MODUL 13: TOPIC MODELING

13.1 Deskripsi Singkat

Topic modeling atau pemodelan topik menyediakan metode untuk mengatur, memahami, mencari, dan meringkas koleksi dokumen berukuran besar secara otomatis dengan tujuan untuk menemukan tema tersembunyi yang meliputi koleksi. Adapun beberapa teknik pemodelan topic yaitu:

1. Latent Semantic Analysis (LSA)
2. Latent Dirichlet Allocation (LDA)

13.2 Tujuan Praktikum

Setelah praktikum pada modul 13 ini diharapkan mahasiswa mempunyai kompetensi sebagai berikut:

1) Dapat mempraktekkan topic modeling dengan LSA.
2) Dapat mempraktekkan topic modeling dengan LDA.

13.3 Material Praktikum

Tidak ada

13.4 Kegiatan Praktikum

Gunakan data teks yang Anda gunakan pada modul 10, yaitu dari dataset analisis sentimen pada tweet (Koto and Rahmaningtyas, 2017) yang diambil dari:
https://github.com/indolem/indolem/tree/main/sentiment
Pada praktikum ini Anda akan melakukan topic modeling pada keseluruhan teks tweet bersentimen negatif. Oleh karena itu, gabungkan terlebih dahulu tweet bersentimen negatif di data train dan testing.

```
text_negatif =
pd.concat([df_train[df_train['sentiment']==0]['clean_text'],
df_test[df_test['sentiment']==0]['clean_text']])

print(text_negatif)
```

Anda dapat memvisualisasikan teks tersebut menggunakan word cloud. Instal terlebih dahulu library wordcloud sebagai berikut.

```
pip install wordcloud
```

Kemudian visualisasikan teks bersentimen negatif dengan kode berikut.

```
import matplotlib.pyplot as plt
%matplotlib inline
from wordcloud import WordCloud
text = " ".join(text_negatif.tolist())
word_cloud = WordCloud(collocations = False, background_color =
```

Output yang ditampilkan untuk tweet bersentimen negatif ditampilkan pada gambar berikut.

Selanjutnya, Anda juga dapat mencoba untuk memvisualisasikan tweet bersentimen positif menggunakan word cloud.

A. Topic Modeling dengan LSA

Lakukan ekstraksi fitur dengan menggunakan TF.IDF.

```
tfidf_vectorizer = TfidfVectorizer(use_idf=True)
text_negatif_vectors = tfidf_vectorizer.fit_transform(text_negatif)
print(text_negatif_vectors.shape)
```

Kemudian, lakukan pemodelan topik dengan menggunakan LSA.

```
from sklearn.decomposition import TruncatedSVD

# SVD represent documents and terms in vectors
svd_model = TruncatedSVD(n_components=5, algorithm='randomized',
n_iter=100, random_state=122)

svd_model.fit(text_negatif_vectors)

len(svd_model.components_)
```

Sepuluh kata yang paling tinggi nilainya pada setiap topik dapat ditampilkan sebagai berikut.

```
terms = tfidf_vectorizer.get_feature_names()

for i, comp in enumerate(svd_model.components_):
    terms_comp = zip(terms, comp)
    sorted_terms = sorted(terms_comp, key= lambda x:x[1],
reverse=True)[:10]
    print("Topic "+str(i)+": ")
    for t in sorted_terms:
        print(t[0])
    print("")
```

B. Topic Modeling dengan LDA

Untuk model LDA, Anda akan menggunakan library gensim. Sebelumnya lakukan tokenisasi terlebih dahulu.

```
def tokenisasi(text):
      tokens = text.split(" ")
      return tokens

tokens_negatif = [tokenisasi(t) for t in text_negatif]
print(tokens_negatif)
```

