Project 1 - Topic Modeling

Adrian Ispas
Grupa 507 - Master IA

Informații generale

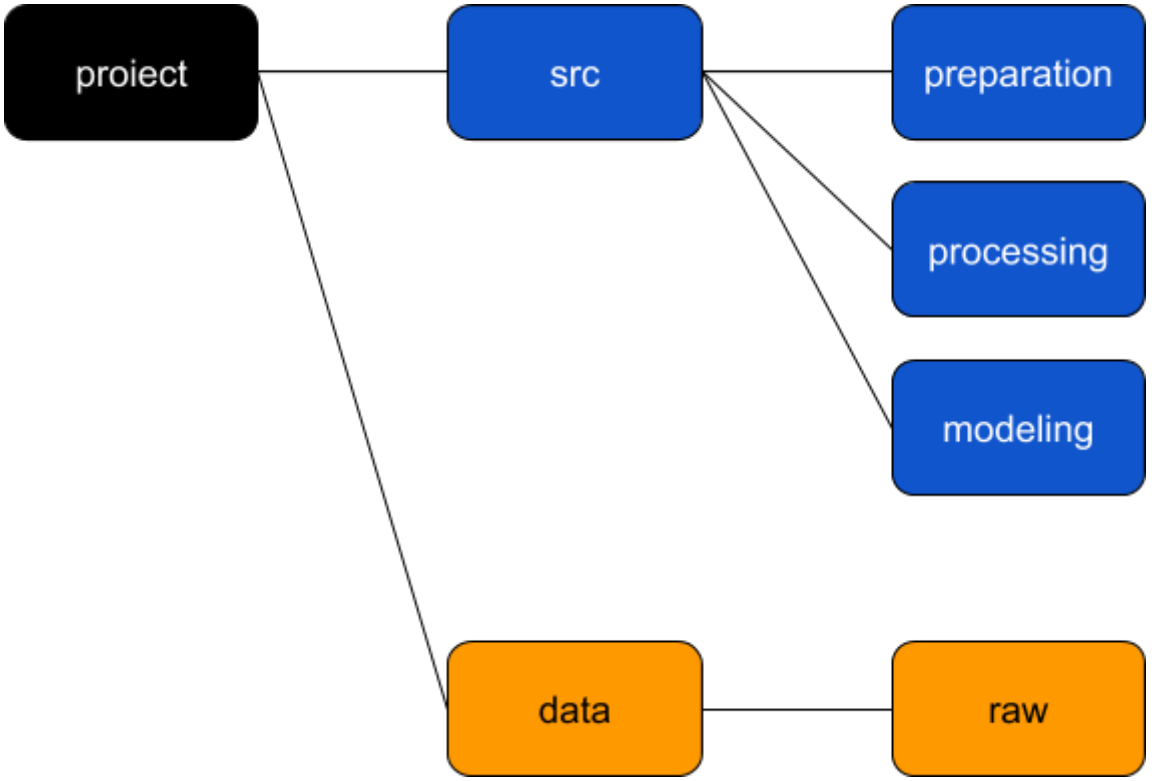
Prima variantă a proiectului (cea care nu conține extra-urile asociate) este descrisă de-a lungul acestui document până la capitolul **Extra.**

Implementarea până în acest punct se poate găsi pe github cu tagul asociat **v1.0.0** la următorul link: https://github.com/adiIspas/Topic-Modeling-System/tree/v1.0.0

A doua parte a proiectului este descrisă începând cu capitolul **Extra** în care menționez tot ceea ce s-a adăugat pe parcursul implementări punctelor extra.

Această implementare poate fi găsită atât pe github cu tagul asociat **v2.0.0** la următorul link: https://github.com/adiIspas/Topic-Modeling-System/tree/v2.0.0 dar și atașată mail-ului trimis.

Descriere structură proiect



1.1 src

Înglobează fişierele python în care se realizează extragerea datelor, procesarea, descrierea modelului şi antrenarea lui.

1.1.1 preparation

Citirea conţinutul fişierele stocate în folderul **/data/raw**.

1.1.2 processing

Prelucrarea datelor citite din fişiere şi punerea lor în format corespunzător.

