Judul Praktikum: Pemodelan Topik dengan Natural Language Processing

Materi Praktikum:

1. Pendahuluan

a. Topik Modeling

- Topic modeling adalah seperangkat teknik yang bertujuan untuk menemukan dan memberikan keterangan (annotate) pada kumpulan dokumen-dokumen dengan informasi tematik.
- Topic modeling adalah sekumpulan metode yang bertujuan menganalisis kata (words) dari dokumen- dokumen untuk menemukan tema –tema didalamnya, dan bagaimana tema-tema tersebut terkoneksi satu sama lain dan bagaimana berubah setiap waktu.
- Jumlah topic modeling ditentukan sebelumnya.
- Topic modelling juga dapat digunakan sebagai teknik reduksi dimensi.
- Topic modeling seperti clustering, tidak memerlukan pelabelan terlebih dahulu, tetapi jumlah topic bisa ditentukan.
- Tipe-tipe Topic Modeling:
    - Linier Algebra Based (LSA)
    - Probabilistic Modeling Based (pLSA, LDA, Random Projection)

Latent Semantic Analysis (LSA)

- LSA atau disebut juga Latent Semantic Indexing
- LSA merupakan pendekatan statistik
- LSA adalah metode untuk mengekstraksi dan merepresentasikan makna penggunaan kontekstual dari kata-kata dengan perhitungan statistic yang diterapkan pada korpus teks yang besar.

LDA

- LDA atau Latent Dirichlet Allocation
- Menggunakan pendekatan probabilistik
- LDA adalah model probabilitas generatif yang dirancang untuk mengidentifikasi topik utama yang muncul dalam koleksi dokumen atau korpus teks.

2. Praktikum

Install library

```
pip install regex matplotlib sastrawi xlsxwriter nltk
#import modul
import pandas as pd
```

```
import numpy as np
import re
import re as reg
import matplotlib.pyplot as plt
%matplotlib inline
```

Read Data:

```
data=pd.read_csv('Kasus1.csv', delimiter=';')

data
```

Output:

	screen_name	username	user_id	tweet_id	tweet_url	timestamp	timestamp_epochs	text
0	BimaBurhanuddin	Burhanuddin Bima	3058961791	1,12E+18	/BimaBurhanuddin/status/1124100611925000193	02/05/2019 23:57	1556841445	Apa yg bisa diharapkan dari politisi gaya pela...
1	ngowatchindo	Waw Lam'alif	8,95E+17	1,12E+18	/ngowatchindo/status/1124099972046163968	02/05/2019 23:54	1556841292	Kalau sy jadi @DennyJA_WORLD , Syahganda aka...
2	socmed_anak	AnakSocmed	174945628	1,12E+18	/socmed_anak/status/1124098250083131393	02/05/2019 23:48	1556840882	Di @PDemokrat banyak org2 yg punya akal sehat ...

Bersihkan Data:

```
#Preprocessing
from Sastrawi.StopWordRemover.StopWordRemoverFactory import
StopWordRemoverFactory
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

slangs={'yg':'yang', 'tdk':'tidak', 'pd':'pada', 'mlh':'malah',
'jgn':'jangan', 'jg':'juga', 'tp':'tapi', 'blkg': 'belakang', 'dr':'dari',
'klo':'kalo', 'lg':'lagi', 'btw':'buat',
        'dlm':'dalam','dgn':'dengan', 'poto':'foto', 'g':'tidak', 'n':'dan'}
processed_comments = []

for sentence in data['customer_isi_komen,,,,,,,,,,,,,,,,']:
    # Remove all the special characters
    processed_comment = re.sub(r'\W', ' ', str(sentence))

    # Converting to Lowercase
    processed_comment = processed_comment.lower()

    #Remove number
    processed_comment = re.sub(r'\d+', ' ', processed_comment)

    # remove all single characters
    processed_comment = re.sub(r'\s+[a-zA-Z]\s+', ' ', processed_comment)

    #remove duplicate character
```

```
    pattern=reg.compile(r"(.)\1{1,}",reg.DOTALL)
    processed_comment=pattern.sub(r"\1",processed_comment)


    #Corrected Slang words
    words = processed_comment.split()
    rfrm=[slangs[word] if word in slangs else word for word in words]
    processed_comment= " ".join(rfrm)

    #remove stopword
    factory = StopWordRemoverFactory()
    more_stopword = ['tak', 'jd', 'per', 'nya', 'terjemah', 'diterjemahkan',
'oleh', 'gogle', 'google' ,'nan', 'baik', 'sangat', 'batas', 'coba',
                     'ada','bersih', 'salur', 'baru', 'purwokerto', 'batas',
'hotel', 'coba', 'putus', 'ada'] #menambahkan stopword
    stopwords = factory.get_stop_words() + more_stopword
    temp = [t for t in re.findall(r'\b[a-z]+-?[a-z]+\b',processed_comment) if
t not in stopwords]
    processed_comment = ' '.join(temp)

    #stemming
    stemmer = StemmerFactory().create_stemmer()
    processed_comment = stemmer.stem(processed_comment)

    # Substituting multiple spaces with single space
    processed_comment = re.sub(r'\s+', ' ', processed_comment, flags=re.I)

    processed_comments.append(processed_comment)

#output data Preprocessing
processed_comments
```

