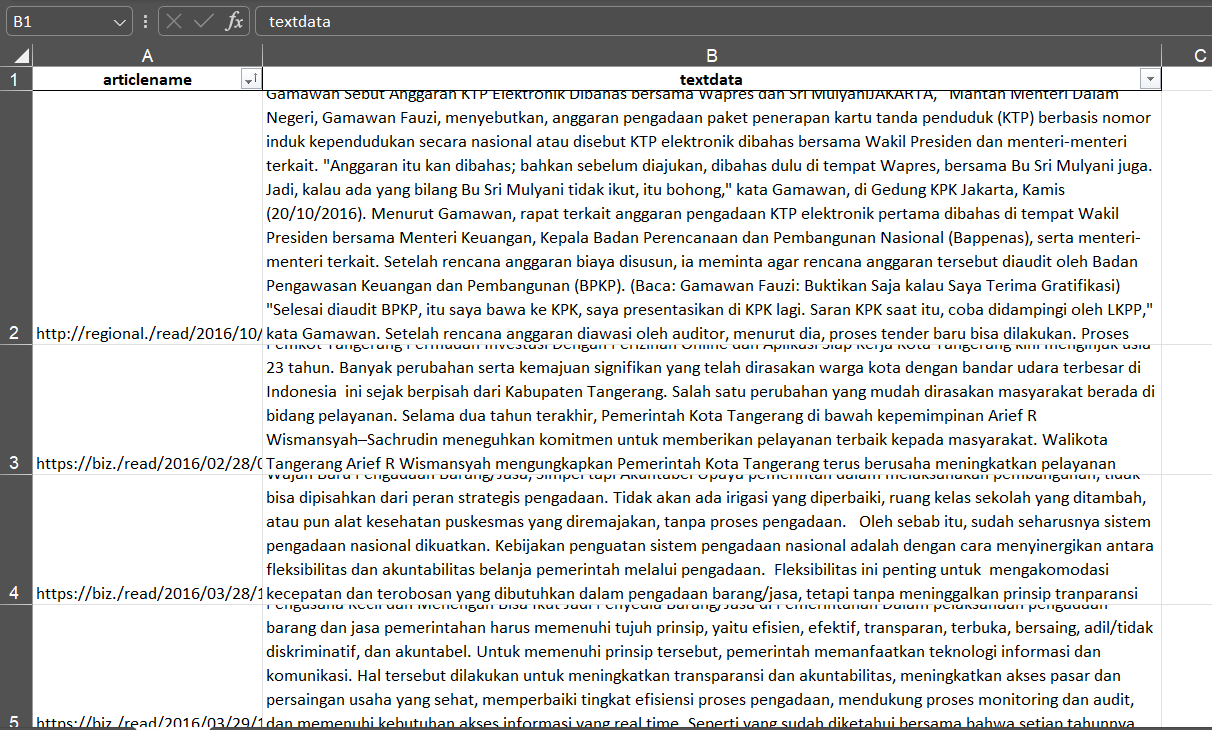
MODELING ARTICLE NEWS DENGAN BITERM TOPIC

Berikut hasil pengolahan data kumpulan dari beberapa artikel dengan menggunakan py



Berikut scrypt pengolahan data dengan menggunakan

import bitermplus as btm

import numpy as np

import pandas as pd

import re

import string

from nltk.tokenize import word\_tokenize

import tomotopy as tp

import pyLDAvis

import pyLDAvis.gensim\_models as gensimvis

from gensim import corpora

from gensim.models import CoherenceModel

# IMPORTING DATA

df = pd.read\_excel("dataBerita.xlsx")

def remove\_tweet\_special(text):

    text = text.replace('\\t', " ").replace('\\n', " ").replace('\\u', " ").replace('\\', "")

    text = text.encode('ascii', 'replace').decode('ascii')

    return text.replace("http://", " ").replace("https://", " ")

df['textdata'] = df['textdata'].apply(remove\_tweet\_special)

def remove\_number(text):

    return re.sub(r"\d+", "", text)

df['textdata'] = df['textdata'].apply(remove\_number)

def remove\_punctuation(text):

    return text.translate(str.maketrans("", "", string.punctuation))

df['textdata'] = df['textdata'].apply(remove\_punctuation)

def remove\_whitespace\_LT(text):

    return text.strip()

df['textdata'] = df['textdata'].apply(remove\_whitespace\_LT)

def remove\_single\_char(text):

    return re.sub(r"\b[a-zA-Z]\b", "", text)

df['textdata'] = df['textdata'].apply(remove\_single\_char)

def word\_tokenize\_wrapper(text):

    return word\_tokenize(text)

df['textdata\_tokens'] = df['textdata'].apply(word\_tokenize\_wrapper)

texts = df["textdata"].str.strip().to\_list()

# PREPROCESSING

# Obtaining terms frequency in a sparse matrix and corpus vocabulary

X, vocabulary, vocab\_dict = btm.get\_words\_freqs(texts)

tf = np.array(X.sum(axis=0)).ravel()

