BAB 3
METODE PENELITIAN

Penelitian ini merupakan penelitian *experimental*. Penelitian metode eksperimen merupakan salah satu penelitian yang apabila ada di kondisi-kondisi tertentu yang dikendalikan sehingga satu atau beberapa variabel dapat dikontrol untuk menguji dalam suatu hipotesis. Berikut ini adalah bahan, alat, dan metode jalan penelitiannya untuk memodelkan suatu pola topik yang didapat dari Twitter.

3.1 BAHAN PENELITIAN

Bahan penelitian ini akan menggali data dan informasi dari *tweet* yang ada di dalam website Twitter terkait dengan banjir di wilayah Indonesia.

3.2 ALAT PENELITIAN

Alat yang digunakan dalam penelitian ini adalah komputer dengan spesifikasi cukup untuk menjalankan sistem operasi dan perangkat lunak pengembangan serta koneksitas internet.

Sistem operasi dan program-program aplikasi yang digunakan dalam pengembangan aplikasi ini adalah:

1. Sistem operasi Windows 10 64-bit
2. Microsoft Office Excel 2007
3. Anaconda versi 3 2021.02 64 bi
4. Python versi 3.7.3
5. Jupyter Notebook
6. Library Python

Berikut adalah library python yang digunakan dalam menjalankan proses *text mining* :

a. Pandas adalah *library* Python yang menyediakan struktur data yang cepat, fleksibel, dan ekspresif yang dirancang untuk membuat

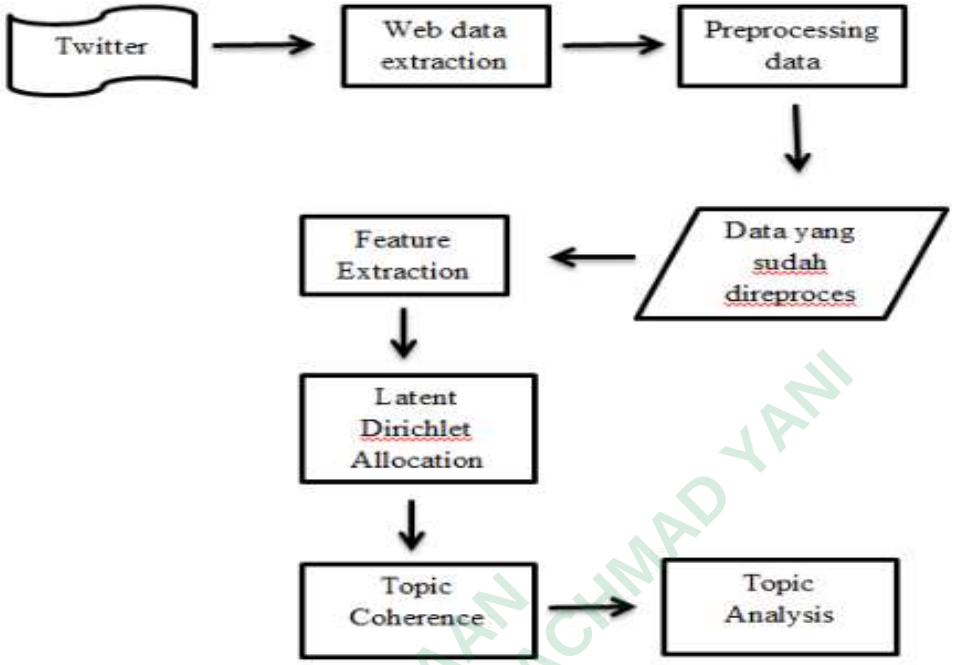
pekerjaan lebih terstruktur dan deretan data dalam bentuk yang mudah dan intuitif.

