Tutorial yang dipakai pada percobaan ini berasal dari laman [Text Mining in Python: Steps and Examples] yang terdiri dari lima tahap utama yang masing-masing akan diuraikan sebagai berikut:

1. Data Collection

Contoh tampilan halaman pada news.detik.com

Dikumpulkan data berupa artikel berita dari situs berita news.detik.com dan dibatasi pengambilannya berupa artikel berita yang dipublikasikan pada tanggal 27-03-2024. Untuk detailnya adalah sebagai berikut:

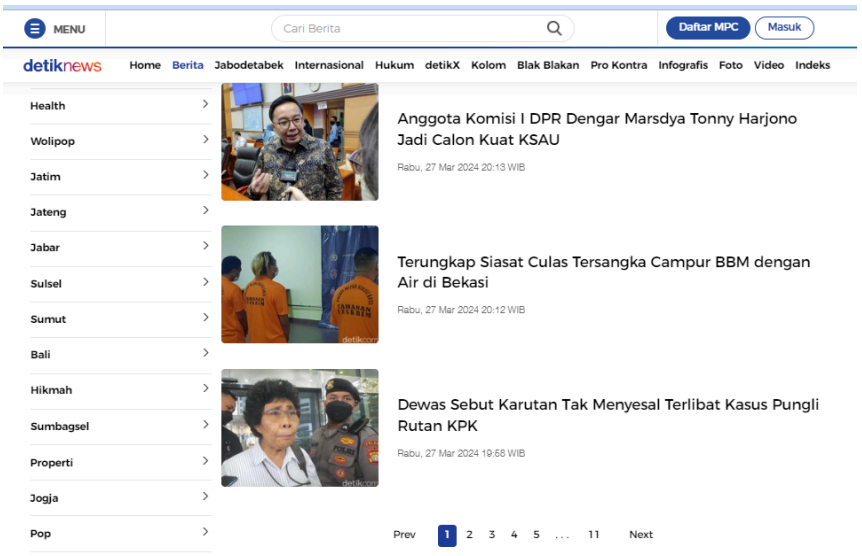
1.1 Digunakan *library* request dan beautifulsoup4 yang berguna untuk melakukan proses scraping data; serta urllib3 yang berguna dalam menangani URL. Ketiga library di-instalasi terlebih dahulu baru di import ke halaman coding sebagai berikut:

```
29s  !pip install nltk requests matplotlib wordcloud beautifulsoup4 urllib3
```

```
1s  [2]  from bs4 import BeautifulSoup
         import urllib.request, sys, time
         import requests
         import pandas as pd
```

Coding untuk instalasi dan import library

1.2 Pada situs berita news.detik.com, didapatkan tampilan *preview* untuk setiap berita dengan hasil pencarian tersebar kedalam 11 halaman dan maksimal banyak artikel yang muncul adalah 20 berita.



Banyak halaman preview untuk artikel berlabel “news” dan dipublikasi pada tanggal 27-03-2024

Karena pada halaman preview hanya muncul sebagian atribut artikel, yaitu judul, tanggal dan hari publikasi, serta foto sampul artikel, maka dikumpulkan terlebih dahulu tautan dari setiap berita yang muncul pada halaman-halaman tersebut dengan kode sebagai berikut:

```
[3]  def getHtmlContent( page_url ):
       # might throw an exception if something goes wrong
       try:
         page_response = requests.get( page_url )

       # what to do if an exception is thrown
       except Exception as e:

         # get exception information
         error_type, error_obj, error_info = sys.exc_info()

         # print the link that cause the problem
         print( "Error for link: ", page_url )

         # print error info and line that threw the exception
         print( error_type, "Line:", error_info.tb_lineno )
       # end try

       # print response object
       return page_response
     # end function
```

Sebuah kode fungsi untuk melakukan koneksi dengan situs terpilih (page_url)

```
[4]  # main page
     main_page_url = 'https://news.detik.com/berita/indeks/1?date=03/27/2024'
     main_page_response = getHtmlContent( main_page_url )
     # main_page_url.status_code

     # parse html using beautifulsoup
     main_soup = BeautifulSoup( main_page_response.text, "html.parser" )

     # get all pages links
     navigation_div = main_soup.find_all( 'a', attrs={ "class": "pagination__item" } )

     # Extract the last page number
     last_page_link = int( navigation_div[-2].text )
     last_page_link
```

Kode untuk mengambil nomor halaman terakhir untuk menentukan url yang akan digunakan untuk mengambil semua artikel berita yang muncul pada masing-masing halaman pencarian

```
# the base url link
base_url = 'https://news.detik.com/berita/indeks/'

# vessel for all collected page text
raw_scrapped_links = []
# Iterate through page numbers from 1 to last_page_link
for page_number in range( 1, last_page_link+1 ):
    # Construct the URL for each page
    current_page_url = f"{base_url}{page_number}?date=03/27/2024"

    # collect data for every page
    current_page_response = getHtmlContent( current_page_url )

    # only collect if current page response is 200
    if current_page_response.status_code == 200:
      # check for response page number
      print( "Yes, Scarpe page", page_number )

      # extracting content from collected html using BeautifulSoup
      soup = BeautifulSoup( current_page_response.text, "html.parser" )

      # filter data based on their tag / class / id / element
      raw_data = soup.find_all( 'article', attrs={ "class":"list-content__item" } )

      # check number of filtered data
      print( "number of news collected: ", len( raw_data ) )

      # colect news link only
      for one_news in raw_data:
        # extract news link
        news_link = one_news.find( "a", attrs={ "class": "media__link" } ).get( "href" )

        # add to final array
        raw_scrapped_links.append( news_link )
      # end loop

      # delay request time
      time.sleep(2)
    # end if
# end loop

# check number of scrapped data
print( len( raw_scrapped_links ) )
```

Kode untuk mengambil semua tautan dari tiap halaman pencarian berita

```python
# to help extract text of news html
def extract_news_text( news_body ):
  # final collected news result
  final_result = []

  # get all text from tag p
  text_raw = news_body.find_all( "p", class_=lambda x: x != "para_caption" and x != "embed video20detik" )

  # keywords exception
  keywords_except = ["simak video", "saksikan live", "halaman berikutnya"]

  # check all collected text lines
  for news_line in text_raw:
    # try to detect any video links
    video_links = news_line.find_all( "a", attrs={"id": "idvideo20detik"} )
    page_number = news_line.find_all( 'div', class_='detail__multiple' )

    # if current line writen as a list
    if ( not video_links and not page_number and news_line.find( 'br' ) ):
      news_text_raw = news_line.get_text( separator="\n" )
      news_text_arr = news_text_raw.split( "\n" )

      # loop all text from list
      for one_txt in news_text_arr:
        # Check if the string does not contain certain keywords
        if len( one_txt ) > 0 and one_txt!=" " and not any( keyword.lower() in one_txt.lower() for keyword in keywords_except ):
          # insert current news line into final array
          final_result.append( one_txt.strip() )
        # end if
      # end loop
    elif ( not video_links and not page_number ):
      # get current text
      news_text = news_line.text

      # Check if the string does not contain certain keywords
      if len( news_text ) > 0 and news_text!=" " and not any( keyword.lower() in news_text.lower() for keyword in keywords_except ):
        # insert current news line into final array
        final_result.append( news_text.strip() )
      # end if
    # end if
  # end loop

  # return all collected news paragraph
  return final_result
# end function
```

Sebuah kode fungsi untuk mengambil semua konten yang ada pada tiap tautan artikel berita terkecuali konten berjenis video

1.3 Kemudian dari tiap tautan yang berisi masing-masing artikel berita, maka selanjutnya adalah mengambil atribut-atribut artikel berita, seperti judul, nama penulis, waktu publikasi, lokasi berita, teks berita, dan tagar berita dengan kode di bawah ini:

```
import html

# vessel of array for new form of data
all_data = []

# from the collected data, loop all the data and extract it into new data
for news_url in raw_scrapped_links:
  # set empty vessel for one data
  one_product = {}

  # collect news content from current extracted news url
  # parse html using beautifulsoup
  news_soup = BeautifulSoup( getHtmlContent( news_url ).text, "html.parser" )

  # filter data based on their tag / class / id / element
  news_content_raw = news_soup.find( 'article', attrs={ "class":"detail" } )

  # extract news header content
  news_header = news_content_raw.find( attrs={"class": "detail__header"} )
  # get news title
  news_title        = news_header.find( attrs={"class": "detail__title"} ).text
  # get news author
  news_author_info  = news_header.find( attrs={"class": "detail__author"} ).text
  news_author_array = news_author_info.split(' - ')
  news_author       = news_author_array[0]
  news_division     = news_author_array[1]

  # get news published date
  news_date         = news_header.find( attrs={"class": "detail__date"} ).text

  # extract news media content
  news_media        = news_content_raw.find( "div", attrs={"class": "detail__media"} )
  news_img_src      = news_media.find( "img" )
  news_img_src      = news_img_src.get( 'src' ) if news_img_src else "N/A"
  news_img_caption  = news_media.find( "figcaption", attrs={"class": "detail__media-caption"} )
  news_img_caption  = news_img_caption.text if news_img_caption else "N/A"

  # extract news body
  news_body         = news_content_raw.find( "div", attrs={"class": "detail__body-text itp_bodycontent"} )
  # extract news location
  news_location     = news_body.find( "strong" ).text
```