Spre exemplu pentru documentele

```
document_1.txt: I had a peanuts butter sandwich for breakfast.
document_2.txt: I like to eat almonds, peanuts and walnuts.
document_3.txt: My neighbor got a little dog yesterday.
document_4.txt: Cats and dogs are mortal enemies.
document_5.txt: You mustn't feed peanuts to your dog.
```

vom avea următoare structură.

```
[0, 1, 2, 3]
[4, 5, 6, 0, 7]
[8, 9, 10, 11, 12]
[13, 14, 15, 16]
[17, 0, 11]
```

Din fiecare document au fost extrase toate cuvintele creând astfel un vocabular, iar pentru fiecare cuvânt am asociat un indice unic. Stopword-urile au fost eliminate.

1.1.3 modeling

Definirea și antrenarea modelului LDA.

1.2 data

Stocarea fișierelor cu texte

1.2.1 raw

Stocarea fișierelor originale cu text, în cazul nostru cinci fișiere text.

Descrierea implementării

2.1 Extragerea datelor

În ceea ce priveşte citirea documentelor, aceasta se poate face intermediul funcţiei

read_documents(directory)

în care citim conţinutul fiecărui fişier din directorul respectiv.
Ca rezultat vom primi o lista de documente cu conţinutul asociat acestora.

2.2 Procesarea datelor

După citirea conţinutului fişierelor vom crea un vocabular ordonat alfabet cu cuvintele unice întâlnite în textele din documente şi fără stopword-uri. Această operaţie se realizează cu funcţia

create_vocabulary(documents)

Ca rezultat vom primi

- vocabularul sortat alfabetic
- maparea de la cuvânt la id
- maparea de la id la cuvânt
- dimensiunea vocabularului

Odată creat vocabularul putem mapa fiecare cuvânt din fiecare document la un id unic cu funcţia

generate_data_from_documents(documents, word_id)

Exemplu de rezultat al funcţiei poate fi văzut în secţiunea **1.1.2**.

2.3 Implementarea modelului

Din implementarea modelului putem evidenţia definirea variabilelor:

- **phi**
- **theta**
- **Z**
- **W**

Phi: pentru fiecare topic avem o distribuție Dirichlet peste vocabular.

```
# Word distribution for each topics
self.phi = pm.Container(
    [pm.CompletedDirichlet("phi_%s" % i,
                           pm.Dirichlet("pphi_%s" % i, theta=self.beta)) for i
in range(self.topics)])
```

Exemplu de rezultat este prezentat în continuare. Fiecare topic conținea ponderea pentru fiecare cuvânt în descrierea lui (în cazul de față două topicuri și 18 cuvinte în vocabular).

```
[[ 0.07569972  0.06319251  0.02777946  0.05403736  0.00033168  0.12001489
   0.06182098  0.0176073   0.03833226  0.01711657  0.02122876  0.0403591
   0.00240565  0.02607294  0.00496632  0.14187776  0.26158966  0.02556706]]
[[ 0.12316364  0.16923682  0.07721261  0.0286569   0.03924738  0.01753802
   0.00380988  0.01043237  0.24955461  0.03503004  0.02332184  0.04151092
   0.00813432  0.0166026   0.0166982   0.072656    0.04557552  0.02161832]]
```

Theta: pentru fiecare document avem o distribuție Dirichlet peste topicuri.

```
# Topic distribution for each document
self.theta = pm.Container(
    [pm.CompletedDirichlet("theta_%s" % i,
                           pm.Dirichlet("ptheta_%s" % i,
theta=self.alpha)) for i in range(self.docs)])
```

Exemplu de rezultat este prezentat în continuare. Fiecare document are asociata o probabilitate pentru fiecare topic (în cazul de față două topicuri).

```
[[ 0.51976407  0.48023593]]
[[ 0.53332427  0.46667573]]
[[ 0.19418843  0.80581157]]
[[ 0.72806479  0.27193521]]
[[ 0.39123421  0.60876579]]
```

Z: pentru fiecare cuvânt din fiecare document avem o distribuție multinomială (categorică) de asignarea a lui la un topic.

```
# Select a topic for each word from each document
self.Z = pm.Container(
    [pm.Categorical("z_%s" % d,
                    p=self.theta[d],
```

```
                    size=self.wd[d],
                    value=np.random.randint(self.topics,
size=self.wd[d])) for d in range(self.docs)])
```