b. Numpy adalah *library* dasar Python untuk array computing. Numpy menyediakan fitur yang salah satunya yaitu dapat membentuk objek N-dimensional *array*.
c. Matplotlib adalah *library* Python yang komprehensif untuk membuat visualisasi statis, animasi, dan interaktif di Python. Pada penelitian ini menggunakan modul matplotlib.pyplot.
d. Seaborn adalah *library* untuk membuat grafik statistik di Python yang dibangun di atas matplotlib dan terintegrasi dengans truktur data *library* Pandas.
e. Sklearn atau scikit-learn adalah modul Python untuk *machine learning* yang dibangun di atas SciPy.
f. Re atau regular expression adalah modul Python yang digunakan untuk mencari string atau teks dengan menggunakan pola dari rentetan karakter.
g. NLTK atau *Natural Language* Toolkit adalah *library*Python yang digunakan untuk permodelan teks dan analisa teks. Pada penelitian ini menggunakan modul nltk.tokenize dengan sub modul regexp tokenizer dan twee ttokenizer dan modul nltk.corpus dengan sub modul *stopwords*.
h. Emoji adalah *library* Python yang digunakan untuk membaca rangkaiankode emoji yang didefinisikan sebagai konsorsi umum nikode.
i. Gensim adalah *library* Python untuk *topic modelling*, pengindeksan dokumen dan pencarian kesamaan dengan *corpus* yang besar. Pada penelitian ini menggunakan modul corpora dan modul gensim.models dengan submodul Coherence Model, LdaModel, LsiModel, HdpModel.
j. Pickle adalah *library* Python yang digunakan untuk menyimpan data ke dalam file dan membaca data dari file.

k. Os adalah *library* Python yang menyediakan cara portable dalam menggunakan fungsionalitas yang tergantung pada system operasi. Pada penelitian ini menggunakan modul path.
l. PIL adalah *library* Python yang digunakan untuk menam bahkan kemampuan pemrosesan gambar di Python. Pada penelitian ini menggunakan modul Image.
m. Wordcloud adalah *library* Python yang digunakan untuk visualisasi data yang mewakili data teks dimana ukuran setiap kata menunjukkan frekuensi atau pentingnya sebuah kata dalam data tekstersebut. Pada penelitian ini menggunakan modul WordCloud, STOPWORDS, Image Color Generator.
n. Pyldavis adalah *library* Python untuk visualisasi topic model yang interaktif. Pada penelitian ini menggunakan modul pyLDAvis.gensim.

3.3 JALAN PENELITIAN

Pada jalan penelitian ini menggunakan software Anaconda versi 3, python dan Jupyter Notebook untuk melakukan pengambilan data yang akan ditampilkan di Microsoft Office Excel dan diolah data tersebut, kemudian setelah itu dimodelkan data tweet yang berkaitan dengan banjir di wilayah Indonesia dari data tweet yang telah diambil. Adapun tahapan alur penelitian ini yaitu:



Gambar 3.1 Alur Penelitian

3.3.1 Web Data Extraction

Tahapan dimulai dengan melakukan web data extraction untuk mengambil dan mengumpulkan data dari *tweet* di website Twitter terkait dengan banjir di wilayah Indonesia yang kemudian ditampung menjadi kumpulan dokumen. (Putra & Kusumawardani, 2017)

1. Pengumpulan Data

Mengambil dan mengumpulkan data dari *tweet* di website Twitter yang terkait dengan Banjir dengan kata kunci "banjir indonesia" menggunakan *twitterscraper*. Data *tweet* diambil dari rentang tanggal 1 juni 2021 sampai dengan tanggal 2 Juni 2021 dengan jumlah data tweet yang diperoleh yaitu sebanyak 2000.

2. Import Data

Sebelum melakukan import data, memasukkan library untuk melihat proses pengolahan struktur data dapat dilihat pada Gambar 3.2

```
# packages to store and manipulate data
import pandas as pd
import numpy as np

# plotting packages
import matplotlib.pyplot as plt
import seaborn as sns

# model building package
import sklearn

# package to clean text
import re
```

Gambar 3.2 Library proses pengolah data

Berdasarkan gambar diatas dapat diketahui bahwa beberapa library yang dibutuhkan dalam import data yaitu pandas, numpy, matplotlib.pyplot, seaborn, sklearn, dan re untuk proses pengolahan data.