Kode untuk mengambil konten dari kumpulan tautan yang sudah dikumpulkan sebelumnya serta pengolahannya - bagian 1

```
  # extract news text
  news_description_final = []
  news_detail_multiple_pages = news_body.find( "div", attrs={"class": "detail__multiple"} )

  # if news content separated into different pages, then
  if news_detail_multiple_pages:

    # ambil semua link berita tiap detil halaman
    array_details = news_detail_multiple_pages.find_all( "a", {"class": "detail__anchor-numb"} )
    for news_detail_page in array_details:
      # get current detil page url
      detil_url = news_detail_page.get( "href" )

      # get all news paragraph
      detil_soup = BeautifulSoup( getHtmlContent( detil_url ).text, "html.parser" )
      detil_body = detil_soup.find( "div", attrs={"class": "detail__body-text itp_bodycontent"} )

      # store result
      news_description_final.extend( extract_news_text( detil_body ) )
    # end if
  else:
    # store result
    news_description_final = extract_news_text( news_body )
  # end if

  # extract news tag(s)
  news_tags_raw = news_body.find_all( attrs={"class": "nav__item"} )
  news_tags_final = []
  for tags in news_tags_raw:
    news_tags_final.append( tags.get( "dtr-ttl" ) )
  # end loop
```

```
   # combine all values into one data with their specific key
   one_product['judul']            = news_title.strip()
   one_product['link']             = news_url
   one_product['penulis']          = news_author
   one_product['divisi']           = news_division
   one_product['waktu_publikasi']  = news_date
   one_product['ilustrasi_gambar'] = news_img_src
   one_product['ilustrasi_caption']= news_img_caption.strip()
   one_product['lokasi_berita']    = news_location
   one_product['teks_berita']      = news_description_final
   one_product['tagar_berita']     = news_tags_final

   # appen data to final product list
   all_data.append( one_product )
# end loop

# check numbers of filtered data
print( "number of all_data: ", len( all_data ) )
```

```
number of all_data:  209
```

Kode untuk mengambil konten dari kumpulan tautan yang sudah dikumpulkan sebelumnya serta pengolahannya - bagian 2

Berikut ini adalah contoh data yang sudah terpetakan berdasarkan jenis atribut / informasinya.

```
[16] # display 5 random data
     df.sample( 5 )
```

	judul	link	penulis	divisi	waktu_publikasi	ilustrasi_gambar	ilustrasi_caption	lokasi_berita	teks_berita	tagar_berita
83	Polres Malang Edukasi Supeltas Sambil Bukber d...	https://news.detik.com/berita/d-7264516/polres...	Fajar Pratama	detikNews	Rabu, 27 Mar 2024 15:25 WIB	https://awsimages.detik.net.id/community/media...	Satlantas Polres Malang membina sukarelawan pe...	Malang	[Satuan Lalu Lintas (Satlantas) Polres Malang ...	['polres malang', 'teman mudik 2024', 'supelta...
108	Kasus Demam Berdarah Dengue di Depok Capai 328...	https://news.detik.com/berita/d-7264217/kasus-...	Devi Puspitasari	detikNews	Rabu, 27 Mar 2024 13:46 WIB	https://awsimages.detik.net.id/community/media...	Ilustrasi nyamuk Aedes aegypti penyebab penyak...	Depok	[Sedikitnya 328 orang per Februari 2024 terken...	['jabodetabek', 'depok', 'demam berdarah dengu...
37	Program Magang ke Jepang Diklaim Tingkatkan Sk...	https://news.detik.com/berita/d-7264935/progra...	Jihaan Khoirunnisa	detikNews	Rabu, 27 Mar 2024 17:53 WIB	https://awsimages.detik.net.id/community/media...	Foto: Kemnaker	Jakarta	[Menteri Ketenagakerjaan (Menaker) Ida Fauziya...	['kemnaker', 'tenaga kerja']
88	Pemkab Lebak Pastikan Stok-Harga Bahan Pokok S...	https://news.detik.com/berita/d-7264477/pemkab...	Fathul Rizkoh	detikNews	Rabu, 27 Mar 2024 15:15 WIB	https://awsimages.detik.net.id/community/media...	Foto: Ilustrasi bahan pokok (dok. Kemendag).	Lebak	[Penjabat Bupati Lebak Iwan Kurniawan memastik...	['pemkab lebak', 'lebak', 'banten', 'lebaran',...
65	Bareskrim Beberkan 3 Tersangka TPPO Ferienjob ...	https://news.detik.com/berita/d-7264715/baresk...	Rumondang Naibaho	detikNews	Rabu, 27 Mar 2024 16:24 WIB	https://awsimages.detik.net.id/community/media...	Direktur Tindak Pidana Umum (Dirtipidum) Bares...	Jakarta	[Direktur Tindak Pidana Umum (Dirtipidum) Bare...	['ferienjob', 'ferienjob adalah', 'ferienjob j...

1.4 Data yang terkumpul kemudian disimpan pada dokumen berformat csv dan excel dan penulisan kode-nya adalah sebagai berikut.

```
import pandas as pd
import csv

# turn collected data in form of dataframe
df = pd.DataFrame( all_data )
df.head( 10 ) # list the first 20 rows

# export collected data into excel file
df.to_excel( 'news_data_03272024.xlsx', index=False )

# export collected data into csv file
df.to_csv( 'news_data_03272024.csv', index=False )
```

2. Data Preprocessing

Pada proses ini data yang terkumpul akan melalui praproses yang terbagi dalam beberapa tahap, yaitu:

2.1 instalasi dan import library untuk membantu pengolahan data

```
[12] !pip install nltk pandas numpy matplotlib
```

```
[ ] import codecs
    import collections

    import numpy as np
    import pandas as pd
    import nltk
    from nltk.stem import PorterStemmer
    from nltk.tokenize import WordPunctTokenizer
    import matplotlib
```

2.2 membuka dokumen csv yang berisi data artikel berita yang dikumpulkan pada tahap sebelumnya

```
import ast

# Custom converter function to safely parse list values
def parse_list(s):
    try:
        return ast.literal_eval(s)
    except (SyntaxError, ValueError):
        return None  # Handle invalid list representations
    # end try
# end function
```

```
[15] # try to open csv file
     # df = pd.read_csv( 'news_data_03272024.csv' )
     # khusus column teks berita yg berbentuk list
     df = pd.read_csv( 'news_data_03272024.csv', converters={ "teks_berita": parse_list } )
```

2.3 memilih atribut judul dan teks_berita sebagai atribut artikel berita yang akan diolah, khusus pada atribut teks_berita yang berbentuk list diubah dahulu menjadi satu teks panjang

```
[18] # Using DataFrame.copy() create new DaraFrame
     df_data = df[ [ 'judul', 'teks_berita' ] ].copy()
     df_data['teks_berita'] = df_data['teks_berita'].tolist()
     # df_data[ 'count_paragraph' ] = df_data[ 'teks_berita' ].apply( lambda words: len( words ) )

     # merge news text from per paragraph into one text value
     df_data[ 'teks_berita' ] = df_data[ 'teks_berita' ].apply( lambda words: ' '.join( words ) )

     # check selected values
     df_data.sample( 5 )
```

2.4 dilakukan pembersihan teks pada kedua atribut dengan mengubah karakter menjadi huruf kecil, menghilangkan format url, serta menghapus semua karakter khusus dan angka

```
import nltk
```

```
[20] # i) converting to lowercase
     df_data[ 'cleaned_judul' ]  = df_data[ 'judul' ].apply( lambda words: words.lower() if isinstance( words, str ) else words ).copy()
     df_data[ 'cleaned_berita' ] = df_data[ 'teks_berita' ].apply( lambda words: words.lower() if isinstance( words, str ) else words ).copy()

     # ii) removing url using regex
     import re
     url_pattern = re.compile(r'https?://\S+')
     df_data[ 'cleaned_judul' ]  = df_data[ 'cleaned_judul' ].apply( lambda words: url_pattern.sub( '', words ) )
     df_data[ 'cleaned_berita' ] = df_data[ 'cleaned_berita' ].apply( lambda words: url_pattern.sub( '', words ) )

     # iii) remove non-word and non-whitespace char
     df_data[ 'cleaned_judul' ]  = df_data[ 'cleaned_judul' ].replace( to_replace=r'[^\w\s]', value='', regex=True )
     df_data[ 'cleaned_berita' ] = df_data[ 'cleaned_berita' ].replace( to_replace=r'[^\w\s]', value='', regex=True )

     # iv) remove digits
     df_data[ 'cleaned_judul' ]  = df_data[ 'cleaned_judul' ].replace( to_replace=r'\d', value='', regex=True )
     df_data[ 'cleaned_berita' ] = df_data[ 'cleaned_berita' ].replace( to_replace=r'\d', value='', regex=True )

     # check values
     df_data[ [ 'cleaned_judul', 'cleaned_berita' ] ].head( 5 )
```

2.5 lanjut memproses kedua atribut melalui tokenisasi

```
# get tokenizer libraries
nltk.download('punkt')
from nltk.tokenize import word_tokenize

# apply tokenization
df_data[ 'tokenized_judul' ]  = df_data[ 'cleaned_judul' ].apply( word_tokenize ).copy()
df_data[ 'tokenized_berita' ] = df_data[ 'cleaned_berita' ].apply( word_tokenize ).copy()

# check values
df_data[ [ 'tokenized_judul', 'tokenized_berita' ] ].head( 5 )
```

