

Visualisasi Data Menggunakan Python

Diajukan untuk melengkapi tugas matakuliah Big Data dari dosen pengajar Dr. Taufik Fuadi
Abidin, S.Si., M.Tech.

Oleh:

BUDI GUNAWAN

1608107010009



**JURUSAN INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS SYIAH KUALA
DARUSSALAM, BANDA ACEH**

DAFTAR ISI

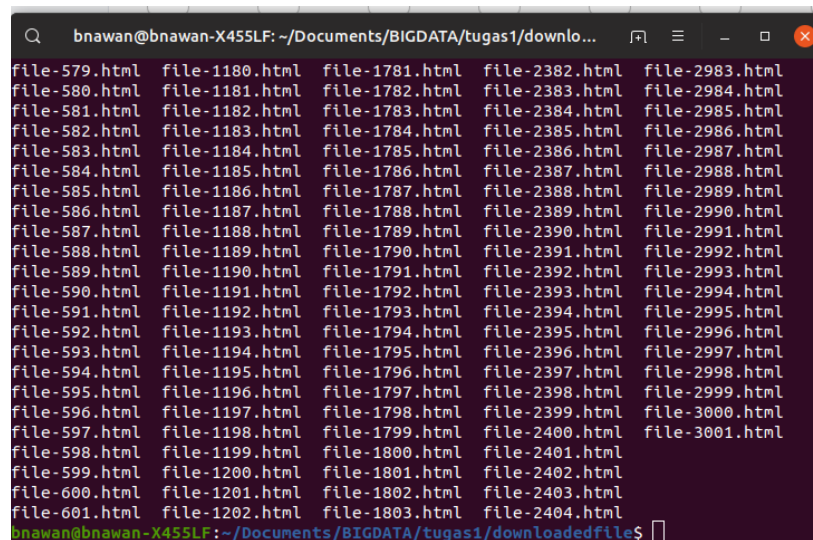
	<i>Halaman</i>
HALAMAN JUDUL	i
DAFTAR ISI	ii
PEMBAHASAN	1
A. Proses Crawling	1
B. Proses Ekstrak Konten	3
C. Bar Chart Plotting	5
D. Line Chart Plotting	6
E. Pie Chart Plotting	6
KESIMPULAN	8
SARAN	8
LAMPIRAN	8

PEMBAHASAN

A. Proses Crawling

Sebelum melakukan visualisasi data, terlebih dahulu kita harus mempersiapkan data berita yang dapat diperoleh dari hasil crawling situs portal berita nasional. Dalam laporan ini, data diambil dari situs detik.com. Total file berjumlah 3001 buah data html. Kode python yang digunakan bernama codeperl.py yang dapat dijalankan dengan

```
python3 codeperl.py <new_file.txt>
```



Gambar 1 : Total data

Sebelum melakukan crawling, carilah terlebih dahulu semua link yang ada pada halaman index portal berita. Karena menggunakan python versi 3, maka digunakanlah library *mechanicalsoup* yang secara teknis memiliki banyak kesamaan dengan mechanize yang tidak dapat dijalankan pada python versi 3.

```
65         halaman+=1
66         browser.open(url)
67         test=browser.links()
68         for link in test:
69             page=link['href']
70             a = re.search("/berita/",page)
71             if (a) and page not in urls:
72                 urls[page]=dateTimeObj
73                 count+=1
74                 print("\rTanggal "+str(tanggal)+" bulan "+str(bulan)+" => Total link(s) : "+str(count))
75                 if count >= 3000:
76                     forcebreak = True
77                     break
78             if forcebreak: break
79             tgl1+=1
80             if forcebreak: break
81             bln1+=1
82             if forcebreak: break
83             thn1+=1
84
85     browser.close()
86     for link in urls.keys():
87         f.write(link+"\n")
88
89
```