Exemplu de rezultat este prezentat în continuare. Fiecare cuvânt (care nu este stopword) din fiecare document are un topic asociat (în cazul de faţă două topicuri).

```
[1 1 1 1]
[0 0 1 0 1]
[1 1 1 1 1]
[1 0 0 0]
[1 1 1]
```

W: identificăm fiecare cuvânt din fiecare document cu indici creaţi în vocabular.

```
# Word distribution associated with selected topic
self.W = pm.Container(
    [pm.Categorical("w_%s,%s" % (d, i),
                    p=pm.Lambda("phi_z_%s_%s" % (d, i), lambda
z=self.Z[d][i], phi=self.phi: phi[z]),
                    value=self.data[d][i], observed=True) for d in
range(self.docs) for i in range(self.wd[d])])
```

Exemplu de rezultat este prezentat în continuare*. Fiecare cuvânt unic identificat, conform vocabularului, are un indice asociat care este folosit mai departe în fiecare document.

*În fapt rezultatul W-ului este o lista de indici dar pentru o mai bună vizualizare a rezultatului am ales să-l prezint ca o grupare de indici a cuvintelor pentru fiecare document. Această grupare reprezentând de fapt valoarea variabilei **data.**

```
[0, 2, 3, 4]
[5, 6, 7, 0, 8]
[9, 10, 11, 1, 12]
[13, 14, 15, 16]
[17, 0, 1]
```

model: definirea modelului cu parametrii **theta, phi, Z** şi **W** pe care îl pasăm mai departe intr-o instanţă **MCMC,** instanţă pe care o vom folosi pentru a apela funcţia **fit.**

```
# Create the model
self.model = pm.Model([self.theta, self.phi, self.Z, self.W])
self.mcmc = pm.MCMC(self.model)
```

Flow-ul

În continuare vom descrie flow-ul aplcaţiei şi cum ajungem la rezultatele dorite.

Pasul 1: Definim calea către fişierele text de analizat, numărul de topicuri estimative şi citim datele din fişiere.

```
documents_path = '~/data/raw'
documents = Reader.read_documents(documents_path)
number_of_topics = 2
```

Pasul 2: Instanţiem un obiect **Processor** cu limba fişierelor text pentru a ştii ce stopword-uri trebuie eliminate din texte şi realizăm vocabularul pe baza documentelor.

```
processor = Processor("english")

sorted_vocabulary, word_id, id_word, vocabulary_size =
processor.create_vocabulary(documents)
```

Pasul 3: Generam datele în formatul necesar pentru antrenarea modelului pe baza documentelor citite.

```
data = Processor.generate_data_from_documents(documents, word_id)
```

Pasul 4: Instanţiem modelul şi îl antrenăm.

```
lda_model = LDA(data, number_of_topics, vocabulary_size)
lda_model.fit()
```

Pasul 5: Pentru afişarea rezultatelor vom utiliza funcţiile **show_topic_words** care primeşte ca parametru maparea de la id-ul cuvântului la cuvânt, **show_document_topics, show_topic_for_word_in_document** şi **show_word_distribution_in_topics**.

Integrarea cu un sistem de Information Retrieval

Pentru a facilita integrarea cu sistem de regăsire a informației punem la dispoziție două metode în model care pot fi utilizate în acest scop.

Prima metoda este cea de obținere a cuvintelor pentru fiecare topic

get_topics_words(id_word)

care primește ca parametru maparea dintre id și cuvântul asociat acestuia și întoarce o listă de topicuri cu cuvintele cele mai reprezentative pentru topicul respectiv.

Exemplu de rezultat

```
{
 0: ['eat', 'yesterday', 'mortal', 'got', 'dogs', 'dog', 'like',
'walnuts', 'neighbor', 'cats'],

 1: ['little', 'sandwich', 'breakfast', 'dogs', 'got', 'walnuts',
'peanuts', 'mortal']
}
```

unde fiecare lista reprezintă un cuvintele dintr-un topic.

Cea de-a doua metoda este folosită pentru a obține pentru fiecare document topicurile reprezentative

get_documents_topics(threshold)

care primește ca parametru un threshold pe care un topic trebuie să-l aibă pentru a fi reprezentativ pentru documentul respectiv.