2.6 lanjut memproses kedua atribut yang sudah berbentuk token melalui penghapusan kata berjenis stopword

```
[22] # get stopword libraries
     nltk.download('stopwords')
     from nltk.corpus import stopwords

     # load stopwords
     stopwords_ind = stopwords.words( 'indonesian' )

     # apply stopwords removal
     df_data[ 'stopremoved_judul' ]  = df_data[ 'tokenized_judul' ].apply( lambda words: [ word for word in words if word not in stopwords_ind ] ).copy()
     df_data[ 'stopremoved_berita' ] = df_data[ 'tokenized_berita' ].apply( lambda words: [ word for word in words if word not in stopwords_ind ] ).copy()

     # check values
     df_data[ [ 'stopremoved_judul', 'stopremoved_berita' ] ].head( 5 )
```

2.7 kedua atribut yang sudah berbentuk token kemudian melalui proses stemming

```
[23] # install Sastrawi to be able to do stemming indonesian text
     !pip install Sastrawi
```

```
Collecting Sastrawi
  Downloading Sastrawi-1.0.1-py2.py3-none-any.whl (209 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 209.7/209.7 kB 3.9 MB/s eta 0:00:00
Installing collected packages: Sastrawi
Successfully installed Sastrawi-1.0.1
```

```
[24] # from nltk.stem import PorterStemmer
     # from nltk.tokenize import word_tokenize
     # # initialize the porter stemmer
     # stemmer = PorterStemmer()

     from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

     # create a stemmer object
     stemmer = StemmerFactory().create_stemmer()

     # apply stemming on the text column
     df_data[ 'stemming_judul' ]  = df_data[ 'stopremoved_judul' ].apply( lambda words: [ stemmer.stem( word ) for word in words ] ).copy()
     df_data[ 'stemming_berita' ] = df_data[ 'stopremoved_berita' ].apply( lambda words: [ stemmer.stem( word ) for word in words ] ).copy()

     # check values
     df_data[ [ 'stemming_judul', 'stemming_berita' ] ].head( 5 )
```

2.8 kedua atribut yang sudah berbentuk token kemudian melalui proses lemmatization

```
nltk.download( 'averaged_perceptron_tagger' )
nltk.download( 'wordnet' )

from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet
import pandas as pd

# initialize the lemmatizer
lemmatizer = WordNetLemmatizer()

# convert POS tag into WordNet format
def get_wordnet_pos( word ):
  tag = nltk.pos_tag( [word] )[0][1][0].upper()
  tag_dict = {"J": wordnet.ADJ,
              "N": wordnet.NOUN,
              "V": wordnet.VERB,
              "J": wordnet.ADV}
  return tag_dict.get( tag, wordnet.NOUN )
# end function

# apply lemmatization to column of dataframe
df_data[ 'lemma_judul' ]  = df_data[ 'stopremoved_judul' ].apply( lambda words: [ lemmatizer.lemmatize( word, get_wordnet_pos( word ) ) for word in words ] ).copy()
df_data[ 'lemma_berita' ] = df_data[ 'stopremoved_berita' ].apply( lambda words: [ lemmatizer.lemmatize( word, get_wordnet_pos( word ) ) for word in words ] ).copy()

# check values
df_data[ [ 'lemma_judul', 'lemma_berita' ] ].head( 5 )
```

2.9 hasil proses tokenisasi, stemming, dan lemmatization disimpan pada dokumen csv dan excel yang baru

```
[122] import pandas as pd
      import csv

      # selected dataframe
      getData = df_data[ [ 'judul', 'teks_berita', 'stopremoved_judul', 'stopremoved_berita', 'stemming_judul', 'stemming_berita', 'lemma_judul', 'lemma_berita'] ]

      # export pre processed data into excel file
      getData.to_excel( 'news_data_03272024-preprocessed.xlsx', index=False )

      # export pre processed collected data into csv file
      getData.to_csv( 'news_data_03272024-preprocessed.csv', index=False )
```

3. Exploratory Data Analysis (EDA)

Tahap selanjutnya adalah melakukan analisis eksplorasi data di mana tiap token atribut judul dan teks berita akan dijadikan dalam dua bentuk, yaitu *word cloud* dan *histogram of word frequencies*.

3.1 *Word cloud*

Dalam membuat word cloud, pertama semua token dari seluruh baris data perlu digabungkan menjadi satu teks panjang untuk masing-masing atribut tokennya dan kode-nya adalah sebagai berikut:

```
[ ] import ast

    # Custom converter function to safely parse list values
    def parse_list(s):
        try:
            return ast.literal_eval(s)
        except (SyntaxError, ValueError):
            return None  # Handle invalid list representations
        # end try
    # end function
```

```
[197] # try to open previous saved pre processed data
      import numpy as np
      import pandas as pd

      # open file to get the data
      df_pre = pd.read_csv( 'news_data_03272024-preprocessed.csv', converters={ "teks_berita": parse_list,
                                                                                "stopremoved_judul": parse_list,
                                                                                "stopremoved_berita": parse_list,
                                                                                "lemma_judul": parse_list,
                                                                                "lemma_berita": parse_list,
                                                                                "stemming_judul": parse_list,
                                                                                "stemming_berita": parse_list
                                                                                } )

      # create new dataframe for lemma pre processed data
      df_eda_1 = pd.DataFrame()
      df_eda_1[ 'words_judul' ]  = df_pre[ 'lemma_judul' ].copy()
      df_eda_1[ 'words_berita' ] = df_pre[ 'lemma_berita' ].copy()

      # create new dataframe for stemming pre processed data
      df_eda_2 = pd.DataFrame()
      df_eda_2[ 'words_judul' ]  = df_pre[ 'stemming_judul' ].copy()
      df_eda_2[ 'words_berita' ] = df_pre[ 'stemming_berita' ].copy()

      # create new dataframe for cleaned pre processed data
      df_eda_3 = pd.DataFrame()
      df_eda_3[ 'words_judul' ]  = df_pre[ 'stopremoved_judul' ].copy()
      df_eda_3[ 'words_berita' ] = df_pre[ 'stopremoved_berita' ].copy()
```

Lalu dibuatlah sebuah fungsi yang akan membuat grafik word cloud dari input judul / teks berita yang jadi masukan dari fungsi tersebut dan untuk meminimalisir penulisan kode yang sama secara berulang. Kode fungsi tersebut adalah sebagai berikut:

```
[199] import matplotlib.pyplot as plt
      from wordcloud import WordCloud

      def createWordCloud( df_one_column_value ):
        # flat the values of judul
        clean_judul_flat_arr = [ word for sublist in df_one_column_value for word in sublist ]
        clean_judul_flat_text = " ".join( clean_judul_flat_arr )

        # create a word cloud for judul
        wordcloud_judul = WordCloud( width=800, height=400 ).generate( clean_judul_flat_text )

        # display the word cloud for judul
        plt.figure( figsize=( 10, 5 ) )
        plt.imshow( wordcloud_judul, interpolation='bilinear' )
        plt.axis( "off" )
        plt.show()
      # end function
```

Berikut ini adalah word cloud dari token judul yang melalui proses lemmatization:

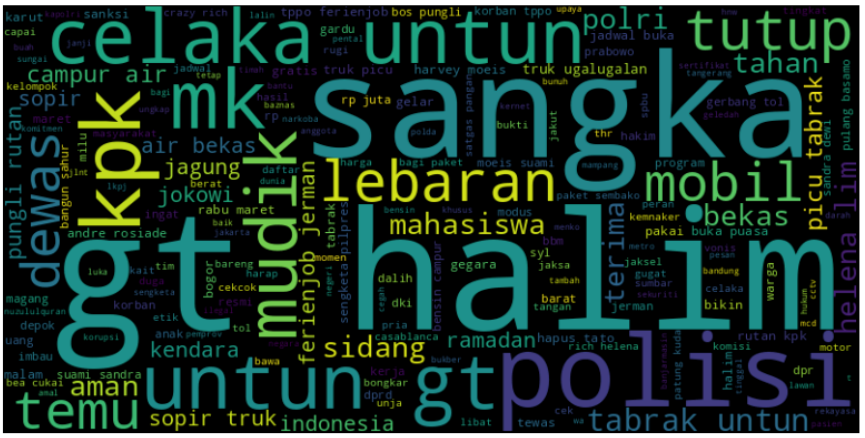
```
# show wrodcloud for df eda 1 judul
createWordCloud( df_eda_1['words_judul'].copy() )
```

Berikut ini adalah word cloud dari token teks berita yang melalui proses lemmatization:

```
# show wrodcloud for df eda 1 berita
createWordCloud( df_eda_1['words_berita'].copy() )
```

