Modelagem de Tópicos e Alocação Latente de Dirichlet (LDA) em Python

Susan Li
31 de maio de 2018 · 5 min de leitura

Crédito da foto: Pixabay

A modelagem de tópicos é um tipo de modelagem estatística para descobrir os "tópicos" abstratos que ocorrem em uma coleção de documentos. **Alocação de Dirichlet Latente** (LDA) é um exemplo de modelo de tópico e é usado para classificar texto em um documento para um tópico específico.Ele cria um tópico por modelo de documento e palavras por modelo de tópico, modelado como distribuições do Dirichlet.

Aqui vamos aplicar o LDA a um conjunto de documentos e dividi-los em tópicos. Vamos começar!

Os dados

O conjunto de dados que usaremos é uma lista de mais de um milhão de manchetes publicadas em um período de 15 anos e pode ser baixado do Kaggle .

```
importar pandas como pd

data = pd.read_csv ('abcnews-date-text.csv', error_bad_lines =
False);
data_text = data [['headline_text']]
data_text ['index'] = data_text.index
documents = data_text
```

Dê uma olhada nos dados.

```
imprimir (len (documentos))
imprimir (documentos [: 5])
```

1048575

	headline_text	index
0	aba decides against community broadcasting lic...	0
1	act fire witnesses must be aware of defamation	1
2	a g calls for infrastructure protection summit	2
3	air nz staff in aust strike for pay rise	3
4	air nz strike to affect australian travellers	4

figura 1

Pré-processamento de Dados

Vamos executar as seguintes etapas:

- **Tokenização** : **divida** o texto em frases e as frases em palavras. Minúsculas as palavras e remova a pontuação.
- Palavras com menos de 3 caracteres são removidas.
- Todas as **palavras irrelevantes** são removidas.
- As palavras são **lematizadas** - as palavras na terceira pessoa são alteradas para primeira pessoa e os verbos no passado e no futuro são alterados para o presente.
- As palavras são **derivadas** - as palavras são reduzidas à sua forma raiz.

Carregando bibliotecas gensim e nltk

```
importar gensim
de gensim.utils importar simple_preprocess
de gensim.parsing.preprocessing importar STOPWORDS
de nltk.stem importar WordNetLemmatizer, SnowballStemmer
de nltk.stem.porter importar *
import numpy como np
np.random.seed (2018)

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

[nltk_data] Fazendo o download do pacote wordnet para
[nltk_data] C: \ Users \ SusanLi \ AppData \ Roaming \ nltk_data...
[nltk_data] O pacote wordnet já está atualizado!

Verdade

Escreva uma função para executar etapas de pré-processamento de lematização e avanço no conjunto de dados .

```
def lemmatize_stemming (texto):
    retorna stemmer.stem (WordNetLemmatizer (). lemmatize (texto,
pos = 'v'))

def preprocess (texto):
    resultado = []
    para o token no gensim.utils.simple_preprocess (texto):
        se o token não estiver no
gensim.parsing.preprocessing.STOPWORDS e len (token)> 3:
```

```
            result.append (lemmatize_stemming (token))
    resultado de retorno
```

Selecione um documento para visualizar após o pré-processamento .

```
doc_sample = documentos [documentos ['index'] == 4310] .valores [0]
[0]

print ('documento original:')
words = []
para word em doc_sample.split (''):
    words.append (word)
print (words)
print ('\ n \ n documento tokenizado e lematizado:')
print (pré-processo (doc_sample))
```

documento original:

['chuva', 'ajuda', 'amortecer', 'incêndios florestais']

documento tokenizado e lematizado:

['chuva', 'ajuda', 'amortecer', 'bushfir']

Funcionou!

Pré-processe o texto do título, salvando os resultados como 'process_docs'

```
process_docs = documentos ['headline_text']. mapa (pré-
processo ) process_docs [: 10]
```

```
0            [decid, communiti, broadcast, licenc]
1                                [wit, awar, defam]
2          [call, infrastructur, protect, summit]
3                     [staff, aust, strike, rise]
4           [strike, affect, australian, travel]
5             [ambiti, olsson, win, tripl, jump]
6          [antic, delight, record, break, barca]
7    [aussi, qualifi, stosur, wast, memphi, match]
8            [aust, address, secur, council, iraq]
```

```
9                              [australia, lock, timet]
Name: headline_text, dtype: object
```

Figura 2

Saco de palavras no conjunto de dados

Crie um dicionário a partir de 'process_docs' contendo o número de vezes que uma palavra aparece no conjunto de treinamento.