Gambar 2 : Penggalan kode crawling

Pada metode mechanicalsoup, terlebih dahulu kita membuat objek yang di sini diberi nama browser. Agar dapat digunakan, kita harus membukanya dengan menjalankan `browser.open(url)`. Setelah itu dapatkan semua link pada halaman tersebut dengan `browser.links()`. Untuk memeriksa satu persatu kode tersebut, maka lakukan foreach untuk setiap link dan karena return dari `browser.links()` masih memiliki atribut html, maka kita dapat mengambil linknya dengan memasukkan key berupa atribut itu sendiri yaitu `link[href]`. Lalu lakukan pengecekan kembali karena berita yang diambil haruslah berita dengan melakukan regex terhadap url yang didapat. Setelah itu, barulah url dapat dimasukkan ke dalam hash yang kemudian akan ditulis ke dalam file txt yang menjadi salah satu input argumen pada konsol untuk menjalankan kode ini.

```
Tanggal 28 bulan 7 => Total link(s) : 2970
Tanggal 28 bulan 7 => Total link(s) : 2971
Tanggal 28 bulan 7 => Total link(s) : 2972
Tanggal 28 bulan 7 => Total link(s) : 2973
Tanggal 28 bulan 7 => Total link(s) : 2974
Tanggal 28 bulan 7 => Total link(s) : 2975
Tanggal 28 bulan 7 => Total link(s) : 2976
Tanggal 28 bulan 7 => Total link(s) : 2977
Tanggal 28 bulan 7 => Total link(s) : 2978
Tanggal 28 bulan 7 => Total link(s) : 2979
Tanggal 28 bulan 7 => Total link(s) : 2980
Tanggal 28 bulan 7 => Total link(s) : 2981
Tanggal 28 bulan 7 => Total link(s) : 2982
Tanggal 28 bulan 7 => Total link(s) : 2983
Tanggal 28 bulan 7 => Total link(s) : 2984
Tanggal 28 bulan 7 => Total link(s) : 2985
Tanggal 28 bulan 7 => Total link(s) : 2986
Tanggal 28 bulan 7 => Total link(s) : 2987
Tanggal 28 bulan 7 => Total link(s) : 2988
Tanggal 28 bulan 7 => Total link(s) : 2989
Tanggal 28 bulan 7 => Total link(s) : 2990
Tanggal 28 bulan 7 => Total link(s) : 2991
Tanggal 28 bulan 7 => Total link(s) : 2992
Tanggal 28 bulan 7 => Total link(s) : 2993
Tanggal 28 bulan 7 => Total link(s) : 2994
Tanggal 28 bulan 7 => Total link(s) : 2995
Tanggal 28 bulan 7 => Total link(s) : 2996
Tanggal 28 bulan 7 => Total link(s) : 2997
Tanggal 28 bulan 7 => Total link(s) : 2998
Tanggal 28 bulan 7 => Total link(s) : 2999
Tanggal 28 bulan 7 => Total link(s) : 3000
Tanggal 28 bulan 7 => Total link(s) : 3001

Berhasil mendapatkan 3001 url

Apakah Anda ingin langsung mendownload halaman dari link yang telah didapat ? Y/N
Y
```

Gambar 3: Penggalan kode crawling

```

download.py > ...
1  #!/usr/bin/python3
2  import wget
3  import os
4  import sys
5
6  output = "/home/bnawan/Documents/BIGDATA/tugas1/downloadedfile/"
7  file= sys.argv[1]
8  # Open file
9  with open (file, "r") as fileHandler:
10     x = 1
11     # Read each line in loop
12     for line in fileHandler:
13         line=line.rstrip("\n")
14         os.system('wget -O '+output+'file-'+str(x)+'.html '+line)
15         x+=1
16

```

Gambar 4: Penggalan kode crawling

Kemudian, setelah semua link terkumpul dalam semua file, lakukanlah crawling di mana kita akan menggunakan download.py yang akan tertriggered setelah menekan input y. Di sini kita akan melakukan wget untuk mengambil data dari server tersebut.

