mamrou / short_text_topic_modeling

Code Issues 3 Pull requests Actions Projects Wiki Security Insights

master

short_text_topic_modeling / notebook_sttm_example.ipynb

mamrou Add files via upload

1 contributor

Raw Blame

906 lines (906 sloc) 29.3 KB

In [1]:
```python
from preprocessing import tokenize, export_to_csv
from gsdmm import MovieGroupProcess
from topic_allocation import top_words, topic_attribution
from visualisation import plot_topic_notebook, save_topic_html
from sklearn.datasets import fetch_20newsgroups


import pickle
import matplotlib as plt
import pandas as pd
import numpy as np
import ast
```

Topic Modeling on 20NewsGroups

Data selection

In [2]:
```python
cats = ['talk.politics.mideast', 'comp.windows.x', 'sci.space']

newsgroups_train = fetch_20newsgroups(subset='train', remove=('heade
rs', 'footers', 'quotes'), categories=cats)
newsgroups_train_subject = fetch_20newsgroups(subset='train', catego
ries=cats)

data = newsgroups_train.data
data_subject = newsgroups_train_subject.data

targets = newsgroups_train.target.tolist()
target_names = newsgroups_train.target_names
```

In [3]:
```python
# Let's see if our topics are evenly distributed
df_targets = pd.DataFrame({'targets': targets})
order_list = df_targets.targets.value_counts()
order_list
```

Out[3]:
```
1    593
0    593
2    564
Name: targets, dtype: int64
```

In [5]:
```python
def extract_first_sentence(data_subject):
    list_first_sentence = []
    for text in data:
        first_sentence = text.split(".")[0].replace("\n", "")
        list_first_sentence.append(first_sentence)
    return list_first_sentence



def extract_subject(data):
    c = 0
    s = "Subject:"
    list_subjects = []
    for new in data_subject:
        lines = new.split("\n")
        b = 0 # Loop out at the first "Subject:", they may be severa
l and we want first one only
```

```
t and we want first one only
        for line in lines:
            if s in line and b == 0:
                subject = " ".join(line.split(":")[1:]).strip()
                subject = subject.replace('Re', '').strip()
                list_subjects.append(subject)
                c += 1
                b = 1
    return list_subjects


def concatenate(list_first_sentence, list_subjects):
    list_docs = []
    for i in range(len(list_first_sentence)):
        list_docs.append(list_subjects[i] + " " + list_first_sentenc
e[i])
    return list_docs


list_first_sentence = extract_first_sentence(data)
list_subjects = extract_subject(data_subject)
list_docs = concatenate(list_first_sentence, list_subjects)
```

```
['Elevator to the top floor Reading from a Amoco Performance Product
s data sheet, theirERL-1906 resin with T40 carbon fiber reinforcemen
t has a compressivestrength of 280,000 psi', 'Title for XTerm Yet ag
ain,the escape sequences you are speaking about here are non standar
d anddangerous', 'From Israeli press. Madness. Before getting excite
d and implying that I am postingfabrications, I would suggest the re
aders to consult thenewspaper in question', 'Accounts of Anti-Armeni
an Human Right Violations in Azerbaijan #011 Accounts of Anti-Armeni
an Human Right Violations in Azerbaijan #011                 Prelude
to Current Events in Nagorno-Karabakh        +----------------------
--------------------------------+        |
|        |   "Right, we should slaughter the Armenians!" and     |
|    "There\'s no need to be afraid, all of Moscow is    |        |
behind us', "How many israeli soldiers does it take to kill a 5 yr o
ld child? Probably not--he's just singing someone else's opera"]
```

In [6]:

```
df = pd.DataFrame(columns=['content', 'topic_id', 'topic_true_name'
])
df['content'] = list_docs
df['topic_id'] = targets

def true_topic_name(x, target_names):
    return target_names[x].split('.')[-1]

df['topic_true_name'] = df['topic_id'].apply(lambda x: true_topic_na
me(x, target_names))
df.head()
```

Out[6]:

	content	topic_id	topic_true_name
0	Elevator to the top floor Reading from a Amoco...	1	space
1	Title for XTerm Yet again,the escape sequences...	0	x
2	From Israeli press. Madness. Before getting ex...	2	mideast
3	Accounts of Anti-Armenian Human Right Violatio...	2	mideast
4	How many israeli soldiers does it take to kill...	2	mideast

Tokenization & preprocessing

In [7]:
```
tokenized_data = tokenize(df, form_reduction='stemming', predict=Fal
se)
```

```
2019-08-22 18:48:35,265 :: [INFO] :: ---- Tokenization started ----
2019-08-22 18:48:35,266 :: [INFO] :: Initializing the preprocessin
g...
2019-08-22 18:48:35,737 :: [INFO] :: Spacy tokenization...
2019-08-22 18:48:52,273 :: [INFO] :: Stop words and one character wo
rds removing...
2019-08-22 18:48:52,289 :: [INFO] :: Stemming...
2019-08-22 18:48:52,788 :: [INFO] :: Remove numeric and empty...
2019-08-22 18:48:52,878 :: [INFO] :: Removing unique tokens...
2019-08-22 18:48:54,072 :: [INFO] :: ---- Tokenization completed ---
-
```