```
dictionary = gensim.corpora.Dictionary (docs processados)

count = 0
para k, v no dictionary.iteritems ():
    print (k, v)
    count + = 1
    se count> 10:
        break
```

0 transmissão

1 communiti

2 decidir

3 licenc

4 awar

5 difamação

6 sagacidade

7 chamada

8 infra-estrutura

9 proteger

10 cimeira

Gensim filter_extremes

Filtrar tokens que aparecem em

- menos de 15 documentos (número absoluto) ou
- mais de 0,5 documentos (fração do tamanho total do corpus, número não absoluto).
- após as duas etapas acima, mantenha apenas os primeiros 100000 tokens mais frequentes.

```
dictionary.filter_extremes (no_below = 15, no_above = 0.5, keep_n =
100000)
```

Gensim doc2bow

Para cada documento, criamos um dicionário relatando quantas palavras e quantas vezes essas palavras aparecem. Salve isso em 'bow_corpus' e verifique nosso documento selecionado anteriormente.

```
bow_corpus = [dictionary.doc2bow (doc) para doc
emdocs processados] bow_corpus [4310]
```

[(76, 1), (112, 1), (483, 1), (3998, 1)]

Visualize Bag Of Words para nosso exemplo de documento pré-processado .

```
bow_doc_4310 = bow_corpus [4310]

para i no intervalo (len (bow_doc_4310)):
    print ("Word {} (\" {} \ ") aparece {} hora.". formato
(bow_doc_4310 [i] [0],
                                                dicionário
[bow_doc_4310 [i] [0 ]],
bow_doc_4310 [i] [1]))
```

A palavra 76 ("bushfir") aparece 1 vez.

A palavra 112 ("ajuda") aparece 1 vez.

A palavra 483 ("chuva") aparece 1 vez.

A palavra 3998 ("amortecer") aparece 1 vez.

TF-IDF

Crie o objeto de modelo tf-idf usando models.TfidfModel em 'bow_corpus' e salve-o em 'tfidf', aplique a transformação em todo o corpus e chame-o de 'corpus_tfidf'. Finalmente, visualizamos as pontuações do TF-IDF para o nosso primeiro documento.

```
de corpora import corpora, modelos

tfidf = models.TfidfModel (bow_corpus)
corpus_tfidf = tfidf [bow_corpus]

de pprint import pprint

para doc em corpus_tfidf:
    pprint (DOC)
    pausa
```

[(0, 0.5907943557842693),

(1, 0,3900924708457926),

(2, 0,49514546614015836),

(3, 0,5036078441840635)]

Executando o LDA usando o Bag of Words

Treine nosso modelo lda usando gensim.models.LdaMulticore e salve-o em 'lda_model'

```
lda_model = gensim.models.LdaMulticore (bow_corpus, num_topics =
10, id2word = dictionary, passa = 2, trabalhadores = 2)
```

Para cada tópico, exploraremos as palavras que ocorrem nesse tópico e seu peso relativo.

```
para idx, tópico em lda_model.print_topics (-1):
    print ('Tópico: {} \ nWords: {}'. formato (idx, tópico))
```