Output:

```
['datang pertama kali inap dominic rekomendasi orang teman tangal des hotel cukup nyaman dekat pusat makan jauh mal besar rita m
 'lokasi ada tengah kota kamar standar istimewa sarap kurang tarik parkir mobil luas dapat layan mobil bersih pegawai harga kama
 'ukur kota cukup kamar cukup nyaman fasilitas loby personal cukup tempat parkir luas breakfast standart cukup lumayan sayang ke
 'luar biasa senang inap sini inap tempat strategis banyak sekali kuliner sekitar sarapanya perlu baik seluruh senang',
 'punya fasilitas cukup lengkap bagai jenis tipe kamar sedia beberapa pilih kamar letak jalan komisaris bambang suprapto no lor
 'inap satu malam dar transit lanjut jalan esok hari lokasi strategis mana walkable sedia banyak stret fod murah ukur budget kam
 'kamar besar adem layan staf cukup junior suite rom murah pernah kunjung lama kurang muas sisi breakfast menu sarap pagi biasa
 'letak tepi jalan raya letak strategis dekat stasiun kereta arah timur kamu kamar lumayan bagus nyaman kamar dapat meja kerja d
 'kamar bagus mantap lumayan sarap nga terlalu banyak pilih sih overal oke kalau dar numpang tidur nyenyak dar sarap biasa harga
 'beberapa kali inap sini and fel so satisfied harga jangkau tempat nyaman lokasi oke selalu nginap sini sedikit saran breakfast
 'hotel letak strategis deket sama alun alun jadi mudah sat pergi fasilitas kamar cukup breakfast cukup lengkap enak recomended
 'lokasi strategis tengah kota kamar cukup luas loby besar nyaman harga jangkau menu breakfast istimewa lumayan banding harga',
```

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

Tokenizing

```
from nltk.tokenize import word_tokenize

Docs = processed_comment
Sentences = nltk.sent_tokenize(Docs)
print ("Kalimat Awal: ", Sentences)
hasil_tokenizing = nltk.word_tokenize(Docs)
print("Setelah Tokenizing: ", hasil_tokenizing)
```

Output:

```
Kalimat Awal:  ['harap politis gaya lacur pramagtis jelas warna politik punya pr
Setelah Tokenizing:  ['harap', 'politis', 'gaya', 'lacur', 'pramagtis', 'jelas',
```

Lakukan Ekstraksi Fitur:

```
#Ekstraksi fitur / representasi dokumen menggunakan TF
from sklearn.feature_extraction.text import CountVectorizer
tf_vectorizer = CountVectorizer(max_df=1.0, min_df=1)
tf = tf_vectorizer.fit_transform(hasil_tokenizing)
#hasil representasi
tf_terms = tf_vectorizer.get_feature_names_out()
print(tf_vectorizer.get_feature_names_out())
matrix = tf.toarray()
print(matrix)
```

Output:

```
['abal' 'abdu' 'abng' ... 'zina' 'zonamerahyesus' 'zul']
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

Menggunakan LDA:

```
# panggil class LDA
from sklearn.decomposition import LatentDirichletAllocation as LDA

n_topics = 10 #untuk mendapatkan jumlah topik terbaik perlu trial
lda = LDA(n_components=n_topics, learning_method='batch',
random_state=0).fit(tf)
lda
```

```
vsm_topics = lda.transform(tf)
#tampilkan hasil
print(vsm_topics.shape)
vsm_topics
```

Output:

```
(4553, 10)

array([[0.05, 0.05, 0.05, ..., 0.05, 0.05, 0.05],
       [0.05, 0.55, 0.05, ..., 0.05, 0.05, 0.05],
       [0.05, 0.05, 0.05, ..., 0.05, 0.05, 0.05],
       ...,
       [0.05, 0.05, 0.55, ..., 0.05, 0.05, 0.05],
       [0.05, 0.05, 0.05, ..., 0.05, 0.55, 0.05],
       [0.05, 0.05, 0.55, ..., 0.05, 0.05, 0.05]])
```

Tampilkan Nilai Setiap Fitur

```
#Tampilkan nilai-nilai setiap fitur
print(lda.components_)
```

Output:

```
[[1.09999998 0.1        0.1        ... 0.1        0.1        0.1       ]
 [0.1        0.1        1.09999998 ... 3.09999999 0.1        0.1       ]
 [0.1        0.1        0.1        ... 0.1        0.1        0.1       ]
 ...
 [0.1        5.09999999 0.1        ... 0.1        0.1        3.09999999]
 [0.1        0.1        0.1        ... 0.1        0.1        0.1       ]
 [0.1        0.1        0.1        ... 0.1        0.1        0.1       ]]
```

```
 #hasil label topic model untuk setiap dokumen
import numpy as np
topics = np.argmax(vsm_topics, axis=1)
topics
```

Cetak nilai hasil topik modeling:

```
#mencetak word fitur dengan nilai tertinggi pada setiap topik
n_top_words = 10 # jumlah fitur tertinggi yang kita tentukan
topic_words = {}
for topic, comp in enumerate(lda.components_):
 word_idx = np.argsort(comp)[::-1][:n_top_words] # susun indeks secara
terbalik, sehingga yang paling tinggi muncul di awal.
 # store the words most relevant to the topic
 topic_words[topic] = [tf_vectorizer.get_feature_names_out()[i]+'
'+str(comp[i]) for i in word_idx]
```

Rapihkan Hasilnya:

```
for topic, words in topic_words.items():
```

```
print('Topic: %d' % topic)
print(' %s' % ', '.join(words))
```

Output:

```
Topic: 0
 lacur 232.09999998665333, tutup 11.09999998617831,
Topic: 1
 ahok 14.099999987493604, cina 12.099999987433055,
Topic: 2
 ras 57.09999998695473, twiter 55.09999998695149, v
Topic: 3
 jadi 23.099999986880743, mau 21.0999999868588, kat
```

MENGGUNAKAN LSA

```
from sklearn.feature_extraction.text import TfidfVectorizer
vektor = TfidfVectorizer(max_features=400)
```

Buat dataframe baru

```
df = pd.DataFrame(processed_comments,
                  columns=['clean_text'])
df
```

Output:

	clean_text
0	harap politis gaya lacur pramagtis jelas warna...
1	syahganda gugat syahganda mg benar mg lacur in...
2	pdemokrat org punya akal sehat maju bangsa beb...
3	cypridophobia fobia takut lacur tular sakit ke...
4	ramai kerja waktu malam biasa lacur siapa idea...
...	...
395	leta palak sepatu mau kotor kq repot banget la...
396	media lacur lahir rahim jujur twiter status
397	lacur ko dijawa nak
398	lacur
399	rlt filter samsung negara serba ganja lacur le...

400 rows × 1 columns

Tampilkan fiturnya:

```
#menghitung tf-idf dengan TfidfTransformer
vektor_dt=vektor.fit_transform(df['clean_text'].values.astype('U'))
print (vektor_dt)
print (vektor_dt.shape)
```

Output:

```
 (0, 318)      0.23951034279177555
 (0, 354)      0.27489780842884926
 (0, 42)       0.28629002782428437
 (0, 74)       0.27489780842884926
 (0, 256)      0.28629002782428437
 (0, 316)      0.20412287715470182
 (0, 305)      0.33248902319130913
 (0, 397)      0.27489780842884926
 (0, 158)      0.2509025621872107
 (0, 209)      0.06884614540450945
 (0, 306)      0.5154395665260972
 (0, 123)      0.2577197832630486
 (1, 38)       0.29012595329817603
 (1, 86)       0.24842815157491535
 (1, 85)       0.29012595329817603
```

LSA

```
# topic modeling using LSA
from sklearn.decomposition import TruncatedSVD
lsa_model = TruncatedSVD(n_components=10, algorithm='randomized', n_iter=10,
random_state=42)
lsa_top=lsa_model.fit_transform(vektor_dt)
print(lsa_top)
```

Output:

```
[[ 0.24933111 -0.07865109  0.27356868 ...  0.04831845  0.06059273
   0.25087654]
 [ 0.09067863 -0.00828392  0.05273156 ...  0.02819498  0.05368213
   0.0106251 ]
 [ 0.32842781  0.07386586  0.08055762 ... -0.04981908  0.01311987
   0.29610409]
 ...
 [ 0.16773243 -0.01246947 -0.07982067 ...  0.02739688 -0.05927835
  -0.07618904]
 [ 0.80163145 -0.09502307 -0.25809688 ...  0.01755924 -0.09604623
   0.03274495]
 [ 0.1982976   0.06913418 -0.09880784 ... -0.08252284  0.28339126
  -0.1142912 ]]
```

Tampilkan nilai LSA setiap topik:

```
# Memunculkan nilai lsa setiap topik
l=lsa_top[0]
print("Topik- Topik:")
for i,topic in enumerate(l):
 print("Topic ",i," : ",topic*100)
```

Output:

```
Topik- Topik:
Topic  0  :  24.933110625656457
Topic  1  :  -7.865109327338452
Topic  2  :  27.356867809772023
Topic  3  :  9.787982590809786
Topic  4  :  0.8411114254137254
Topic  5  :  7.092801335249677
Topic  6  :  8.518368969199575
Topic  7  :  4.831844681220452
Topic  8  :  6.059273095701476
Topic  9  :  25.08765436868258
```

```
# Memunculkan jumlah kata-kata dalam setiap topik
print(lsa_model.components_.shape)
print(lsa_model.components_)
```

```
(10, 400)
[[ 0.0169678   0.04754346  0.01526395 ...  0.01024565  0.01479111
   0.00539855]
 [ 0.00133987  0.02545798  0.05166031 ...  0.00315093  0.04479447
   0.00151055]
 [-0.01568738 -0.01248813 -0.00830845 ...  0.01645678  0.01487077
   0.00125546]
 ...
 [ 0.01080666  0.06102241 -0.02989535 ...  0.01509123 -0.01221045
   0.00815337]
 [ 0.1060698   0.2640066   0.07185527 ...  0.0076618   0.00719594
   0.00240419]
 [-0.08711626 -0.02858979  0.01924043 ...  0.04512131 -0.01348672
  -0.00847053]]
```

Tampilkan rapi kata yang muncul:

```
# Word/ kata paling penting dalam setiap topik
vocab = vektor.get_feature_names_out()
for i, comp in enumerate(lsa_model.components_):
    vocab_comp = zip(vocab, comp)
    sorted_words = sorted(vocab_comp, key= lambda x:x[1], reverse=True)[:10]
    print("Topic "+str(i)+": ")
    for t in sorted_words:
        print(t[0],end=", ")
    print("\n")
```

```
Topic 0:
lacur, politik, orang, twiter, status, aku, jadi, ras, agama, tuhan,

Topic 1:
ras, orang, twiter, status, mesum, anjing, cina, aku, mau, ahok,

Topic 2:
politik, ras, negri, org, jelas, cina, rakyat, satu, politis, hitam,

Topic 3:
mesum, twiter, status, politik, otak, punya, pic, lihat, emang, kan,

Topic 4:
twiter, status, intlektual, leceh, pic, verbal, bukan, tutup, tuh, perempuan,

Topic 5:
aku, izin, jadi, tuhan, politik, binatang, paksa, hidup, baca, muslimah,
```