Buat dictionary dari data teks, kemudian konversi menjadi bag of word menggunakan term frequency.

```
import gensim.corpora as corpora
# Create Dictionary
id2word = corpora.Dictionary(tokens_negatif)
# Create Corpus
# Term Frequency
corpus = [id2word.doc2bow(text) for text in tokens_negatif]
# View
print(id2word)
print(corpus[:1])
```

```
Dictionary(8136 unique tokens: ['banyak', 'bisa', 'di', 'dipakai', 'noda']...)
[[(0, 2), (1, 1), (2, 1), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1), (8, 1)]]
```

Atau Anda juga dapat menampilkan isi dari corpus tersebut dengan kode berikut

```
# Human readable format of corpus (term-frequency)
[[(id2word[id], freq) for id, freq in cp] for cp in corpus[:1]]
```

```
[[('banyak', 2),
  ('bisa', 1),
  ('di', 1),
  ('dipakai', 1),
  ('noda', 1),
  ('nyamuk', 1),
  ('sprei', 1),
  ('tidak', 1),
  ('wifi', 1)]]
```

Selanjutnya, lakukan pemodelan topik dengan LDA.

```
# Build LDA model
import gensim
lda_model = gensim.models.LdaMulticore(corpus=corpus,
                                       id2word=id2word,
                                       num_topics=10,
                                       random_state=100,
```

```
                                        chunksize=100,
                                        passes=10,
                                        per_word_topics=True)
```

Dokumentasi mengenai parameter dari fungsi LdaMulticore pada gensim dapat dilihat pada link berikut: https://radimrehurek.com/gensim/models/ldamulticore.html

Topik yang dihasilkan dari model tersebut dapat ditampilkan dengan kode berikut.

```
lda_model.print_topics() #num_word default is 10
```

Hasil dari pemodelan topik untuk tweet bersentimen negative yaitu sebagai berikut.

```
[(0,
  '0.021*"di" + 0.015*"yang" + 0.013*"kalau" + 0.010*"ada" + 0.008*"sama" + 0.008*"harus" +
0.008*"padahal" + 0.008*"hari" + 0.007*"mau" + 0.007*"jangan"'),
 (1,
  '0.057*"kamar" + 0.053*"tidak" + 0.036*"dan" + 0.033*"kurang" + 0.023*"air" + 0.023*"ada" +
0.022*"mandi" + 0.018*"ac" + 0.016*"kotor" + 0.015*"bau"'),
 (2,
  '0.057*"http" + 0.027*"atUser" + 0.018*"ly" + 0.017*"bit" + 0.016*"pdip" + 0.014*"jokowi" +
0.014*"syahrini" + 0.013*"tak" + 0.012*"di" + 0.010*""'),
 (3,
  '0.020*"pergi" + 0.019*"aku" + 0.016*"sekolah" + 0.015*"com" + 0.014*"p" + 0.012*"https" +
0.011*"path" + 0.009*"kamu" + 0.009*"afgan" + 0.009*"yang"'),
 (4,
  '0.025*"ga" + 0.022*"ada" + 0.019*"di" + 0.019*"yg" + 0.018*"saya" + 0.015*"bisa" +
0.014*"nya" + 0.014*"tv" + 0.013*"tdk" + 0.011*"airy"')]
```

Jika ingin mengetahui proporsi topik dari suatu dokumen berdasarkan model yang sudah dibuat dengan LDA, dapat menggunakan kode berikut.

```
print("Topic distribution for document:", " ".join(tokens_negatif[0]))
print("feature of each token:", [(id2word[id], freq) for id, freq in
corpus[0]])
lda_model.get_document_topics(corpus[0])
```

Outputnya yaitu sebagai berikut.

```
Topic distribution for document: wifi tidak bisa dipakai banyak noda di
sprei banyak nyamuk
feature of each token: [('banyak', 2), ('bisa', 1), ('di', 1),
('dipakai', 1), ('noda', 1), ('nyamuk', 1), ('sprei', 1), ('tidak', 1),
('wifi', 1)]
[(0, 0.12771416),
 (1, 0.81735796),
 (2, 0.018249353),
 (3, 0.01823659),
 (4, 0.018441964)]
```