```
Topic: 0
Words: 0.035*"govern" + 0.024*"open" + 0.018*"coast" + 0.017*"tasmanian" + 0.017*"gold" + 0.014*"australia" + 0.013*"beat" + 0.
010*"win" + 0.010*"ahead" + 0.009*"shark"
Topic: 1
Words: 0.023*"world" + 0.014*"final" + 0.013*"record" + 0.012*"break" + 0.011*"lose" + 0.011*"australian" + 0.011*"leagu" + 0.0
11*"test" + 0.010*"australia" + 0.010*"hill"
Topic: 2
Words: 0.018*"rural" + 0.018*"council" + 0.015*"fund" + 0.014*"plan" + 0.013*"health" + 0.012*"chang" + 0.011*"nation" + 0.010
*"price" + 0.010*"servic" + 0.009*"say"
Topic: 3
Words: 0.025*"elect" + 0.022*"adelaid" + 0.012*"perth" + 0.011*"take" + 0.011*"say" + 0.010*"labor" + 0.010*"turnbul" + 0.009
*"vote" + 0.009*"royal" + 0.009*"time"
Topic: 4
Words: 0.032*"court" + 0.022*"face" + 0.020*"charg" + 0.020*"home" + 0.018*"tasmania" + 0.017*"murder" + 0.015*"trial" + 0.012
*"accus" + 0.012*"abus" + 0.012*"child"
Topic: 5
Words: 0.024*"countri" + 0.021*"hour" + 0.020*"australian" + 0.019*"warn" + 0.016*"live" + 0.013*"indigen" + 0.011*"call" + 0.0
09*"victorian" + 0.009*"campaign" + 0.008*"show"
Topic: 6
Words: 0.027*"south" + 0.024*"year" + 0.020*"interview" + 0.020*"north" + 0.019*"jail" + 0.018*"west" + 0.014*"island" + 0.013
*"australia" + 0.013*"victoria" + 0.010*"china"
Topic: 7
Words: 0.031*"queensland" + 0.029*"melbourn" + 0.018*"water" + 0.017*"claim" + 0.013*"hunter" + 0.012*"green" + 0.012*"resid" +
0.011*"darwin" + 0.010*"young" + 0.009*"plead"
Topic: 8
Words: 0.017*"attack" + 0.016*"kill" + 0.012*"victim" + 0.012*"violenc" + 0.010*"hobart" + 0.010*"rugbi" + 0.010*"secur" + 0.01
0*"say" + 0.009*"state" + 0.008*"domest"
Topic: 9
Words: 0.052*"polic" + 0.020*"crash" + 0.019*"death" + 0.017*"sydney" + 0.016*"miss" + 0.016*"woman" + 0.015*"die" + 0.015*"cha
```

Figura 3

Você consegue distinguir tópicos diferentes usando as palavras em cada tópico e seus pesos correspondentes?

Executando LDA usando TF-IDF

```
lda_model_tfidf = gensim.models.LdaMulticore (corpus_tfidf,
num_topics = 10, id2word = dictionary, passa = 2, trabalhadores =
4)

para idx, tópico em lda_model_tfidf.print_topics (-1):
    print ('Tópico: {} Word: {}'. formato (idx, tópico))
```

```
Topic: 0 Word: 0.008*"octob" + 0.006*"search" + 0.006*"miss" + 0.006*"inquest" + 0.005*"stori" + 0.005*"jam" + 0.004*"john" +
0.004*"harvest" + 0.004*"australia" + 0.004*"world"
Topic: 1 Word: 0.006*"action" + 0.006*"violenc" + 0.006*"thursday" + 0.005*"domest" + 0.005*"cancer" + 0.005*"legal" + 0.005*"u
nion" + 0.005*"breakfast" + 0.005*"school" + 0.004*"student"
Topic: 2 Word: 0.023*"rural" + 0.018*"govern" + 0.013*"news" + 0.012*"podcast" + 0.008*"grandstand" + 0.008*"health" + 0.007*"b
udget" + 0.007*"busi" + 0.007*"nation" + 0.007*"fund"
Topic: 3 Word: 0.030*"countri" + 0.028*"hour" + 0.009*"sport" + 0.008*"septemb" + 0.008*"wednesday" + 0.007*"commiss" + 0.006
*"royal" + 0.006*"updat" + 0.006*"station" + 0.005*"bendigo"
Topic: 4 Word: 0.014*"south" + 0.009*"weather" + 0.009*"north" + 0.008*"west" + 0.008*"coast" + 0.008*"australia" + 0.006*"eas
t" + 0.006*"queensland" + 0.006*"storm" + 0.005*"season"
Topic: 5 Word: 0.008*"monday" + 0.008*"august" + 0.006*"babi" + 0.005*"shorten" + 0.005*"hobart" + 0.004*"victorian" + 0.004*"d
onald" + 0.004*"safe" + 0.004*"scott" + 0.004*"donat"
Topic: 6 Word: 0.022*"interview" + 0.013*"market" + 0.009*"share" + 0.008*"cattl" + 0.008*"trump" + 0.008*"turnbul" + 0.007*"no
vemb" + 0.007*"michael" + 0.006*"australian" + 0.006*"export"
Topic: 7 Word: 0.019*"crash" + 0.014*"kill" + 0.009*"fatal" + 0.009*"dead" + 0.007*"die" + 0.007*"truck" + 0.007*"polic" + 0.00
6*"attack" + 0.006*"injur" + 0.006*"bomb"
Topic: 8 Word: 0.008*"drum" + 0.007*"abbott" + 0.007*"farm" + 0.006*"dairi" + 0.006*"asylum" + 0.006*"tuesday" + 0.006*"water"
+ 0.006*"labor" + 0.006*"say" + 0.005*"plan"
Topic: 9 Word: 0.017*"charg" + 0.014*"murder" + 0.011*"court" + 0.011*"polic" + 0.009*"woman" + 0.008*"assault" + 0.008*"jail"
+ 0.008*"alleg" + 0.007*"accus" + 0.007*"guilti"
```

Figura 4

Mais uma vez, você pode distinguir tópicos diferentes usando as palavras em cada tópico e seus pesos correspondentes?

Avaliação de desempenho classificando documento de amostra usando o modelo LDA Bag of Words

Vamos verificar onde nosso documento de teste seria classificado.