Untuk langkah selanjutnya adalah melakukan mengimport data pada yang dapat dilihat pada Gambar 3.3.

```
def load_data():
    data = pd.read_excel('Banjir indonesia.xlsx')
    return data

df = pd.DataFrame(data)
df
```

Gambar 3.3 Perintah import data

Kode diatas adalah kode yang digunakan untuk membuka file berekstensi *excel*, kemudian memuat data dari file *excel*, dan memberikan struktur pada data yang di-*load*

3.3.2 Preprocessing

Setelah melakukan tahap web data extraction, kemudian data yang sudah ada diproses terlebih dahulu sebelum melakukan text mining. Berikut adalah tahapan yang dilakukan dalam proses preprocessing. Selanjutnya adalah masukkan library yang sudah di install dan dapat dilihat pada gambar 3.4

```
import nltk
from nltk.tokenize import RegexpTokenizer
from nltk.corpus import stopwords
import emoji
```

Gambar 3.4 Library preprocessing

Pada Gambar 3.4 diatas dipakai untuk memasukkan *library* dan modul yaitu nltk, RegexpTokenizer, *stopwords*, dan emoji untuk proses *preprocessing*.

1. Cleaning

Cleaning disini adalah menghilangkan atribut yang tidak penting seperti tag dan url dari data tweet menggunakan perintah yang bisa dilihat pada gambar 3.5

```
def remove_links(caption):
    '''Takes a string and removes web links from it'''
    caption = re.sub(r'http\S+', '', caption) # remove http links
    caption = re.sub(r'bit.ly/\S+', '', caption) # rempve bitly links
    caption = caption.strip('[link]') # remove [links]
    return caption

def remove_users(caption):
    '''Takes a string and removes retweet and @user information'''
    caption = re.sub(r'(RT\s@[A-Za-z]+[A-Za-z0-9-_]+)', '', caption) # remove retweet
    caption = re.sub(r'(@[A-Za-z]+[A-Za-z0-9-_]+)', '', caption) # remove tweeted at
    return caption

def give_emoji_free_text(caption):
    allchars = [text for text in caption]
    emoji_list = [c for c in allchars if c in emoji.UNICODE_EMOJI]
    clean_text = ' '.join([text for text in caption.split() if not any(i in text for i in emoji_list)])
    return clean_text
```

Gambar 3.5 Tahapan Cleaning

Pada baris def remove_links sampai dengan baris return caption fungsinya untuk menghilangkan link dan url, pada baris def remove_users sampai dengan baris return caption fungsinya untuk menghilangkan tweeted dan retweet, dan pada baris def give_emoji_free_text sampai dengan return clean_text fungsinya untuk menghilangkan emoji.

2. *Casefolding* dan *Stopwords*

Casefolding serta *stopwords* disini kegunaannya untuk menghilangkan kata dan atribut yang sering muncul seperti kata dan, yang, di, ini, dari, dengan, dan sebagainya serta mengubah semua huruf menjadi huruf kecil menggunakan

perintah pada gambar 3.6 dan daftar kata yang digugnakan dapat dilihat pada gambar Tabel 3.1.