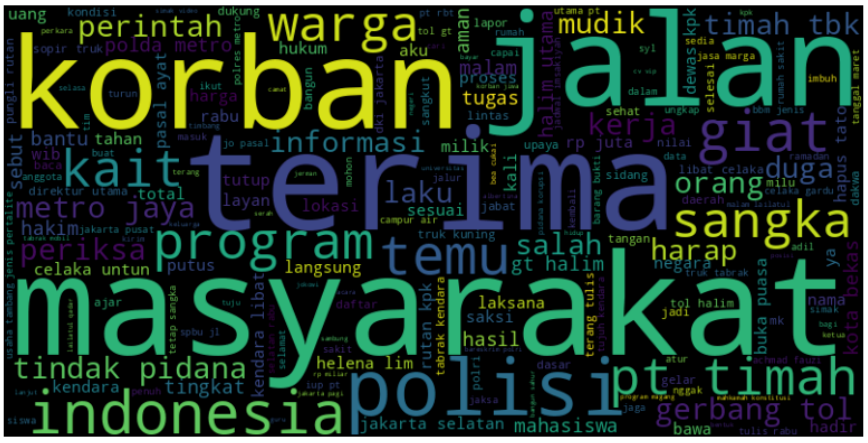
Berikut ini adalah word cloud dari token judul yang melalui proses stemming:

```
# show wrodcloud for df eda 2 judul
createWordCloud( df_eda_2['words_judul'].copy() )
```



Berikut ini adalah word cloud dari token teks berita yang melalui proses stemming:

```
[204] # show wrodcloud for df eda 2 berita
      createWordCloud( df_eda_2['words_berita'].copy() )
```



Berikut ini adalah word cloud dari token judul yang tidak melalui proses lemmatization ataupun stemming:

```
[ ] # show wrodcloud for df eda 3 judul
    createWordCloud( df_eda_3['words_judul'].copy() )
```

Berikut ini adalah word cloud dari token teks berita yang tidak melalui proses lemmatization ataupun stemming:

```
# show wrodcloud for df eda 3 berita
createWordCloud( df_eda_3['words_berita'].copy() )
```

3.2 *Histogram of word frequencies*

Referensi: [Word Frequency with Python]

Dalam histogram dengan nilai frekuensi kemunculan kata, pertama semua token dari seluruh baris data perlu digabungkan menjadi satu teks panjang untuk masing-masing atribut tokennya. Setelah seluruh gabungan token pada masing-masing atribut didapatkan, maka selanjutnya dipisahkan dalam satu list token baru dan dilakukan perhitungan frekuensi kemunculan tiap token dengan menggunakan Counter dari *library* collections. Pada perhitungan frekuensi token juga digunakan metode ngram yang bertujuan untuk mendapatkan frekuensi dari sepasang atau sekelompok kata. Untuk proses ini dibuat dalam sebuah kode fungsi yang ditulis sebagai berikut:

```
# Ngrams allows to group words in common pairs or trigrams..etc
from nltk import ngrams

# to count the objects
from collections import Counter

# visual library
import seaborn as sns
import matplotlib.pyplot as plt

# to generate histogram of word frequencies
def word_frequency( list_words ):
  # joins all the sentenses
  clean_words_flat_arr = [ word for sublist in list_words.copy() for word in sublist ]
  clean_words_flat_text = " ".join( clean_words_flat_arr )

  # tokens of all joined words
  new_tokens = clean_words_flat_text.split()
  # print( "len total tokens:",len( new_tokens ) )

  # counts the words, pairs and trigrams
  counted   = Counter( new_tokens )
  counted_2 = Counter( ngrams( new_tokens, 2 ) )
  new_counted_2 = { ", ".join( pair ): freq for pair, freq in counted_2.items() }
  counted_3 = Counter( ngrams( new_tokens, 3 ) )
  new_counted_3 = { ", ".join( pair ): freq for pair, freq in counted_3.items() }

  # creates 3 data frames and returns them
  word_freq  = pd.DataFrame( counted.items(), columns=['word','frequency'] ).sort_values( by='frequency', ascending=False )
  word_pairs = pd.DataFrame( new_counted_2.items(), columns=['pairs', 'frequency'] ).sort_values( by='frequency', ascending=False )
  trigrams   = pd.DataFrame( new_counted_3.items(), columns=['trigrams','frequency'] ).sort_values( by='frequency', ascending=False )

  # return results
  return word_freq, word_pairs, trigrams
```

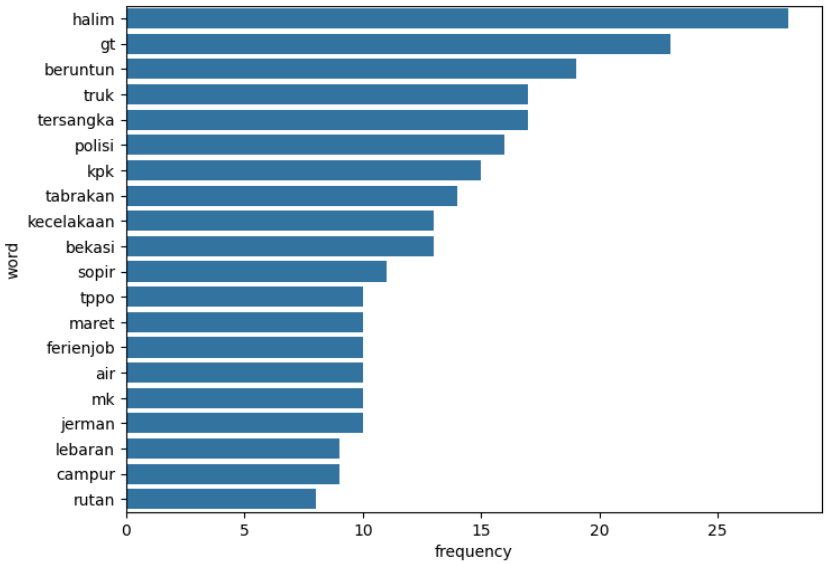
Lalu fungsi di atas digunakan pada masing-masing **token judul** hasil **lemmatization, stemming,** dan **stopwords** sebagai berikut:

```
# for df eda 1 (token judul of lemmatization)
unigram_eda1, bigram_eda1, trigram_eda1 = word_frequency( df_eda_1['words_judul'] )

# for df eda 2 (token judul of stemming)
unigram_eda2, bigram_eda2, trigram_eda2 = word_frequency( df_eda_2['words_judul'] )

# for df eda 3 (token judul of stopwords)
unigram_eda3, bigram_eda3, trigram_eda3 = word_frequency( df_eda_3['words_judul'] )
```

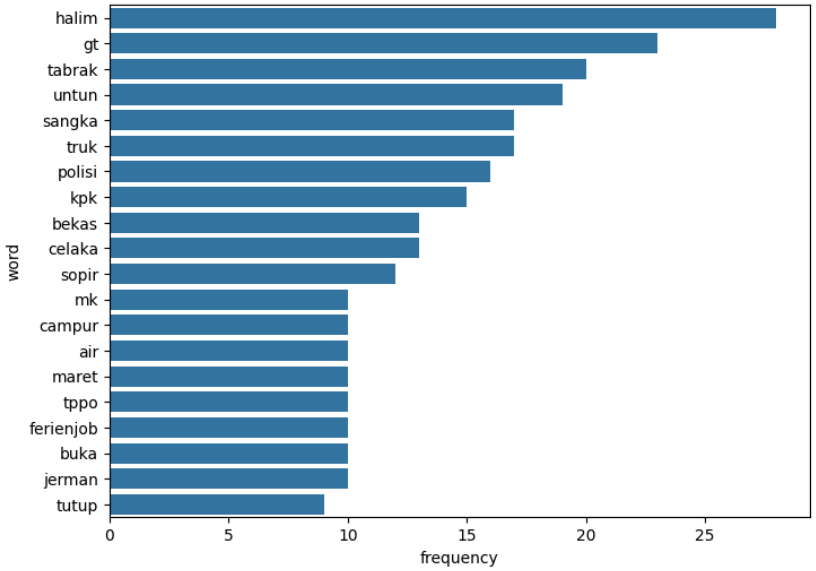
Untuk grafik frekuensi masing-masing **token judul** dalam **bentuk unigram** dituliskan dalam kode berikut ini:

```
# create subplot of the different unigram data frames
fig, axes = plt.subplots( 3, 1, figsize=( 8, 20 ) )
sns.barplot( ax=axes[0], x='frequency', y='word', data=unigram_eda1.head( 20 ) )
sns.barplot( ax=axes[1], x='frequency', y='word', data=unigram_eda2.head( 20 ) )
sns.barplot( ax=axes[2], x='frequency', y='word', data=unigram_eda3.head( 20 ) )
```

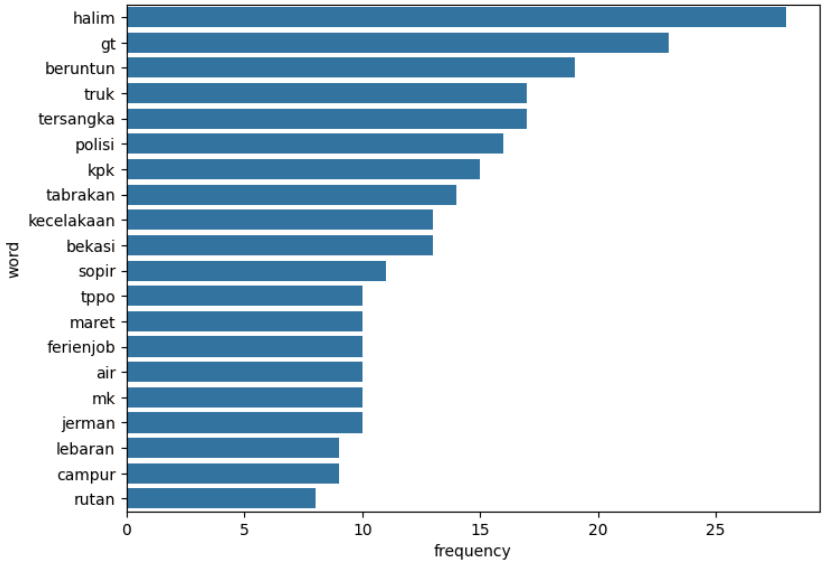
Berikut ini adalah grafik frekuensi unigram untuk **atribut judul** berbentuk **token lemmatization**:



Berikut ini adalah grafik frekuensi unigram untuk **atribut judul** berbentuk **token stemming**:



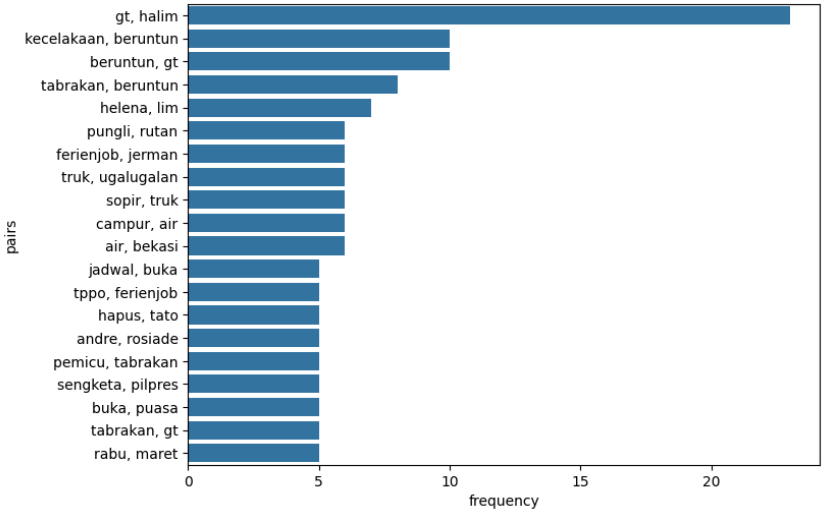
Berikut ini adalah grafik frekuensi unigram untuk **atribut judul** berbentuk **token stopwords**:



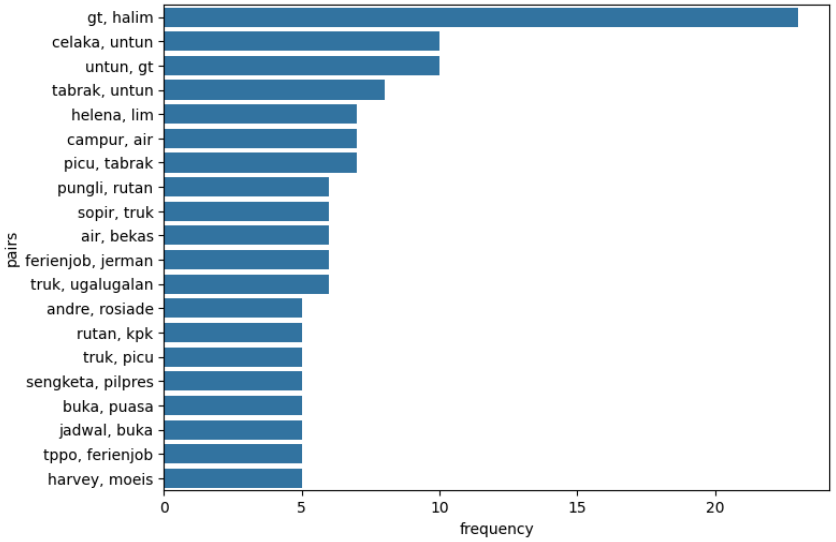
Untuk grafik frekuensi masing-masing **token judul** dalam **bentuk bigram** dituliskan dalam kode berikut ini:

```
[149] # create subplot of the different bigram data frames
      fig, axes = plt.subplots( 3, 1, figsize=( 8, 20 ) )
      sns.barplot( ax=axes[0], x='frequency', y='pairs', data=bigram_eda1.head( 20 ) )
      sns.barplot( ax=axes[1], x='frequency', y='pairs', data=bigram_eda2.head( 20 ) )
      sns.barplot( ax=axes[2], x='frequency', y='pairs', data=bigram_eda3.head( 20 ) )
```

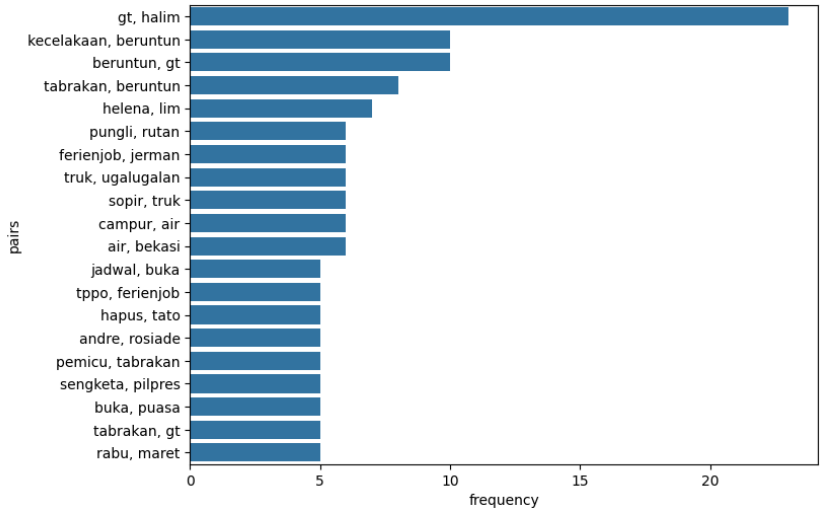
Berikut ini adalah grafik frekuensi unigram untuk **atribut judul** berbentuk **token lemmatization**:



Berikut ini adalah grafik frekuensi unigram untuk **atribut judul** berbentuk **token stemming**:



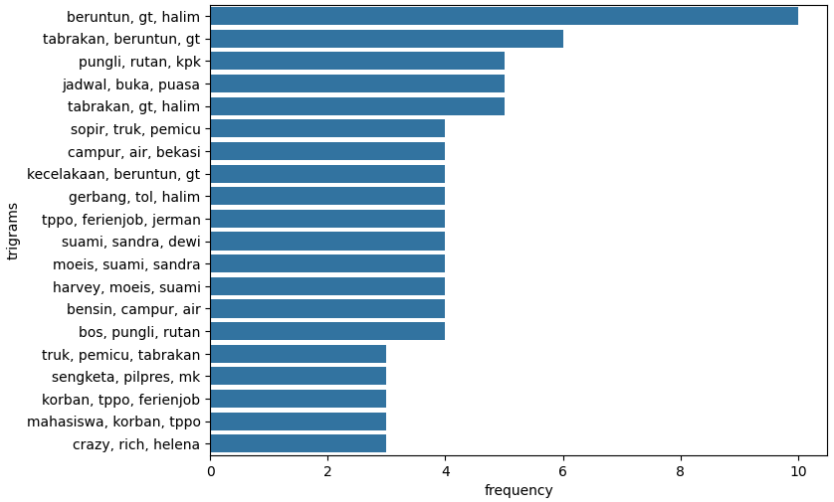
Berikut ini adalah grafik frekuensi unigram untuk **atribut judul** berbentuk **token stopwords**:



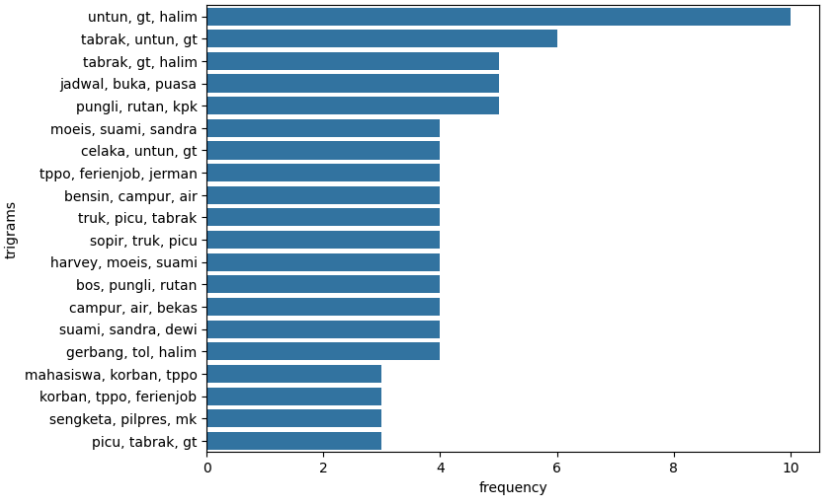
Untuk grafik frekuensi masing-masing **token judul** dalam **bentuk trigram** dituliskan dalam kode berikut ini beserta hasilnya:

```
[151] # create subplot of the different trigram data frames
      fig, axes = plt.subplots( 3, 1, figsize=( 8, 20 ) )
      sns.barplot( ax=axes[0], x='frequency', y='trigrams', data=trigram_eda1.head( 20 ) )
      sns.barplot( ax=axes[1], x='frequency', y='trigrams', data=trigram_eda2.head( 20 ) )
      sns.barplot( ax=axes[2], x='frequency', y='trigrams', data=trigram_eda3.head( 20 ) )
```