# Vectorizing documents

docs\_vec = btm.get\_vectorized\_docs(texts, vocabulary)

docs\_lens = list(map(len, docs\_vec))

# Generating biterms

biterms = btm.get\_biterms(docs\_vec)

# INITIALIZING AND RUNNING MODEL

model = btm.BTM(

    X, vocabulary, seed=12321, T=8, M=20, alpha=50/8, beta=0.01)

model.fit(biterms, iterations=20)

p\_zd = model.transform(docs\_vec)

# METRICS

perplexity = btm.perplexity(model.matrix\_topics\_words\_, p\_zd, X, 8)

coherence = btm.coherence(model.matrix\_topics\_words\_, X, M=20)

print("coherence :")

print(coherence)

print(model.labels\_)

print("\n")

print(btm.get\_docs\_top\_topic(texts, model.matrix\_docs\_topics\_))

print("perplexity :")

print(perplexity)

# Using tomotopy for graphical representation

corpus = [text.split() for text in texts]

mdl = tp.LDAModel(k=8, alpha=0.1, eta=0.01)

for doc in corpus:

    mdl.add\_doc(doc)

mdl.train(0)

print('Num of docs:', len(mdl.docs))

print('Vocab size:', len(mdl.used\_vocabs))

print('Num of words:', mdl.num\_words)

for i in range(100):

    mdl.train(10)

    print('Iteration:', i, 'LL:', mdl.ll\_per\_word)

# Prepare data for pyLDAvis

def prepare\_lda\_vis\_data(mdl):

    topic\_term\_dists = np.array([mdl.get\_topic\_word\_dist(k) for k in range(mdl.k)])

    doc\_topic\_dists = np.array([doc.get\_topic\_dist() for doc in mdl.docs])

    doc\_lengths = [len(doc.words) for doc in mdl.docs]

    vocab = list(mdl.used\_vocabs)

    term\_frequency = mdl.get\_count\_by\_topics()

    return pyLDAvis.prepare(topic\_term\_dists, doc\_topic\_dists, doc\_lengths, vocab, term\_frequency)

# Visualizing the results

vis\_data = prepare\_lda\_vis\_data(mdl)

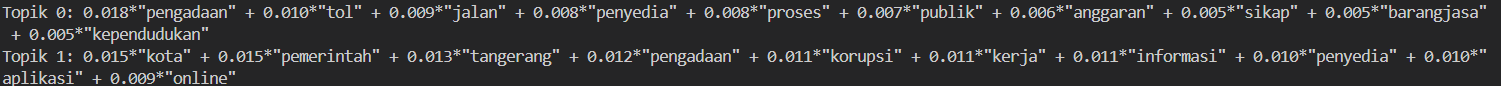
pyLDAvis.show(vis\_data)

# Top words in topics

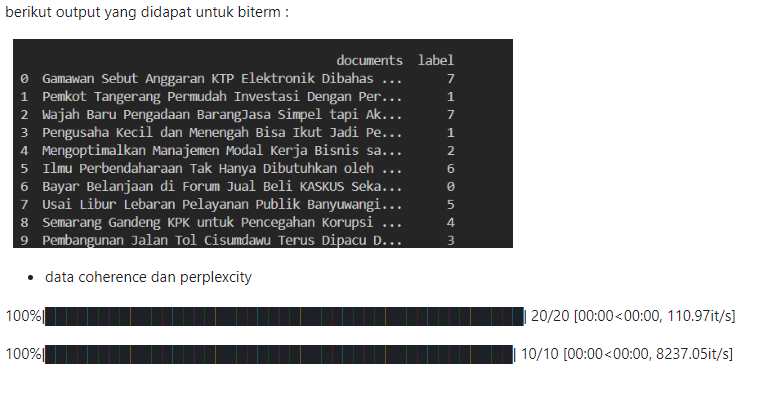
for k in range(mdl.k):

    print('Topic #{}'.format(k), mdl.get\_topic\_words(k, top\_n=10))

output dengan ditentukan 2 topik pada lda sebagai berikut:



Berikut output dan data coherence yang didapat



Nilai coherence dan perplexity

* coherence :

[         inf -41.52685673 -46.3970468           inf -12.78689599

 -38.57783073 -74.73049976 -55.86616029]

* perplexity :

1062.3598610194715

Dari nilai coherence model ini tidak terlalu bagus,kemungkinan karena preprosesing belum maksimal. Karena nilai cohenrence itu Semakin besar coherence score, maka semakin baik pula hasil interpretasi topic modeling yang dihasilkan.

nilai perplexity terlalu besar seperti paragraph menandakan pembentukan topik kurang maksimal,karena semakin kecil perplexity, semakin baik model dengan jumlah topik tersebut. Meskipun perplexity dapat menilai kemampuan prediktif model pelatihan topik sampai batas tertentu, ketika jumlah topik dipilih oleh perplexity, jumlah topik yang dipilih sering kali besar, dan topik yang serupa cenderung muncul, sehingga menghasilkan pengenalan topik yang rendah.