B. Proses Ekstrak Konten

Jika anda telah mencapai tahap ini, berarti anda telah memperoleh data sebanyak 3001 buah. Pada tahap ini, dilakukan pembersihan terhadap tag-tag html yang tidak diperlukan sehingga hanya menyisakan title, tanggal, dan konten teks. Di sini kita akan menggunakan kode extcontent.py dengan menjalankan perintah `python3 extcontent.py`. Data hasil ekstrak konten akan menghasilkan output berekstensi *bersih.dat*

```

file-980.html.bersih.dat file-1981.html.bersih.dat file-2982.html.bersih.dat
file-981.html.bersih.dat file-1982.html.bersih.dat file-2983.html.bersih.dat
file-982.html.bersih.dat file-1983.html.bersih.dat file-2984.html.bersih.dat
file-983.html.bersih.dat file-1984.html.bersih.dat file-2985.html.bersih.dat
file-984.html.bersih.dat file-1985.html.bersih.dat file-2986.html.bersih.dat
file-985.html.bersih.dat file-1986.html.bersih.dat file-2987.html.bersih.dat
file-986.html.bersih.dat file-1987.html.bersih.dat file-2988.html.bersih.dat
file-987.html.bersih.dat file-1988.html.bersih.dat file-2989.html.bersih.dat
file-988.html.bersih.dat file-1989.html.bersih.dat file-2990.html.bersih.dat
file-989.html.bersih.dat file-1990.html.bersih.dat file-2991.html.bersih.dat
file-990.html.bersih.dat file-1991.html.bersih.dat file-2992.html.bersih.dat
file-991.html.bersih.dat file-1992.html.bersih.dat file-2993.html.bersih.dat
file-992.html.bersih.dat file-1993.html.bersih.dat file-2994.html.bersih.dat
file-993.html.bersih.dat file-1994.html.bersih.dat file-2995.html.bersih.dat
file-994.html.bersih.dat file-1995.html.bersih.dat file-2996.html.bersih.dat
file-995.html.bersih.dat file-1996.html.bersih.dat file-2997.html.bersih.dat
file-996.html.bersih.dat file-1997.html.bersih.dat file-2998.html.bersih.dat
file-997.html.bersih.dat file-1998.html.bersih.dat file-2999.html.bersih.dat
file-998.html.bersih.dat file-1999.html.bersih.dat file-3000.html.bersih.dat
file-999.html.bersih.dat file-2000.html.bersih.dat file-3001.html.bersih.dat
file-1000.html.bersih.dat file-2001.html.bersih.dat
file-1001.html.bersih.dat file-2002.html.bersih.dat
bnawan@bnawan-X455LF:~/Documents/BIGDATA/tugas1/cleaned$

```

Gambar 5: Output ekstrak konten

```

try:
    with open (fileout, "w") as fileHandler:
        html=os.popen('cat '+rawdir+'/'+thefile).read()
        ekstraktor=BeautifulSoup(html, 'html.parser')
        title=ekstraktor.title.string
        print(title)
        htmltime=ekstraktor.find(attrs={"name":"publishdate"})
        time=htmltime['content']
        print(time)
        # kill all script and style elements

        content=str(ekstraktor.find_all(id='detikdetailtext'))
        contenttext=BeautifulSoup(content, 'html.parser')

        for script in contenttext(["script", "style"]):
            script.extract()    # rip it out

        detailtag=contenttext.find("div", {'class':'detail_tag'})
        if detailtag:
            detailtag.decompose()

        text=contenttext.get_text()
        # break into lines and remove leading and trailing space on each
        lines = (line.strip() for line in text.splitlines())
        # break multi-headlines into a line each
        chunks = (phrase.strip() for line in lines for phrase in line.split(" "))
        # drop blank lines
        text = '\n'.join(chunk for chunk in chunks if chunk)

        text=re.sub(r'[\!@#$\%]', '', text)
        text=re.sub('\n', '', text)
        print(text)
        fileHandler.write(title+"\n"+time+"\n"+text)

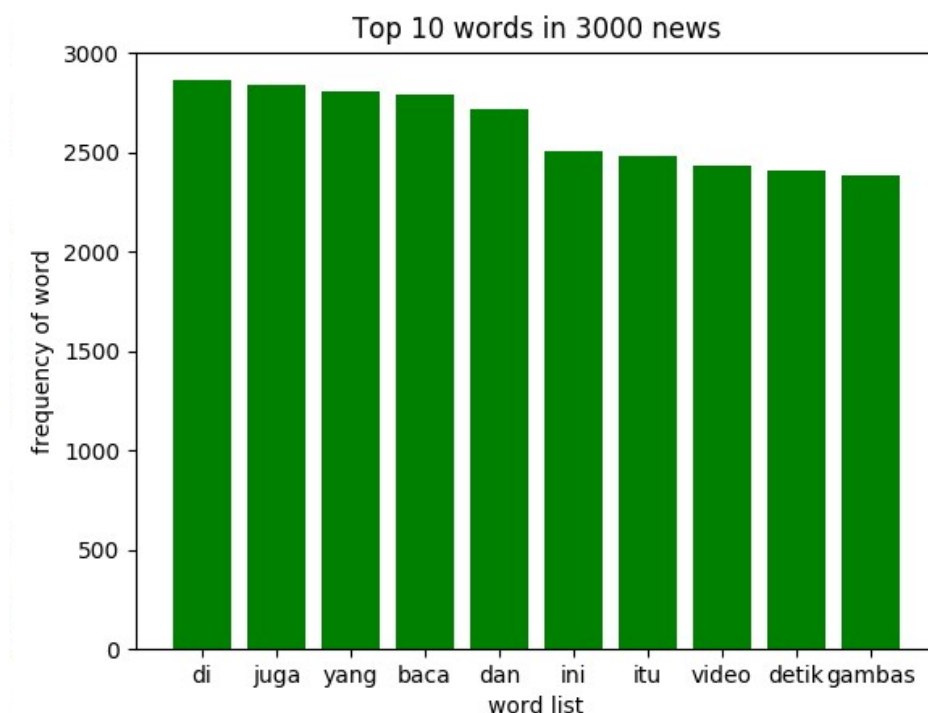
```