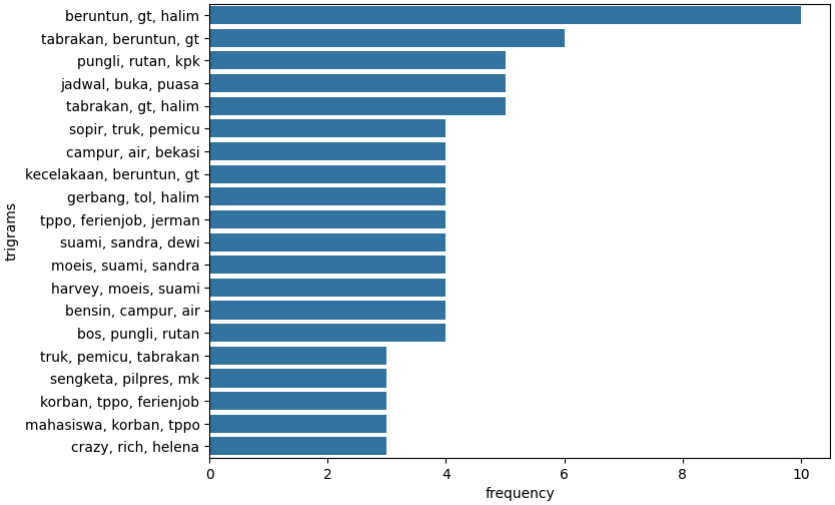
Berikut ini adalah grafik frekuensi unigram untuk **atribut judul** berbentuk **token lemmatization**:



Berikut ini adalah grafik frekuensi unigram untuk **atribut judul** berbentuk **token stemming**:



Berikut ini adalah grafik frekuensi unigram untuk **atribut judul** berbentuk **token stopwords**:



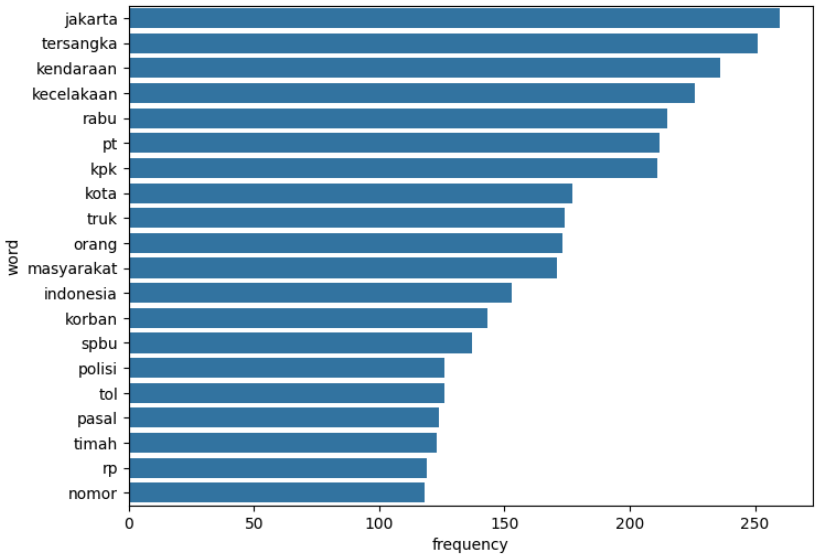
Lalu fungsi tersebut juga digunakan pada masing-masing **token berita** hasil **lemmatization, stemming,** dan **stopwords** sebagai berikut:

```
[255] # for df eda 1 (token berita of lemmatization)
      unigram_eda1, bigram_eda1, trigram_eda1 = word_frequency( df_eda_1['words_berita'] )

      # for df eda 2 (token berita of stemming)
      unigram_eda2, bigram_eda2, trigram_eda2 = word_frequency( df_eda_2['words_berita'] )

      # for df eda 3 (token berita of stopwords)
      unigram_eda3, bigram_eda3, trigram_eda3 = word_frequency( df_eda_3['words_berita'] )
```

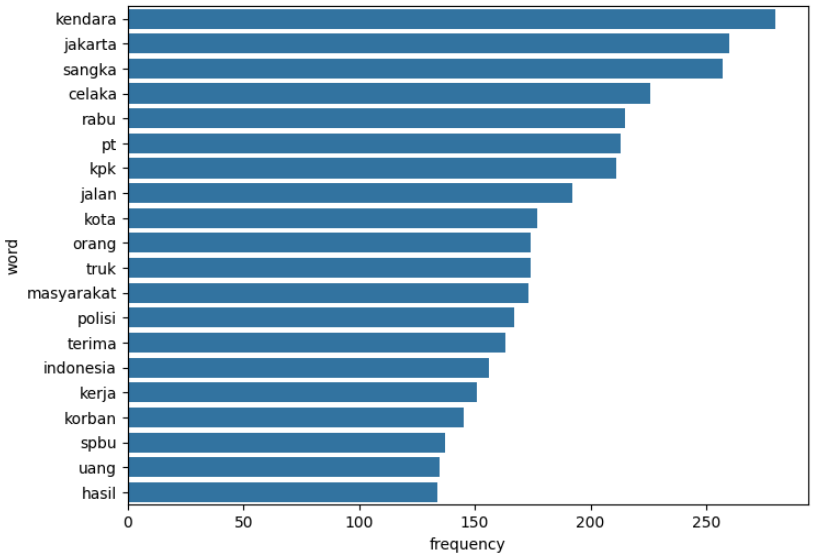
Untuk grafik frekuensi masing-masing **token berita** dalam **bentuk unigram** dituliskan dalam kode berikut ini:

```
# create subplot of the different unigram token berita data frames
fig, axes = plt.subplots( 3, 1, figsize=( 8, 20 ) )
sns.barplot( ax=axes[0], x='frequency', y='word', data=unigram_eda1.head( 20 ) )
sns.barplot( ax=axes[1], x='frequency', y='word', data=unigram_eda2.head( 20 ) )
sns.barplot( ax=axes[2], x='frequency', y='word', data=unigram_eda3.head( 20 ) )
```

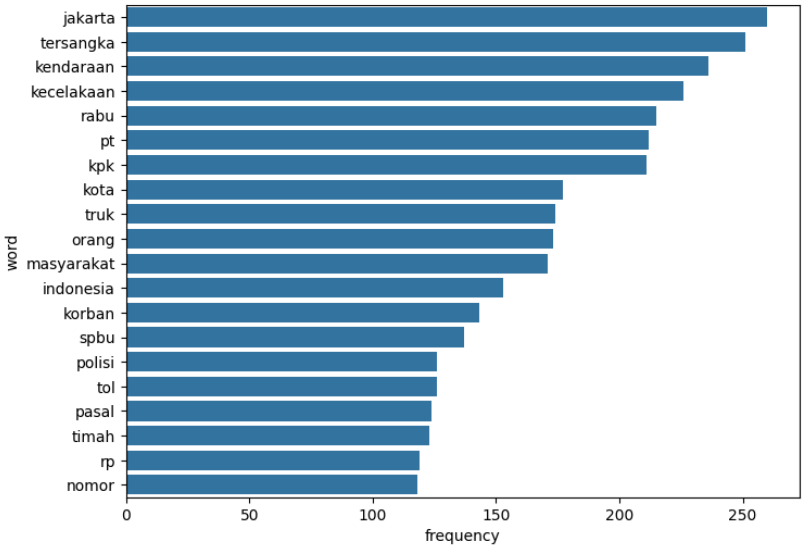
Berikut ini adalah grafik frekuensi unigram untuk **atribut berita** berbentuk **token lemmatization**:



Berikut ini adalah grafik frekuensi unigram untuk **atribut berita** berbentuk **token stemming**:



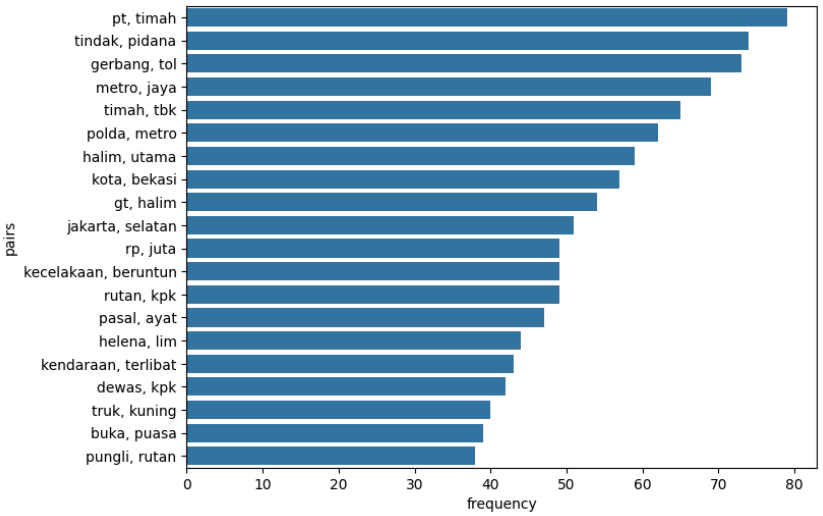
Berikut ini adalah grafik frekuensi unigram untuk **atribut berita** berbentuk **token stopwords**:



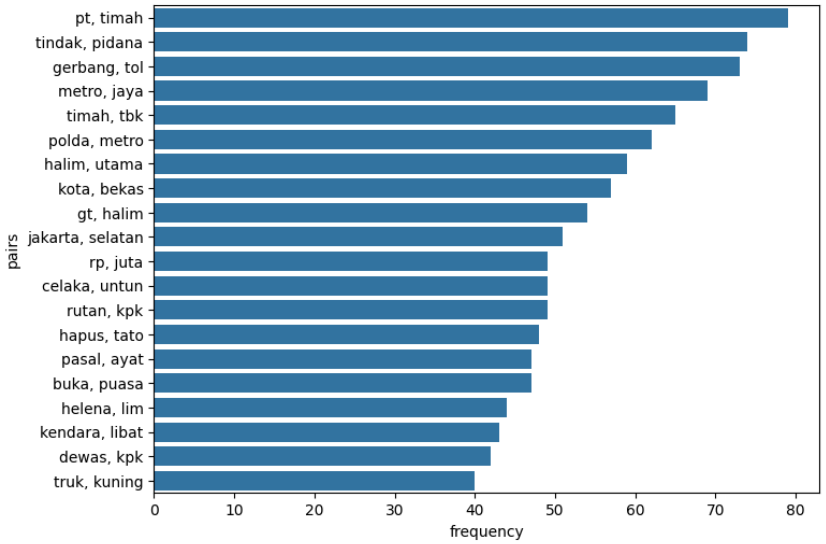
Untuk grafik frekuensi masing-masing **token berita** dalam **bentuk bigram** dituliskan dalam kode berikut ini:

```
[266] # create subplot of the different bigram token berita data frames
      fig, axes = plt.subplots( 3, 1, figsize=( 8, 20 ) )
      sns.barplot( ax=axes[0], x='frequency', y='pairs', data=bigram_eda1.head( 20 ) )
      sns.barplot( ax=axes[1], x='frequency', y='pairs', data=bigram_eda2.head( 20 ) )
      sns.barplot( ax=axes[2], x='frequency', y='pairs', data=bigram_eda3.head( 20 ) )
```

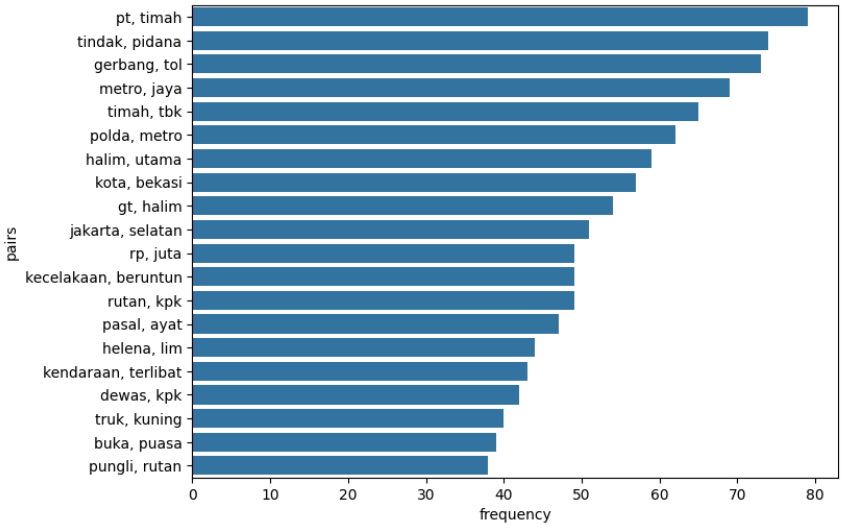
Berikut ini adalah grafik frekuensi unigram untuk **atribut berita** berbentuk **token lemmatization**:



Berikut ini adalah grafik frekuensi unigram untuk **atribut berita** berbentuk **token stemming**:



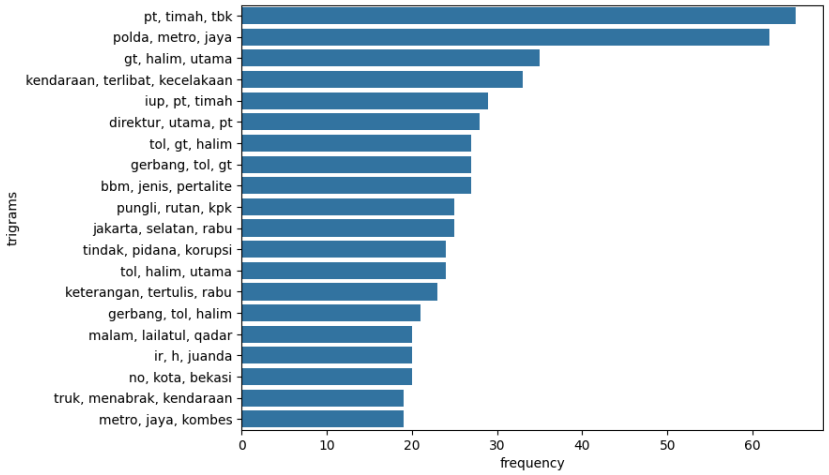
Berikut ini adalah grafik frekuensi unigram untuk **atribut berita** berbentuk **token stopwords**:



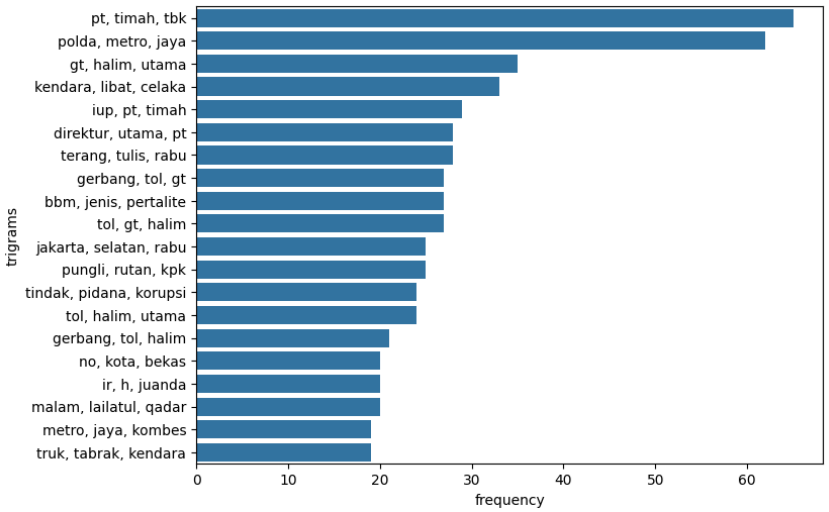
Untuk grafik frekuensi masing-masing **token berita** dalam **bentuk trigram** dituliskan dalam kode berikut ini beserta hasilnya:

```
# create subplot of the different trigram token berita data frames
fig, axes = plt.subplots( 3, 1, figsize=( 8, 20 ) )
sns.barplot( ax=axes[0], x='frequency', y='trigrams', data=trigram_eda1.head( 20 ) )
sns.barplot( ax=axes[1], x='frequency', y='trigrams', data=trigram_eda2.head( 20 ) )
sns.barplot( ax=axes[2], x='frequency', y='trigrams', data=trigram_eda3.head( 20 ) )
```

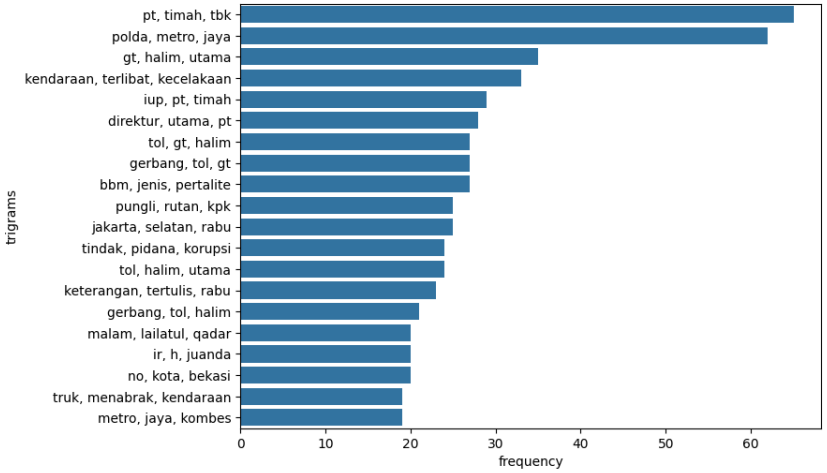
Berikut ini adalah grafik frekuensi unigram untuk **atribut berita** berbentuk **token lemmatization**:



Berikut ini adalah grafik frekuensi unigram untuk **atribut judul** berbentuk **token stemming**:



Berikut ini adalah grafik frekuensi unigram untuk **atribut judul** berbentuk **token stopwords**:



4. Text Mining Techniques (Sentiment Analysis)

Tahap selanjutnya adalah melakukan analisis sentimen pada atribut judul dan berita di mana tiap token atribut judul dan berita akan ditentukan nilai sentimennya, antara sentimen positif, netral, atau negatif. Pertama data didapatkan dengan membuka dokumen csv yang berisi data yang telah diproses pada tahap pra proses dengan kode sebagai berikut.

```
import ast

# Custom converter function to safely parse list values
def parse_list( s ):
    try:
        return ast.literal_eval( s )
    except ( SyntaxError, ValueError ):
        return None  # Handle invalid list representations
    # end try
# end function


# try to open previous saved pre processed data
import numpy as np
import pandas as pd

# open file to get the data
df_pre = pd.read_csv( 'news_data_03272024-preprocessed.csv', converters={ "stopremoved_judul": parse_list,
                                                                          "stopremoved_berita": parse_list,
                                                                          "lemma_judul": parse_list,
                                                                          "lemma_berita": parse_list,
                                                                          "stemming_judul": parse_list,
                                                                          "stemming_berita": parse_list
                                                                          } )

# create new dataframe for lemma pre processed data
df_sentiment = pd.DataFrame()
df_sentiment[ 'lemma_judul' ]    = df_pre[ 'lemma_judul' ].copy()
df_sentiment[ 'lemma_berita' ]   = df_pre[ 'lemma_berita' ].copy()

df_sentiment[ 'stemming_judul' ] = df_pre[ 'stemming_judul' ].copy()
df_sentiment[ 'stemming_berita' ]= df_pre[ 'stemming_berita' ].copy()

df_sentiment[ 'cleaned_judul' ]  = df_pre[ 'stopremoved_judul' ].copy()
df_sentiment[ 'cleaned_berita' ] = df_pre[ 'stopremoved_berita' ].copy()
```

Lalu dibuatlah fungsi untuk menilai sentimen suatu teks masukan dengan menggunakan library textblob dan penulisan kode fungsinya adalah sebagai berikut.

```
[155] from textblob import TextBlob

      sentiment_cateogry_list =[ "positive", "neutral", "negative" ]

      # to get sentiment polarity
      def getSentimentValue( text ):
        # analyze the input
        analysis = TextBlob( text )
        sentiment = analysis.sentiment.polarity

        # decide the sentimen ppolarity score
        if sentiment > 0:
          return sentiment_cateogry_list[0]
        elif sentiment < 0:
          return sentiment_cateogry_list[2]
        else:
          return sentiment_cateogry_list[1]
        # end if
      # end function
```

Fungsi getSentimentValue tersebut kemudian digunakan pada **token judul hasil lemmatization** sebagai berikut:

```
[ ]  df_sentiment['merged_lemma_judul']    = df_sentiment[ 'lemma_judul' ].apply( lambda words: " ".join( words ) ).copy()
     df_sentiment['sentiment_lemma_judul'] = df_sentiment[ 'merged_lemma_judul' ].apply( getSentimentValue ).copy()
     df_sentiment['sentiment_lemma_judul'].value_counts()

     neutral     204
     negative      5
     Name: sentiment_lemma_judul, dtype: int64
```

Fungsi getSentimentValue tersebut kemudian digunakan pada **token berita hasil lemmatization** sebagai berikut:

```
[ ]  df_sentiment['merged_lemma_judul']    = df_sentiment[ 'lemma_judul' ].apply( lambda words: " ".join( words ) ).copy()
     df_sentiment['sentiment_lemma_judul'] = df_sentiment[ 'merged_lemma_judul' ].apply( getSentimentValue ).copy()
     df_sentiment['sentiment_lemma_judul'].value_counts()

     neutral     204
     negative      5
     Name: sentiment_lemma_judul, dtype: int64
```

Fungsi getSentimentValue tersebut kemudian digunakan pada **token judul hasil stemming** sebagai berikut:

```
     df_sentiment['merged_stemming_judul']    = df_sentiment[ 'stemming_judul' ].apply( lambda words: " ".join( words ) ).copy()
     df_sentiment['sentiment_stemming_judul'] = df_sentiment[ 'merged_stemming_judul' ].apply( getSentimentValue ).copy()
     df_sentiment['sentiment_stemming_judul'].value_counts()

     neutral     203
     negative      6
     Name: sentiment_stemming_judul, dtype: int64
```

Fungsi getSentimentValue tersebut kemudian digunakan pada **token berita hasil stemming** sebagai berikut:

```
df_sentiment['merged_stemming_berita']    = df_sentiment[ 'stemming_berita' ].apply( lambda words: " ".join( words ) ).copy()
df_sentiment['sentiment_stemming_berita'] = df_sentiment[ 'merged_stemming_berita' ].apply( getSentimentValue ).copy()
df_sentiment['sentiment_stemming_berita'].value_counts()
```

```
neutral     142
negative     44
positive     23
Name: sentiment_stemming_berita, dtype: int64
```

Fungsi getSentimentValue tersebut kemudian digunakan pada **token judul hasil stopwords** sebagai berikut:

```
[164] df_sentiment['merged_cleaned_judul']    = df_sentiment[ 'cleaned_judul' ].apply( lambda words: " ".join( words ) ).copy()
      df_sentiment['sentiment_cleaned_judul'] = df_sentiment[ 'merged_cleaned_judul' ].apply( getSentimentValue ).copy()
      df_sentiment['sentiment_cleaned_judul'].value_counts()
```

```
neutral     204
negative      5
Name: sentiment_cleaned_judul, dtype: int64
```

Fungsi getSentimentValue tersebut kemudian digunakan pada **token berita hasil stopwords** sebagai berikut:

```
df_sentiment['merged_cleaned_berita']    = df_sentiment[ 'cleaned_berita' ].apply( lambda words: " ".join( words ) ).copy()
df_sentiment['sentiment_cleaned_berita'] = df_sentiment[ 'merged_cleaned_berita' ].apply( getSentimentValue ).copy()
df_sentiment['sentiment_cleaned_berita'].value_counts()
```

```
neutral     155
negative     31
positive     23
Name: sentiment_cleaned_berita, dtype: int64
```

5. Machine Learning Models (Text Classification)

Tahap terakhir adalah menggunakan atribut judul pada sebuah machine learning untuk tugas klasifikasi teks. Untuk melakukannya, pertama dibuat fungsi yang berisi sebuah mesin klasifikasi teks sebagai berikut.

```
[175] from sklearn.feature_extraction.text import TfidfVectorizer
      from sklearn.naive_bayes import MultinomialNB
      from sklearn.model_selection import train_test_split
      from sklearn.metrics import accuracy_score

      # to make a simple text classification machine learning
      def simpleTextClassificationMachine( texts, labels ):
        # Vectorize text data
        vectorizer = TfidfVectorizer()
        X = vectorizer.fit_transform( texts )

        # Split data into training and testing sets
        X_train, X_test, y_train, y_test = train_test_split( X, labels, test_size=0.4, random_state=42 )

        # Train a classifier
        classifier = MultinomialNB()
        classifier.fit( X_train, y_train )

        # Make predictions
        y_pred = classifier.predict( X_test )

        # Calculate accuracy
        accuracy = accuracy_score( y_test, y_pred )
        print(f"Accuracy: { accuracy }")
      # end function
```

Digunakan fungsi simpleTextClassificationMachine pada **token judul hasil lemmatization** yang kode dan hasilnya keluaran fungsi adalah sebagai berikut:

```
simpleTextClassificationMachine( df_sentiment['merged_lemma_judul'].copy(), df_sentiment['sentiment_lemma_judul'].copy() )
```

```
Accuracy: 0.9761904761904762
```

Digunakan fungsi simpleTextClassificationMachine pada **token berita hasil lemmatization** yang kode dan hasilnya keluaran fungsi adalah sebagai berikut:

```
[250] simpleTextClassificationMachine( df_sentiment['merged_lemma_berita'].copy(), df_sentiment['sentiment_lemma_berita'].copy() )
```

```
Accuracy: 0.7738095238095238
```

Digunakan fungsi simpleTextClassificationMachine pada **token judul hasil stemming** yang kode dan hasilnya keluaran fungsi adalah sebagai berikut:

```
[251] simpleTextClassificationMachine( df_sentiment['merged_stemming_judul'].copy(), df_sentiment['sentiment_stemming_judul'].copy() )
```

```
Accuracy: 0.9761904761904762
```

Digunakan fungsi simpleTextClassificationMachine pada **token berita hasil stemming** yang kode dan hasilnya keluaran fungsi adalah sebagai berikut:

```
[252] simpleTextClassificationMachine( df_sentiment['merged_stemming_berita'].copy(), df_sentiment['sentiment_stemming_berita'].copy() )
```

```
Accuracy: 0.7380952380952381
```

Digunakan fungsi simpleTextClassificationMachine pada **token judul hasil stopword** yang kode dan hasilnya keluaran fungsi adalah sebagai berikut:

```
simpleTextClassificationMachine( df_sentiment['merged_cleaned_judul'].copy(), df_sentiment['sentiment_cleaned_judul'].copy() )
```

```
Accuracy: 0.9761904761904762
```

Digunakan fungsi simpleTextClassificationMachine pada **token berita hasil stopword** yang kode dan hasilnya keluaran fungsi adalah sebagai berikut:

```
simpleTextClassificationMachine( df_sentiment['merged_cleaned_berita'].copy(), df_sentiment['sentiment_cleaned_berita'].copy() )
```

```
Accuracy: 0.7857142857142857
```