Tabel 3.1 Daftar kata stopwords

Kata
"…","””","“","‘","’","x","a","d","di","yg","oi","le","ke","es","mu","ya","rb","lu","ga","pd","ka","gk","dg","jg","va","dan","ini","itu","ayo","loh","ada","com","kwh","pic","ada","spt","loh","kwh","mis","ada","apa","akn","kok","dlm","kan","tak","nya","dgn","klu","tau","yaa","aja","lah","was","tdk","kalo","atau","kita","gitu","atau","yang","dari","saya","tidak","lebih","dengan","hmmm","btshanzt","dybala","xysfquwy","xklnraulqv","jnoxtij","rowkxk","nfptgegajz","osqddqkuqn","ogjlwenigg","fhacvubxr","\u2066","\u2069","\u2063","avhsnhvmt","امہ ین""","opxrvgb","fcguuggjf","lajwise","xhbcv","skfjevnaq","apnmfkcb","odtjfwy","euimwubt","jfpuijov","nictizits","qniqcjnble","rowkxk","trxtnokah","usiiagvxi","yxnocztyg","uhkkazep","btshanzt","rlfwtdjyn","nfptgegajz","sdszcjpz","odtjfwytak","pidie","anda","amin","baik","cara","sdh","klo","lg","jgn","bark","aam","kepada","hrs","jga","bisa","lagi","tp","blm","mer","adalah","sudah","sih","kenapa","gak","qvbwxswxl","si","eqfmfcus","trs","aam","aku","iii","cuma","kau","qpixhjtm","agavsbzmx","icncpjlfk","frans","ane","ttg","ifgwdnfg","luxrqu","ezboaqmwg","hs","jkn","yfrlkbe","pgnv","gw","eh","fobxesf","guooooooobloooooook","na","afc","twzbxavbdo","xqrdsmnz","elzxadcfef","utk","fadilah","supari","siti","ygeuecm","pada","untuk","امہ ین","'akan","krdy","juga","kpd","mau","krn","t","ia","ngga","knp","s","nich","—",“banget”,”gue”,”gua”,”nang”,”neng”,”konte”,”konten”,”bgt”,”aaa”,”rindu awakseparuhnyawa”,”aa”,”aaa”

```python
my_stopwords = ["    ","-","_","**","**","**","**","x","a","d","di","yg","oi","le","ke","es","mu","ya","rb","lu","ga","pd",
                "ka","gk","dg","jg","va","dan","ini","itu","ayo","loh","ada","com","kwh","pic","ada","spt","loh","kwh","mis","ada"
                "apa","akn","kok","dlm","kan","tak","nya","dgn","klu","tau","yaa","aja","lah","was","tdk","kalo","atau","kita","gi
                "atau","yang","dari","saya","tidak","lebih","dengan","hmmm","btshanzt","dybala","xysfquwy","xklnraulqv","jnoxtij",
                "nfptgegajz","osqddqkuqn","ogjlwenigg","fhacvubxr","\u2066","\u2069","\u2063","avhsmhvmt","opxrvgb","fcguuggjf",
                "lajwise","xhbcv","skfjevnaq","apnmfkzb","odtjfwy","eulmaubt","jfpuljov","nictlzits","qnlqcjnble","rowkxk","trxtnn
                "usllagvxi","yxnocztyg","uhkkazep","btshanzt","rlfwtdjyn","nfptgegajz","sdszcjpz","odtjfwytak","pidie","anda","ami
                "cara","sdh","klo","lg","jgn","bark","aam","kepada","hrs","jga","bisa","lagi","tp","blm","mer","adalah","sudah","s
                "gak","qvbwxswxl","si","eqfmfcus","trs","aam","aku","iii","cuma","kau","qpixhjtm","agavsbzmx","icncpjlfk",
                "frans","ane","ttg","ifgwdnfg","luxrqu","ezboaqmwg","hs","jkn","yfrlkbe","pgnv","gw","eh","fobxesf","guooooooobloo
                "na","afc","twzbxavbdo","xqrdsmnz","elzxadcfef","utk","fadilah","superi","siti","ygeuecm","pada","untuk",
                "آمين","akan","krdy","juga","kpd","mau","krn","t","ia","ngga","knp","s","nich","kami"]
# word_rooter = nltk.stem.snowball.PorterStemmer(ignore_stopwords=False).stem
my_punctuation = '!"$%&\'()*+,-./:;<=>?[\\]^_`{|}~•@'

# cleaning master function
def clean_caption(caption, bigrams=False):
    caption = remove_users(caption)
    caption = remove_links(caption)
    caption = give_emoji_free_text(caption)
    caption = caption.lower() # lower case
    caption = re.sub(r'['+my_punctuation + ']+', ' ', caption) # strip punctuation
    caption = re.sub(r'\s+', ' ', caption) #remove double spacing
    caption = re.sub(r'([0-9]+)', '', caption) # remove numbers
    caption_token_list = [word for word in caption.split(' ')
                          if word not in my_stopwords] # remove stopwords

#     caption_token_list = [word_rooter(word) if '#' not in word else word
#                           for word in caption_token_list] # apply word rooter
    if bigrams:
        caption_token_list = caption_token_list+[caption_token_list[i]+'_'+caption_token_list[i+1]
                                                 for i in range(len(caption_token_list)-1)]
    caption = ' '.join(caption_token_list)
    return caption
```