Gambar 6 : Penggalan kode ekstrak konten

Di sini kita akan menggunakan library *Beautifulsoup* untuk dapat mengekstrak dan melakukan selection terhadap file html. Seperti sebelumnya, kita akan membuat objek yang diberi nama ekstraktor. Untuk mengambil title, dapat menggunakan fungsi bawaannya dengan memanggil *ekstraktor.title.string* Sedangkan untuk mengambil tanggal, karena pada file html detik.com tanggal berada pada atribut name publishdate, maka lakukan find dengan menggunakan objek ekstraktor dan ambil tanggal tersebut dengan memasukkan key yaitu content pada variabel yang menampung hasil find karena tanggal berada pada atribut content. Terakhir, untuk ekstrak isi beritanya, find all semua teks yang berada pada tag “detikdetailteks” dan gunakan fungsi bawaan *get_teks*. Di sini kita akan menambah objek baru bernama *contenttext* agar lebih mudah dalam menghilangkan atribut ataupun semua yang tidak dibutuhkan dalam teks utama tersebut.

C. Bar Chart Plotting

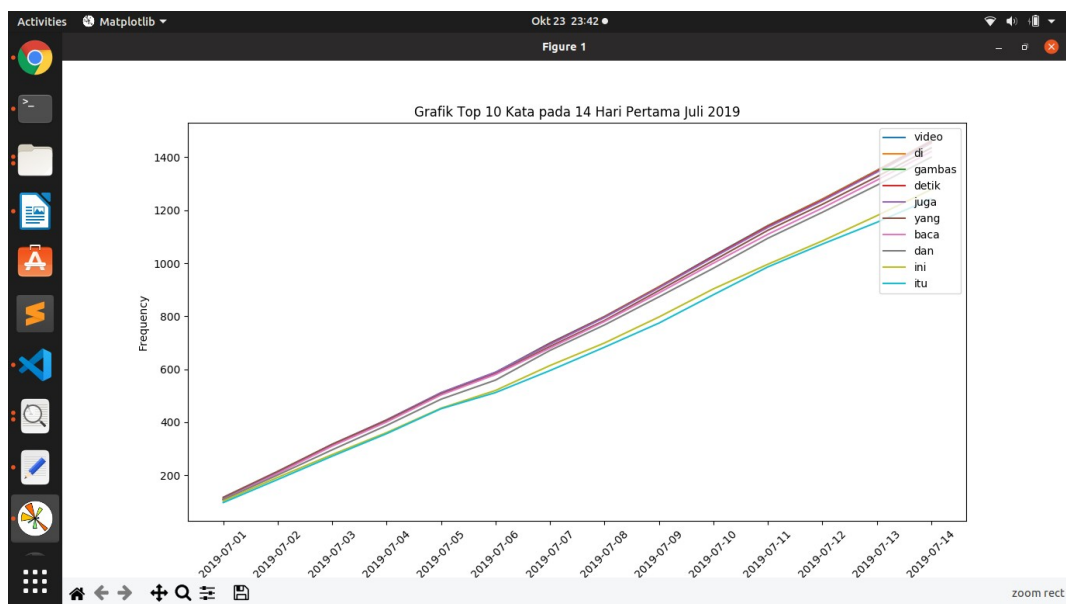
Untuk melakukan plotting 10 kata terbaik dalam 3000 berita tersebut, kode yang akan digunakan adalah `chart10best.py`. Pertama-tama, library yang dibutuhkan adalah library NLTK untuk melakukan pemisahan kata demi kata atau biasa dilakukan untuk ngram dan pembuatan kamus. Sebelum melakukan ngramming, pastikan data telah ditokenize dan dibuat lower semua. Kondisinya di sini, kita tidak menghilangkan *stopword*nya. Setelah itu, barulah dilakukan ngramming. Kata-kata yang didapatkan dimasukkan ke dalam hash dan value yang akan terus diupdate sesuai frekuensinya. Kemudian, lakukan sorting secara descending dan ambillah top 10 tertinggi yang akan dimasukkan ke dalam 2 variabel array karena untuk membuat plotting, dibutuhkan data dalam array untuk melakukan plotting. Lalu, barulah kita akan melakukan plotting dengan menjalankan setiap fungsi bawaan yang terdapat dalam library matplotlib sehingga akan menghasilkan output berupa bar chart histogram.