In [8]:
```
tokenized_data[['content', 'tokens', 'topic_true_name']].head()
```

Out[8]:

	content	tokens	topic_true_name
0	Elevator to the top floor Reading from a Amoco...	[read, perform, product, data, t, carbon, fibe...	space
1	Title for XTerm Yet again,the escape sequences...	[titl, xterm, escap, sequenc, speak, non, stan...	x
2	From Israeli press. Madness. Before getting ex...	[isra, press, mad, get, excit, impli, suggest,...	mideast
4	How many israeli soldiers does it take to kill...	[isra, soldier, kill, yr, old, child, probabl]	mideast
5	NEWS YOU MAY HAVE MISSED, Apr 20 NEWS YOU MAY ...	[news, miss, apr, news, miss, apr, ,...	mideast

In [9]:
```
print("Max number of token:", np.max(tokenized_data.nb_token))
print("Mean number of token:", round(np.mean(tokenized_data.nb_token
),2))

# Input format for the model : list of strings (list of tokens)
docs = tokenized_data['tokens'].tolist()
vocab = set(x for doc in docs for x in doc)
n_terms = len(vocab)

print("Voc size:", n_terms)
print("Number of documents:", len(docs))
```

```
Max number of token: 29
Mean number of token: 9.46
Voc size: 2126
Number of documents: 1705
```

Training

Training

In [ ]:
```
# Train a new model

# Init of the Gibbs Sampling Dirichlet Mixture Model algorithm
mgp = MovieGroupProcess(K=10, alpha=0.1, beta=0.1, n_iters=30)

vocab = set(x for doc in docs for x in doc)
n_terms = len(vocab)
n_docs = len(docs)

# Fit the model on the data given the chosen seeds
y = mgp.fit(docs, n_terms)

# Save model
with open('dumps/trained_models/model_v2.model', "wb") as f:
    pickle.dump(mgp, f)
    f.close()
```

In [10]:
```
# Load the model used in the post
filehandler = open('dumps/trained_models/model_v1.model', 'rb')
mgp = pickle.load(filehandler)
```

In [11]:
```
doc_count = np.array(mgp.cluster_doc_count)
print('Number of documents per topics :', doc_count)
print('*'*20)

# Topics sorted by document inside
top_index = doc_count.argsort()[-10:][::-1]
print('Most important clusters (by number of docs inside):', top_ind
ex)
print('*'*20)


# Show the top 5 words by cluster, it helps to make the topic_dict b
elow
top_words(mgp.cluster_word_distribution, top_index, 5)
```

```
Number of documents per topics : [130 193 151 145 306 140 139 251 11
9 131]
********************
Most important clusters (by number of docs inside): [4 7 1 2 3 5 6 9
0 8]
********************
Cluster 4 : [('problem', 64), ('window', 60), ('xr', 55), ('server',
49), ('run', 47)]
-----------------------------
Cluster 7 : [('isra', 116), ('israel', 56), ('hezbollah', 40), ('exp
ans', 34), ('terror', 31)]
-----------------------------
Cluster 1 : [('motif', 47), ('widget', 44), ('need', 34), ('progra
m', 30), ('window', 22)]
-----------------------------
Cluster 2 : [('moon', 40), ('billion', 34), ('year', 29), ('race', 2
1), ('long', 19)]
-----------------------------
Cluster 3 : [('space', 70), ('station', 28), ('vandal', 28), ('sky',
28), ('design', 21)]
-----------------------------
Cluster 5 : [('armenian', 89), ('turkish', 45), ('armenia', 34), ('m
uslim', 27), ('genocid', 26)]
```

```
-----------------------------
Cluster 6 : [('space', 58), ('news', 26), ('mine', 22), ('time', 1
9), ('commerci', 18)]
-----------------------------
Cluster 9 : [('orbit', 39), ('dc', 24), ('comet', 21), ('temporari',
19), ('jupit', 19)]
-----------------------------
Cluster 0 : [('israel', 26), ('center', 20), ('orion', 17), ('zionis
m', 16), ('polici', 16)]
-----------------------------
Cluster 8 : [('space', 52), ('faq', 43), ('archiv', 26), ('questio
n', 24), ('modifi', 22)]
-----------------------------
```

In [12]:

```
# Must be hand made so the topic names match the above clusters rega
rding their content
topic_dict = {}
topic_names = ['x',
               'mideast',
               'x',
               'space',
               'space',
               'mideast',
               'space',
               'space',
               'mideast',
               'space']

for i, topic_num in enumerate(top_index):
    topic_dict[topic_num]=topic_names[i]

df_pred = topic_attribution(tokenized_data, mgp, topic_dict, thresho
ld=0.4) # threshold can be modified to improve the confidence of the
topics
```

```
2019-08-22 18:49:03,763 :: [INFO] :: ---- Topic allocation started -
---
2019-08-22 18:49:03,765 :: [INFO] :: Topic ID attribution...
2019-08-22 18:49:04,330 :: [INFO] :: Topic ID probability computin
g...
2019-08-22 18:49:04,880 :: [INFO] :: Applying confidence threshold a
nd topic names matching...
2019-08-22 18:49:05,082 :: [INFO] :: ---- Topic allocation completed
----
```

In [13]:

```
#pd.set_option('display.max_columns', None)
```