Gambar 3.6 Tahapan Casefolding serta Stopwords

Berdasarkan pada gambar diatas baris pada my_stopwordsyaitu digunakan untuk menampung kata yang sering muncul tetapi tidak memiliki makna atau data *stopwords*, pada baris my_punctuation yaitu memasukkan atribut yang tidak digunakan. Pada baris def clean_caption sampai dengan return caption fungsinya adalah untuk melakukan proses *casefolding* dan *stopwords*.

3. Tokenizing

Pada tahapan tokenizing disini berfungsi untuk memisahkan kata-kata dari sebuah kalimat sehingga menjadi sebuah kata tunggal agar kata tersebut dapat berdiri sendiri dengan menggunakan libary dan perintah yang dapat dilihat pada Gambar 3.7, Gambar 3.8, Gambar 3.9, dan Gambar 3.10.

```python
from nltk.tokenize import TweetTokenizer
tknzr = TweetTokenizer()

def tokenize(text):
    lda_tokens = []
    tokens = tknzr.tokenize(text)
    for token in tokens:
        lda_tokens.append(token)
    return lda_tokens
```

Gambar 3.7 Library Tokenizing

Pada gambar 3.7 di atas adalah digunakan untuk memasukkan modul di *library* Tweet Tokenizer dan dari def tokenize sampai dengan return lda_tokens fungsinya dalah untuk memisahkan kata dari sebuah kalimat.

```
def prepare_text_for_lda(text):
    tokens = tokenize(text)
    return tokens
```

Gambar 3.8 Fungsi dari token Tokenizing

Pada gambar kode diatas adalah kode yang digunakan untuk fungsi dan membuat kata agar bisa berdiri sendiri

```
for data in df["clean_caption"]:
    tokens = prepare_text_for_lda(data)
    print(tokens)
```

Gambar 3.9 Perulangan data dari range variabel df

Pada gambar kode diatas adalah kode yang digunakan untuk membuat perulangan dari token.

```
# import random
text_data = []
for line in df["clean_caption"]:
    tokens = prepare_text_for_lda(line)
#     if random.random() > .99:
    print(tokens)
    text_data.append(tokens)
```

Gambar 3.10 Perulangan dan menambahkan data tokens

Pada gambar kode diatas adalah kode yang digunakan untuk membuat perulangan dengan menambahkan token ke urutan belakang.

3.3.3 Feature Extraction menggunakan TF-IDF

Setelah selesai dalam melakukan tahap preprocessing, kemudian kumpulan kata dari data tweet diberi nilai atau bobot untuk mengetahui pentingnya dari kata tersebut dengan menggunakan *library* dan perintah pada Gambar 3.11, dan Gambar 3.12.

```
from gensim import corpora, models
dictionary = corpora.Dictionary(text_data)
corpus = [dictionary.doc2bow(text) for text in text_data]
import pickle
pickle.dump(corpus, open('corpus.pkl', 'wb'))
dictionary.save('dictionary.gensim')
```