Gambar 7 : Bar Chart plotting

D. Line Chart Plotting

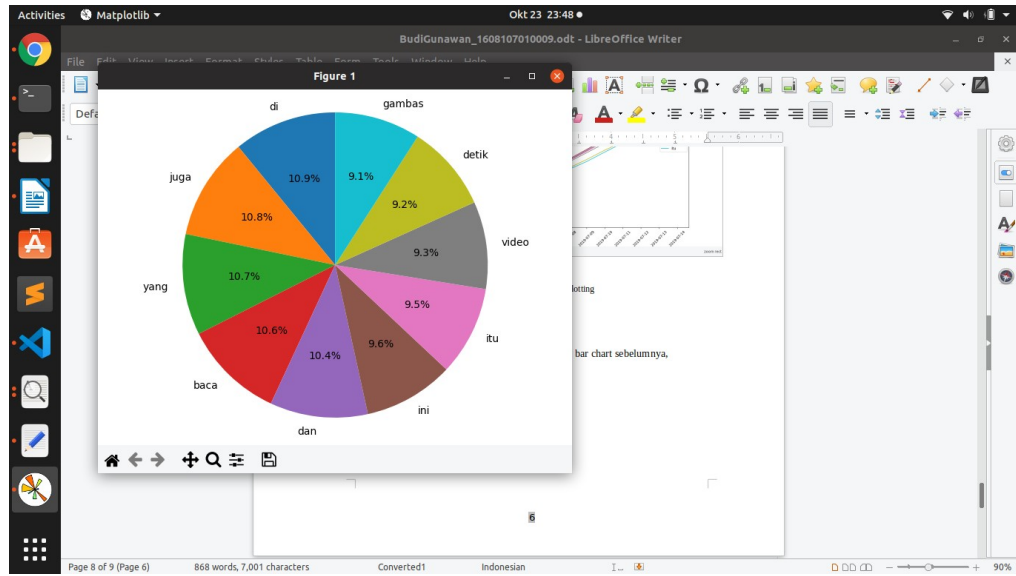
Tahapan ini hampir sama dengan tahapan bar chart plotting, namun karena memiliki ketentuan untuk durasi 14 hari, maka kita akan melakukan pengecekan satu persatu file, apakah tanggal tersebut berkisar dari tanggal 01-07-2019 hingga 14-07-2019. Apabila tanggal sesuai, barulah dilakukan proses ngramming semua dan dimasukkan ke dalam hash untuk dicari top 10 kata terbanyak. Tidak lupa juga, frekuensi tiap kata perhari juga dimasukkan ke dalam hash, sehingga setelah didapatkan top 10 kata terbanyak, kita dapat memeriksa frekuensinya dari hari ke hari selama durasi 14 hari tersebut yang mana hash tersebut merupakan nested hash. Setelah itu, setiap data dimasukkan juga ke dalam array agar dapat dilakukan plotting data. Di sini saya menggunakan kode `chart14days.py`.



Gambar 8 : Line Chart Plotting

E. Pie Chart Plotting

Dengan menggunakan kode yang sama dengan bar chart sebelumnya, yaitu `chart10best.py`, namun pada saat melakukan plotting terdapat fungsi `pie` sehingga menghasilkan hasil berupa chart berbentuk pie. Di sini, tidak lupa value dalam hash sebelumnya dijadikan persentase dengan pembanding yaitu total 10 value tertinggi.



Gambar 9 : Pie Chart Plotting

KESIMPULAN

Adapun kesimpulan dari percobaan ini adalah :

1. Hasil data tertinggi terlalu berdekatan satu sama lain karena tidak menghilangkan stopwords

SARAN

Adapun saran untuk perbaikan dalam project ini adalah agar terlebih dahulu menghilangkan stopwords

LAMPIRAN

Berikut link berisi source code yang telah digunakan :

<https://github.com/budigunawan99/PythonPlotting>