Exemplu de rezultat

```
{
 0: [0, 1],
 1: [0, 1],
 2: [1],
 3: [0, 1],
 4: [1]
}
```

unde fiecare listă reprezintă topicurile reprezentative pentru un document.

Astfel, putem adăuga foarte uşor feature-uri noi sistemului de regăsire a informaţiei făcând căutarea nu numai după cuvintele din query care se găsesc în conţinutul documentului ci şi după topicuri.

Feature util în situaţii în care anumite cuvinte nu se află într-un anumite document dar documentul per ansamblu se află într-un topic comun cu cuvintele căutate de noi.

Extra

Similaritatea între documente

În ceea ce priveşte similaritatea între documente putem utiliza pe de-o parte reprezentarea lor peste mulţimea de cuvinte din vocabular, însă mai eficient putem utiliza reprezentarea documentelor peste mulţimea de topicuri, reducând astfel dimensiunea de reprezentare a fiecărui document.

În implementare putem folosi funcţia

documents_similarity(threshold)

care primeşte ca parametru un threshold ca doua documente să fie considerate similare şi întoarce o lista de intrări, unde fiecare intrare este o pereche de două documente identificate prin indexul lor şi similaritatea dintre ele.

Formula utilizată este descrisă mai jos:

$$\text{document-similarity}_{d,f} = \sum_{k=1}^{K} \left(\sqrt{\widehat{\theta}_{d,k}} - \sqrt{\widehat{\theta}_{f,k}} \right)^2$$

Un exemplu de rezultat cu un threshold fixat la 0.8 este următorul:

```
[0, 3, 0.9091288677864572]
[2, 3, 0.8786451515330838]
[3, 0, 0.9091288677864572]
[3, 2, 0.8786451515330838]
```

Adăugarea unui nou document

Pentru adăugarea unui nou document putem utiliza următorul algoritm:

One iteration of mean field variational inference for LDA

(1) For each topic k and term v:

$$\lambda_{k,v}^{(t+1)} = \eta + \sum_{d=1}^{D}\sum_{n=1}^{N} 1(w_{d,n} = v)\phi_{n,k}^{(t)}. \tag{8}$$

(2) For each document d:

(a) Update γ_d:

$$\gamma_{d,k}^{(t+1)} = \alpha_k + \sum_{n=1}^{N} \phi_{d,n,k}^{(t)}. \tag{9}$$

(b) For each word n, update $\vec{\phi}_{d,n}$:

$$\phi_{d,n,k}^{(t+1)} \propto \exp\left\{\Psi(\gamma_{d,k}^{(t+1)}) + \Psi(\lambda_{k,w_n}^{(t+1)}) - \Psi\left(\sum_{v=1}^{V}\lambda_{k,v}^{(t+1)}\right)\right\}, \tag{10}$$

where Ψ is the digamma function, the first derivative of the log Γ function.

FIGURE 5. One iteration of mean field variational inference for LDA. This algorithm is repeated until the objective function in Eq. (6) converges.

Correlated topic model

Pentru topicuri corelate putem utiliza algoritmul:

(1) Draw $\eta \mid \{\mu, \Sigma\} \sim N(\mu, \Sigma)$.
(2) For $n \in \{1, \ldots, N\}$:
  (a) Draw topic assignment $Z_n \mid \eta$ from Mult$(f(\eta))$.
  (b) Draw word $W_n \mid \{z_n, \beta_{1:K}\}$ from Mult(β_{z_n}).

The function that maps the real-vector η to the simplex is

$$f(\eta_i) = \frac{\exp\{\eta_i\}}{\sum_j \exp\{\eta_j\}}. \tag{15}$$

Dynamic topic model

Pentru topicuri dinamice putem utilza algoritmul:

(1) Draw topics $\vec{\pi}_t \mid \vec{\pi}_{t-1} \sim N(\vec{\pi}_{t-1}, \sigma^2 I)$
(2) For each document:
    (a) Draw $\theta_d \sim \text{Dir}(\vec{\alpha})$
    (b) For each word:
        (i) Draw $Z \sim \text{Mult}(\theta_d)$
        (ii) Draw $W_{t,d,n} \sim \text{Mult}(f(\vec{\pi}_{t,z}))$.