Gambar 3.11 Membuat kamus dari kumpulan beberapa tulisan

Pada gambar 3.11 diatas adalah digunakan untuk memasukkan modul library corpora, models, dan pickle. Setelah itu kumpulan kata disimpan kedalam *file* corpus.pkl dan kamus kata disimpan kedalam *file* dictionary.gensim.

```
tfidf = models.TfidfModel(corpus)
corpus = tfidf[corpus]
from pprint import pprint
for doc in corpus:
    pprint(doc)
    break
```

Gambar 3.12 Perhitungan TF-IDF

Pada gambar 3.12 diatas adalah digunakan untuk memasukkan modul library pprint, setelah itu membuat variabel tf-idf untuk melakukan perhitungan dari frekuensi data yang bsering muncul, dan terakhir membuat perulangan untuk menampilkan nilai dari variabel tf-idf.

3.3.4 *Topic Modelling* Menggunakan LDA

Setelah tahapan *feature extraction* selesai, kemudian yang akan dilakukan adalah dengan membuat pemodelan topik menggunakan algoritma LDA dengan menggunakan *library* dan perintah yang terdapat pada gambar 3.13 dan gambar 3.14.

```
import gensim
ldamodel = gensim.models.LdaMulticore(corpus, num_topics=20, id2word=dictionary, passes=2, workers=4)
ldamodel.save('model_tfidf20.gensim')
topics = ldamodel.print_topics(num_words=30)
for topic in topics:
    print(topic)
```

Gambar 3.13 Library gensim dan menjalankan LDA

Pada gambar 3.13 diatas adalah digunakan untuk memasukkan library gensim, setelah itu menyimpan model *LDA* kedalam file model_tfidf20.gensim, dan terakhir menampilkan model *LDA*.

```
x=ldamodel.show_topics(num_topics=20, num_words=100,formatted=False)
topics_words = [(tp[0], [wd[0] for wd in tp[1]]) for tp in x]

#Below Code Prints Topics and Words
for topic,words in topics_words:
    print(str(topic)+ "::"+ str(words))
print()

#Below Code Prints Only Words
topik=[]
for topic,words in topics_words:
     topik.append(" ".join(words))
#     topik.append(words)
#     print(" ".join(words))
```

Gambar 3.14 Memasukkan kata-kata pertopik kedalam list

Pada gambar 3.14 diatas adalah digunakan untuk menampilkan data topik yang didapatkan dari perulangan for topic,words in topic_words sampai dengan print(str(topic)+ "::"+ str(words)).

3.3.5 Visualisasi

Setelah melakukan dari tahapan *topic modelling*, kemudian dari topik-topik tersebut, selanjutnya adalah memvisualisasikan dengan menggunakan library, dan file yang tersimpan, dan perintah yang bisa dilihat pada Gambar 3.15

```
dictionary = gensim.corpora.Dictionary.load('dictionary.gensim')
corpus = pickle.load(open('corpus.pkl', 'rb'))
lda = gensim.models.ldamodel.LdaModel.load('model_tfidf20.gensim')
import pyLDAvis.gensim
lda_display = pyLDAvis.gensim.prepare(lda, corpus, dictionary, sort_topics=True)
pyLDAvis.display(lda_display)
```

Gambar 3.15 Diagram LDA

Pada gambar 3.15 diatas adalah digunakan untuk memasukkan modul *library* pyLDAvis.gensim, memuat file yaitu dictionary.gensim, corpus.pkl, dan model_tfidf20.gensim, dan terakhir menampilkan *topic modelling* dari file yang sudah dimuat.

Selain itu, ada beberapa kumpulan topik-topik yang bisa divisualisasikan kedalam melalui kata dalam topik dimana ukuran dari ukuran setiap kata menunjukkan frekuensi atau akan pentingnya sebuah kata dari topik tersebut.