Untuk mendapatkan output dalam csv berisi topik dominan untuk keseluruhan dokumen, Anda bisa menggunakan kode berikut.

```
def format_topics_sentences(ldamodel, corpus, texts):
    # Init output
    sent_topics_df = pd.DataFrame()

    # Get main topic in each document
    for i, row in enumerate(ldamodel[corpus]):
        row = sorted(row, key=lambda x: (x[1]), reverse=True)
        # Get the Dominant topic, Perc Contribution and Keywords for
each document
        for j, (topic_num, prop_topic) in enumerate(row):
            if j == 0:  # => dominant topic
                wp = ldamodel.show_topic(topic_num, 5) #get most
significant topic
                topic_keywords = ", ".join([word for word, prop in wp])
                sent_topics_df =
sent_topics_df.append(pd.Series([int(topic_num), round(prop_topic,4),
topic_keywords]), ignore_index=True)
            else:
                break
    sent_topics_df.columns = ['Dominant_Topic', 'Perc_Contribution',
'Topic_Keywords']

    # Add original text to the end of the output
    contents = pd.Series(texts)
    sent_topics_df = pd.concat([sent_topics_df, contents], axis=1)
    return(sent_topics_df)


df_topic_sents_keywords = format_topics_sentences(ldamodel=lda_model,
corpus=corpus, texts=tokens_negatif)

# Format
df_dominant_topic = df_topic_sents_keywords.reset_index()
df_dominant_topic.columns = ['Document_No', 'Dominant_Topic',
'Topic_Perc_Contrib', 'Topic Keywords', 'Text']

#Save to csv
df_dominant_topic.to_csv('/content/drive/MyDrive/kuliah/Information
Retrieval 22-23/Bahan Modul/bahan_latihan13/doc_topic_dominant.csv')

# Show
df_dominant_topic.head(5)
```

Output yang dihasilkan sebagai berikut.

	Document_No	Dominant_Topic	Topic_Perc_Contrib	Topic Keywords	Text
0	0	1.0	0.8174	kamar, tidak, dan, kurang, ada	[wifi, tidak, bisa, dipakai, banyak, noda, di,...
1	1	3.0	0.9267	pergi, aku, sekolah, com, p	[kangen, karimata, pengen, makan, kepiting, re...
2	2	1.0	0.6131	kamar, tidak, dan, kurang, ada	[kamar, oke, fasilitas, oke, kasur, keras, mun...
3	3	3.0	0.5019	pergi, aku, sekolah, com, p	[tidak, ada, pelayanan, saat, datang, dan, ban...
4	4	3.0	0.9598	pergi, aku, sekolah, com, p	[inikah, namanya, cinta, sendirian, yang, kura...
5	5	1.0	0.6103	kamar, tidak, dan, kurang, ada	[check, in, yang, lama, ruangan, kamar, kurang...

Model yang baik akan menghasilkan topik dengan skor koherensi topik yang tinggi. Skor koherensi digunakan untuk mengukur derajat kemiripan semantik dari kata-kata dengan skor tertinggi yang ada dalam topik tersebut. Skor koherensi dari model tersebut didapatkan dengan kode berikut.

```
from gensim.models import CoherenceModel
# Compute Coherence Score
coherence_model_lda = CoherenceModel(model=lda_model,
texts=tokens_negatif, dictionary=id2word, coherence='c_v')
coherence_lda = coherence_model_lda.get_coherence()
print('\nCoherence Score: ', coherence lda)
```

Untuk melihat skor koherensi per topik, dapat menggunakan kode berikut.

```
coherence_per_topic = coherence_model_lda.get_coherence_per_topic()
print('\nCoherence Per Topic Score: ', coherence_per_topic)
```

Penentuan banyaknya model topik dilakukan dengan cara melihat visualisasi pada grafik coherence score. Buat terlebih dahulu fungsi untuk menghitung coherence score untuk jumlah topik tertentu.