```
docs processados [4310]
```

['chuva', 'ajuda', 'amortecer', 'bushfir']

```
para o índice, a pontuação é classificada (lda_model [bow_corpus
[4310]], key = lambda tup: -1 * tup [1]):
    print ("\ nScore: {} \ t \ nTopic: {}". formato (pontuação,
lda_model.print_topic (índice, 10)))
```

```
Score: 0.41997694969177246
Topic: 0.017*"attack" + 0.016*"kill" + 0.012*"victim" + 0.012*"violenc" + 0.010*"hobart" + 0.010*"rugbi" + 0.010*"secur" + 0.01
0*"say" + 0.009*"state" + 0.008*"domest"

Score: 0.21999986469745636
Topic: 0.023*"world" + 0.014*"final" + 0.013*"record" + 0.012*"break" + 0.011*"lose" + 0.011*"australian" + 0.011*"leagu" + 0.0
11*"test" + 0.010*"australia" + 0.010*"hill"

Score: 0.21999594569206238
Topic: 0.027*"south" + 0.024*"year" + 0.020*"interview" + 0.020*"north" + 0.019*"jail" + 0.018*"west" + 0.014*"island" + 0.013
*"australia" + 0.013*"victoria" + 0.010*"china"

Score: 0.020009687170386314
Topic: 0.018*"rural" + 0.018*"council" + 0.015*"fund" + 0.014*"plan" + 0.013*"health" + 0.012*"chang" + 0.011*"nation" + 0.010
*"price" + 0.010*"servic" + 0.009*"say"

Score: 0.020008400082588196
Topic: 0.024*"countri" + 0.021*"hour" + 0.020*"australian" + 0.019*"warn" + 0.016*"live" + 0.013*"indigen" + 0.011*"call" + 0.0
09*"victorian" + 0.009*"campaign" + 0.008*"show"

Score: 0.02000494673848152
Topic: 0.031*"queensland" + 0.029*"melbourn" + 0.018*"water" + 0.017*"claim" + 0.013*"hunter" + 0.012*"green" + 0.012*"resid" +
0.011*"darwin" + 0.010*"young" + 0.009*"plead"

Score: 0.020004209131002426
Topic: 0.052*"polic" + 0.020*"crash" + 0.019*"death" + 0.017*"sydney" + 0.016*"miss" + 0.016*"woman" + 0.015*"die" + 0.015*"cha
rg" + 0.014*"shoot" + 0.013*"arrest"
```

Figura 5

Nosso documento de teste tem a maior probabilidade de fazer parte do tópico que nosso modelo atribuiu, que é a classificação precisa.

Avaliação de desempenho, classificando o documento de amostra usando o modelo LDA TF-IDF.

