanhos muito diferentes.
- ▶ Imagine 2 textos com as mesmas palavras, mas texto 1 tem dois parágrafos e texto 2 tem 400 páginas.

transformando textos em dados: textos curtos e longos

- ▶ É difícil comparar documentos de tamanhos muito diferentes.
- ▶ Imagine 2 textos com as mesmas palavras, mas texto 1 tem dois parágrafos e texto 2 tem 400 páginas.
- ▶ Solução: *normalizar* cada vetor.

transformando textos em dados: textos curtos e longos

- ▶ É difícil comparar documentos de tamanhos muito diferentes.
- ▶ Imagine 2 textos com as mesmas palavras, mas texto 1 tem dois parágrafos e texto 2 tem 400 páginas.
- ▶ Solução: *normalizar* cada vetor.
- ▶ L1: usar freqüências relativas em vez de absolutas.

transformando textos em dados: textos curtos e longos

- ▶ É difícil comparar documentos de tamanhos muito diferentes.
- ▶ Imagine 2 textos com as mesmas palavras, mas texto 1 tem dois parágrafos e texto 2 tem 400 páginas.
- ▶ Solução: *normalizar* cada vetor.
- ▶ L1: usar frequências relativas em vez de absolutas.
- ▶ L2: dividir cada elemento do vetor pelo comprimento do vetor.

transformando textos em dados: textos curtos e longos

- ▶ É difícil comparar documentos de tamanhos muito diferentes.
- ▶ Imagine 2 textos com as mesmas palavras, mas texto 1 tem dois parágrafos e texto 2 tem 400 páginas.
- ▶ Solução: *normalizar* cada vetor.
- ▶ L1: usar freqüências relativas em vez de absolutas.
- ▶ L2: dividir cada elemento do vetor pelo comprimento do vetor.
- ▶ Exemplo: $v = [3; 10; 4]$

transformando textos em dados: textos curtos e longos

- ▶ É difícil comparar documentos de tamanhos muito diferentes.
- ▶ Imagine 2 textos com as mesmas palavras, mas texto 1 tem dois parágrafos e texto 2 tem 400 páginas.
- ▶ Solução: *normalizar* cada vetor.
- ▶ L1: usar freqüências relativas em vez de absolutas.
- ▶ L2: dividir cada elemento do vetor pelo comprimento do vetor.
- ▶ Exemplo: $v = [3; 10; 4]$
- ▶ Comprimento de $v = \sqrt{3^2 + 10^2 + 4^2} = 11,38$

transformando textos em dados: textos curtos e longos

- ▶ É difícil comparar documentos de tamanhos muito diferentes.
- ▶ Imagine 2 textos com as mesmas palavras, mas texto 1 tem dois parágrafos e texto 2 tem 400 páginas.
- ▶ Solução: *normalizar* cada vetor.
- ▶ L1: usar freqüências relativas em vez de absolutas.
- ▶ L2: dividir cada elemento do vetor pelo comprimento do vetor.
- ▶ Exemplo: $v = [3; 10; 4]$
- ▶ Comprimento de $v = \sqrt{3^2 + 10^2 + 4^2} = 11,38$
- ▶ v normalizado = $\left[\frac{3}{11,38}; \frac{10}{11,38}; \frac{4}{11,38} \right] = [0,26; 0,89; 0,35]$

transformando textos em dados: textos curtos e longos

TF:

Não	trabalho	para	ter	clientes	tenho	poder	trabalhar	se	pode	forçar	a	inteligência
1	1	2	1	2	1	1	1	0	0	0	0	0
1	0	0	0	0	0	0	1	1	1	1	2	1

TF normalizada:

Não	trabalho	para	ter	clientes	tenho	poder	trabalhar	se	pode	forçar	a	inteligência
0,27	0,27	0,53	0,27	0,53	0,27	0,27	0,27	0	0	0	0	0
0,32	0	0	0	0	0	0	0,32	0,32	0,32	0,32	0,63	0,32

transformando textos em dados: o peso de cada palavra

- ▶ A matriz de termos-freqüências (TF) dá igual peso a todas as palavras.
- ▶ I.e., na hora de regredir/classificar/clusterizar, “a”, “de”, “para”, “não”, etc, têm peso igual a “trabalho”, “clientes”, “inteligência”, etc.
- ▶ Isso é ruim: palavras mais comuns (artigos, preposições, etc) deveriam ter peso menor, pois são menos discriminantes.
- ▶ Solução #1: ignorar palavras comuns como artigos e preposições (stopwords).
- ▶ É uma solução bastante comum. E vários pacotes de mineração de textos já vêm com listas de stopwords p/ diferentes idiomas.
- ▶ Ex.: NLTK (Python)

```
import nltk  
stopwords = nltk.corpus.stopwords.words('portuguese')
```

transformando textos em dados: o peso de cada palavra

- ▶ Solução #1: ignorar palavras comuns como artigos e preposições (stopwords).
- ▶ É uma solução bastante comum. E vários pacotes de mineração de textos já vêm com listas de stopwords p/ diferentes idiomas.
- ▶ Ex.: NLTK (Python)

```
import nltk  
stopwords = nltk.corpus.stopwords.words('portuguese')
```

transformando textos em dados: o peso de cada palavra

- ▶ Mas quão comum precisa ser a palavra p / ser considerada uma stopword?

transformando textos em dados: o peso de cada palavra

- ▶ Mas quão comum precisa ser a palavra p / ser considerada uma stopword?
- ▶ E o que fazer quando o problema concreto em mãos é um problema de *estilometria*? Nesses casos podemos estar interessados justamente nas stopwords.

transformando textos em dados: o peso de cada palavra

- ▶ Solução #2: usar *TFIDF* em vez de *TF*

transformando textos em dados: o peso de cada palavra

- ▶ IDF = inverse document frequency
- ▶ $IDF_t = \log \frac{N}{df_t}$
- ▶ N = qtde. de documentos
- ▶ df_t = qtde. de documentos em que o termo t aparece
- ▶ $TFIDF_{t,d} = TF_{t,d} \times IDF_t$
- ▶ O peso de cada termo (t):
- ▶ ... aumenta quando t aparece muitas vezes em poucos documentos
- ▶ ... diminui quando t aparece em muitos documentos
- ▶ ... diminui quando t aparece poucas vezes no documento
- ▶ I.e., o peso de t é maior quando t é mais *discriminante*.

transformando textos em dados: o peso de cada palavra

- ▶ Documento 1: “Não trabalho para ter clientes; tenho clientes para poder trabalhar.”
- ▶ Documento 2: “Não se pode forçar a inteligência a trabalhar.”

TF (normalizada):

Não	trabalho	para	ter	clientes	tenho	poder	trabalhar	se	pode	forçar	a	inteligência
0,27	0,27	0,53	0,27	0,53	0,27	0,27	0,27	0	0	0	0	0
0,32	0	0	0	0	0	0	0,32	0,32	0,32	0,32	0,63	0,32

TFIDF:

Não	trabalho	para	ter	clientes	tenho	poder	trabalhar	se	pode	forçar	a	inteligência
0	0,08	0,16	0,08	0,16	0,08	0,08	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0,1	0,1	0,1	0,19	0,1

transformando textos em dados: o peso de cada palavra

- ▶ Todo pacote decente de mineração de dados tem uma função p/ gerar TFIDF.
- ▶ Exemplo: scikit-learn (Python)

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
tfidf_maker = TfidfVectorizer()
```

```
documents = ['Não trabalho para ter clientes; tenho clientes para  
poder trabalhar.', 'Não se pode forçar a inteligência a trabalhar.']
```