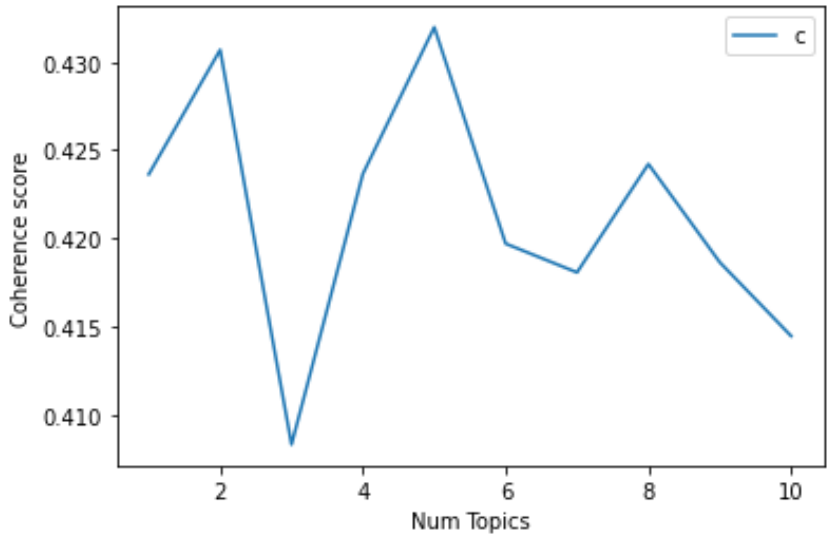
```
#function to compute coherence values
def compute_coherence_values(dictionary, corpus, texts, limit, start,
step):
    coherence_values = []
    model_list = []
    for num_topics in range(start, limit, step):
        model = gensim.models.LdaMulticore(corpus=corpus,
                                           id2word=id2word,
                                           num_topics=num_topics,
                                           random_state=100,
                                           chunksize=100,
                                           passes=10,
                                           per_word_topics=True)
        model_list.append(model)
        coherencemodel = CoherenceModel(model=model, texts=texts,
dictionary=dictionary, coherence='c_v')
        coherence_values.append(coherencemodel.get_coherence())

    return model_list, coherence_values
```

Kemudian panggil fungsi di atas dengan jumlah topik 1 hingga 10 dan visualisasikan skornya.

```
start=1
limit=11
step=1
model_list, coherence_values = compute_coherence_values(id2word,
corpus,  tokens_negatif, start=start, limit=limit, step=step)
#show graphs
import matplotlib.pyplot as plt
x = range(start, limit, step)
plt.plot(x, coherence_values)
plt.xlabel("Num Topics")
plt.ylabel("Coherence score")
plt.legend(("coherence_values"), loc='best')
plt.show()
```

Grafik yang ditampilkan yaitu sebagai berikut.



Kemudian gunakan jumlah topik k dengan coherence score tertinggi. Misalnya, berdasarkan grafik di atas, Anda dapat menggunakan k=5. Kemudian, hasil topic modelling dapat divisualisasikan menggunakan library pyLDAvis. Install terlebih dahulu library tersebut.

```
pip install pyLDAvis
```

Kemudian tuliskan kode berikut.

```
import pyLDAvis.gensim_models
import pickle
import pyLDAvis
# Visualize the topics
pyLDAvis.enable_notebook()
LDAvis_prepared = pyLDAvis.gensim_models.prepare(lda_model, corpus,
id2word)
LDAvis_prepared
```

Perhatikan visualisasi yang ditampilkan, kemudian interpretasikan hasil visualisasi tersebut.

13.5 Penugasan

1. Lakukan duplikasi dari latihan di modul ini dengan Google Colab masing-masing.