```
para índice, pontuação em ordenada (lda_model_tfidf [bow_corpus
[4310]], chave = lambda tup: -1 * tup [1]):
    print ("\ nScore: {} \ t \ nTopic: {}". formato (pontuação,
lda_model_tfidf.print_topic (índice, 10)))
```

```
Score: 0.44014573097229004
Topic: 0.014*"south" + 0.009*"weather" + 0.009*"north" + 0.008*"west" + 0.008*"coast" + 0.008*"australia" + 0.006*"east" + 0.00
6*"queensland" + 0.006*"storm" + 0.005*"season"

Score: 0.3998423218727112
Topic: 0.023*"rural" + 0.018*"govern" + 0.013*"news" + 0.012*"podcast" + 0.008*"grandstand" + 0.008*"health" + 0.007*"budget" +
0.007*"busi" + 0.007*"nation" + 0.007*"fund"

Score: 0.02000250481069088
Topic: 0.006*"action" + 0.006*"violenc" + 0.006*"thursday" + 0.005*"domest" + 0.005*"cancer" + 0.005*"legal" + 0.005*"union" +
0.005*"breakfast" + 0.005*"school" + 0.004*"student"

Score: 0.020002111792564392
Topic: 0.008*"drum" + 0.007*"abbott" + 0.007*"farm" + 0.006*"dairi" + 0.006*"asylum" + 0.006*"tuesday" + 0.006*"water" + 0.006
*"labor" + 0.006*"say" + 0.005*"plan"

Score: 0.020001791417598724
Topic: 0.030*"countri" + 0.028*"hour" + 0.009*"sport" + 0.008*"septemb" + 0.008*"wednesday" + 0.007*"commiss" + 0.006*"royal" +
0.006*"updat" + 0.006*"station" + 0.005*"bendigo"

Score: 0.02000163123011589
Topic: 0.008*"octob" + 0.006*"search" + 0.006*"miss" + 0.006*"inquest" + 0.005*"stori" + 0.005*"jam" + 0.004*"john" + 0.004*"ha
rvest" + 0.004*"australia" + 0.004*"world"

Score: 0.020001478493213654
Topic: 0.008*"monday" + 0.008*"august" + 0.006*"babi" + 0.005*"shorten" + 0.005*"hobart" + 0.004*"victorian" + 0.004*"donald" +
0.004*"safe" + 0.004*"scott" + 0.004*"donat"
```

Figura 6

Nosso documento de teste tem a maior probabilidade de fazer parte do tópico que nosso modelo atribuiu, que é a classificação precisa.

Modelo de teste em documento invisível

```
unseen_document = 'Como um acordo do Pentágono se tornou uma crise
de identidade para o Google'
bow_vector = dictionary.doc2bow (pré-processo (unseen_document))

para índice, pontue em ordenado (lda_model [bow_vector], key =
lambda tup: -1 * tup [1]):
    print ("Pontuação: {} \ t Tópico: {}". formato (pontuação,
lda_model.print_topic (index 5)))
```

```
Score: 0.3500000238418579       Topic: 0.017*"attack" + 0.016*"kill" + 0.012*"victim" + 0.012*"violenc" + 0.010*"hobart"
Score: 0.34998345375061035      Topic: 0.025*"elect" + 0.022*"adelaid" + 0.012*"perth" + 0.011*"take" + 0.011*"say"
Score: 0.18333327770233154      Topic: 0.052*"polic" + 0.020*"crash" + 0.019*"death" + 0.017*"sydney" + 0.016*"miss"
Score: 0.01667369343340397      Topic: 0.035*"govern" + 0.024*"open" + 0.018*"coast" + 0.017*"tasmanian" + 0.017*"gold"
```

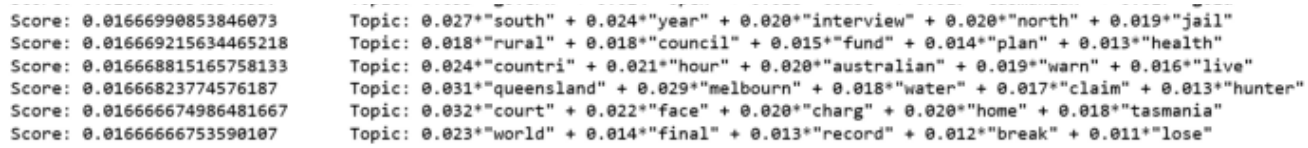

Figura 7

O código fonte pode ser encontrado no Github . Estou ansioso para ouvir quaisquer comentários ou perguntas.

Referência:

Udacity - PNL