Sebelum itu masukkan terlebih dahulu *library* dan perintah yang dapat dilihat pada Gambar 3.16, Gambar 3.17, dan Gambar 3.18.

```
from os import path
from PIL import Image
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator

import matplotlib.pyplot as plt
```

Gambar 3.16 Library untuk modul wordcloud

Pada Gambar 3.16 diatas adalah digunakan untuk untuk memasukkan modul *library* path, Image, wordCloud, STOPWORDS, ImageColorGenerator, dan matplotlib.pyplot.

```
#convert list to string and generate
my_list=tops
# unique_string=(" ").join(my_list)
# wordcloud = WordCloud(width = 1000, height = 500).generate(unique_string)
wordcloud = WordCloud(width = 1000, height = 500, background_color="white").generate(my_list)
plt.figure(figsize=(15,8))
plt.imshow(wordcloud)
plt.axis("off")
plt.savefig("your_file_name"+".png", bbox_inches='tight')
plt.show()
plt.close()
```

Gambar 3.17 Wordcloud keseluruhan topik

Pada Gambar 3.17 diatas adalah digunakan untuk menampilkan wordcloud dari keseluruhan topik, kemudian menyimpannya dengan nama your_file_name.png.

```
for data in topik:
#     print(data)
    my_list=data
    wordcloud = WordCloud(width = 1000, height = 500, background_color="white").generate(my_list)
    plt.figure(figsize=(10,10))
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.savefig("your_file_name"+".png", bbox_inches='tight')
    plt.show()
    plt.close()
```

Gambar 3.18 Wordcloud pertopik

Pada gambar 3.18 diatas adalah digunakan untuk menampilkan *wordcloud* dari masing-masing topik, setelah itu menyimpannya dengan nama your_file_name.png.

3.3.6 Topic Analysis

Pada tahapan dari topic analisis disini adalah untuk menguji kualitas topik yang dihasilkan dari *topic modelling* dengan menggunakan hasil dari *topic coherence* yang ditampilkan dalam bentuk grafik diagram. Berikut ini adalah perintah dan *library* yang digunakan yang dapat dilihat pada Gambar 3.19, Gambar 3.20, dan Gambar 3.21.

```
ldatopics = ldamodel.show_topics(formatted=False)
```

Gambar 3.19 Memanggil topik LDA yang sudah ada

Pada gambar diatas adalah bertujuan untuk membuat variabel ldatopics untuk memanggil topik dari proses *topic modelling*.

```
from gensim.models import CoherenceModel, LdaModel, LsiModel, HdpModel
def evaluate_graph(dictionary, corpus, texts, limit):
    """
    Function to display num_topics - LDA graph using c_v coherence

    Parameters:
    ----------
    dictionary : Gensim dictionary
    corpus : Gensim corpus
    limit : topic limit

    Returns:
    -------
    lm_list : List of LDA topic models
    c_v : Coherence values corresponding to the LDA model with respective number of topics
    """
    c_v = []
    lm_list = []
    for num_topics in range(1, limit):
        lm = LdaModel(corpus=corpus, num_topics=num_topics, id2word=dictionary)
        lm_list.append(lm)
        cm = CoherenceModel(model=lm, texts=texts, dictionary=dictionary, coherence='c_v')
        c_v.append(cm.get_coherence())

    # Show graph
    x = range(1, limit)
    plt.plot(x, c_v)
    plt.xlabel("num_topics")
    plt.ylabel("Coherence score")
    plt.legend(("c_v"), loc='best')
    plt.show()

    return lm_list, c_v
```

Gambar 3.20 Memasukkan library gensim dan fungsi dari evaluate_graph

Pada Gambar 3.20 diatas adalah digunakan untuk memasukkan modul *library* CoherenceModel, LsiModel, LdaModel, HdpModel dan setelah itu

membuat fungsi dalam pemrosesan Topic Coherence dari baris def evaluate_graph sampai dengan baris return lm_list, c_v.

```
%%time
lmlist, c_v = evaluate_graph(dictionary=dictionary, corpus=corpus, texts=text_data, limit=20)
```

Gambar 3.21 Menampilkan diagram Topic Coherence