```
tfidf = tfidf_maker.fit_transform(documents)
```

transformando textos em dados: pré-processamento

- ▶ Caracteres especiais (caracteres acentuados, símbolos, etc) freqüentemente resultam em erros.
- ▶ É preciso descobrir qual o encoding (codificação) dos documentos (UTF-8, Windows-1252, etc) e informar esse encoding p/ o pacote que vai carregar os dados.

transformando textos em dados: pré-processamento

- ▶ Às vezes não queremos diferenciar entre, digamos, “trabalho” e “trabalhar”.
- ▶ Nesse caso substituímos a palavra por sua raiz: “trabalh”.
- ▶ O nome disso é lematização (stemming).

transformando textos em dados: pré-processamento

- ▶ Às vezes não estamos interessados em determinadas expressões: “saia justa”, “processamento de linguagem natural”, etc.
- ▶ Nesses casos podemos escolher trabalhar com bigramas, trigramas, etc, em vez de unigramas.

minerando textos: similaridade do co-seno

- ▶ Como medir a similaridade entre dois documentos?

minerando textos: similaridade do co-seno

- ▶ Como medir a similaridade entre dois documentos?
- ▶ Passo 1: vetorizar cada documento (bag of words, normalização, TFIDF).

minerando textos: similaridade do co-seno

- ▶ Como medir a similaridade entre dois documentos?
- ▶ Passo 1: vetorizar cada documento (bag of words, normalização, TFIDF).
- ▶ Passo 2: computar o produto escalar dos dois vetores.

minerando textos: similaridade do co-seno

- ▶ Como medir a similaridade entre dois documentos?
- ▶ Passo 1: vetorizar cada documento (bag of words, normalização, TFIDF).
- ▶ Passo 2: computar o produto escalar dos dois vetores.
- ▶ Esse produto é o co-seno do ângulo entre os dois vetores. Quanto mais próximo de 0, mais diferentes os documentos; quanto mais próximo de 1, mais similares os documentos.

minerando textos: similaridade do co-seno

- ▶ Como medir a similaridade entre dois documentos?
- ▶ Passo 1: vetorizar cada documento (bag of words, normalização, TFIDF).
- ▶ Passo 2: computar o produto escalar dos dois vetores.
- ▶ Esse produto é o co-seno do ângulo entre os dois vetores. Quanto mais próximo de 0, mais diferentes os documentos; quanto mais próximo de 1, mais similares os documentos.
- ▶ (O co-seno varia, em tese, entre -1 e +1; mas quando se trata de documentos o co-seno é sempre não-negativo, pois o TF ou TFIDF não tem elementos negativos.)

minerando textos: similaridade do co-seno

- ▶ Como medir a similaridade entre dois documentos?
- ▶ Passo 1: vetorizar cada documento (bag of words, normalização, TFIDF).
- ▶ Passo 2: computar o produto escalar dos dois vetores.
- ▶ Esse produto é o co-seno do ângulo entre os dois vetores. Quanto mais próximo de 0, mais diferentes os documentos; quanto mais próximo de 1, mais similares os documentos.
- ▶ (O co-seno varia, em tese, entre -1 e +1; mas quando se trata de documentos o co-seno é sempre não-negativo, pois o TF ou TFIDF não tem elementos negativos.)
- ▶ Aplicação comum: text retrieval (similaridade entre expressão pesquisada e documentos disponíveis.).

minerando textos: análise de sentimentos

- ▶ Objetivo: medir “sentimento” de cada texto, ao longo de alguma escala: triste-feliz, satisfeito-insatisfeito, etc.

minerando textos: análise de sentimentos

- ▶ Objetivo: medir “sentimento” de cada texto, ao longo de alguma escala: triste-feliz, satisfeito-insatisfeito, etc.
- ▶ Manualmente: lista de palavras que denotam sentimento e respectivos scores. Score do documento é o score médio de suas palavras.

minerando textos: análise de sentimentos

- ▶ Objetivo: medir “sentimento” de cada texto, ao longo de alguma escala: triste-feliz, satisfeito-insatisfeito, etc.
- ▶ Manualmente: lista de palavras que denotam sentimento e respectivos scores. Score do documento é o score médio de suas palavras.
- ▶ Automaticamente: documentos pré-scored como dataset de treinamento e então regressão.

exercícios

- ▶ Notícias (vamos fazer juntos).
- ▶ Tuítes (vocês sozinhos).