Praktikum 1

Information Retrieval

"Term Frequency Penerapan Zipf's Law"

Prepared By:
Salman El Farisi S.Kom,M.Kom
Elyas Randi Renaldi

Tujuan :

1. Menerapkan tahapan preprocessing teks, meliputi case folding, cleanning, dan tokenization.
2. Menghitung dan menganalisis frekuensi kata (Term Frequency) serta memvisualisasikannya dalam grafik untuk membuktikan pola distribusi Zipf.
3. Memahami konsep dasar Hukum Zipf (Zipf's Law) dalam pemrosesan bahasa alami, khususnya hubungan antara frekuensi kemunculan kata dan peringkatnya dalam sebuah korpus teks.

Aturan Pengerjaan:

1. Gunakan text editor yang nyaman bagi anda
2. Diperkenankan mengerjakan langsung bagi yang sudah memahami dan menguasai materi
3. Dilarang melakukan tindakan plagiarism (asisten lab akan mengecek hasil pekerjaan)
    a. 1x nilai praktikum terkait bernilai 0
    b. 2x nilai matakuliah pemrograman web E
    c. 3x mahasiswa akan di sidang komite etik kampus

A. Pengumpulanan Tugas

1. Dikumpulkan di elena
2. Penamaan file contoh(Pratikum1_Nama_NIM_.ipynb)
3. Dikumpulkannya di elena hanya berupa link dari file colab
4. Paskita link dibagikan dalam keadaan view agar dapat dinilai

B. Dataset

Link dataset:
https://drive.google.com/file/d/1NOpfv7EYR7GNk238Xtv3CcpKH2G4vSdG/view?usp=sharing

C. Linkungan Pengembangan ML – Google Colab

Pada praktikum kali ini akan diperkenalkan lingkungan pengembangan pemodelan machine learning menggunakan platform aplikasi Google Colab. Platform Google Colab memungkinkan Anda menulis dan mengeksekusi Python di browser, dengan tidak memerlukan konfigurasi, akses tanpa biaya ke GPU dan dapat berbagi dengan mudah.

1. Tutorial Penggunaan Google Colab

1. **Membuat Notebook di Colab**
   - Buka: https://colab.research.google.com
   - Pilih New Notebook In Drive → simpan di informationRetrieval/praktikum01Week04/notebooks/.

2. **Menjalankan Cell**
   - Gunakan Code Cell untuk Python, Text Cell (Markdown) untuk catatan.
   - Shortcut: Shift+Enter untuk run cell.

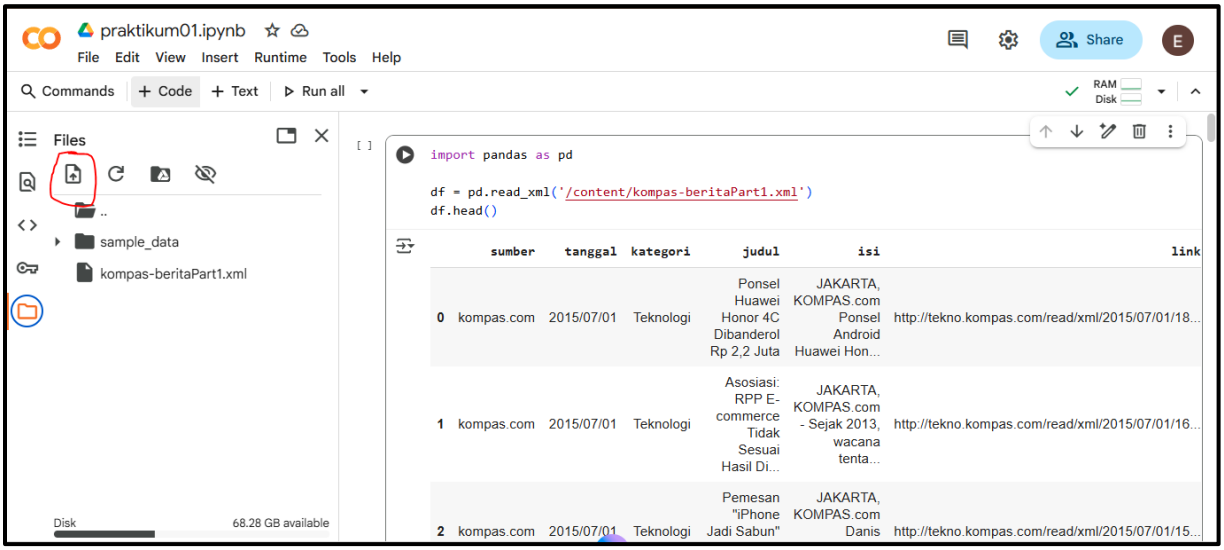
3. **Menghubungkan dengan Google Drive dan Membaca File XML**

   Sel ini berfungsi untuk menghubungkan lingkungan Google Colab dengan akun Google Drive-mu. Setelah kode ini dijalankan, akan muncul tautan otorisasi. Kamu harus mengklik tautan tersebut, memilih akun Google, dan memberikan izin agar Colab bisa mengakses file-file yang tersimpan di Google Drive-mu. Proses ini hanya perlu dilakukan satu kali per sesi.

   Sel ini menggunakan library Pandas untuk membaca file data. Variabel path menyimpan lokasi folder di Google Drive tempat file kompas-beritaPart1.xml berada. Fungsi pd.read_xml() kemudian membaca file tersebut dan menyimpannya ke dalam sebuah DataFrame.

   Menjalankan df.head() di akhir sel akan menampilkan lima baris pertama dan terakhir dari data yaitu 'sumber', 'tanggal', 'kategori', 'judul', 'isi', 'link' dan 'xml'. Ini memberikan gambaran awal tentang struktur data.

Untuk datanya bisa dilakukan via uplouad, namun bila runtime terputus perlu mengupload datanya lagi. Atau bisa disimpan dulu pada google drive untuk mengatasi hal tersebut.



2.1 Eksplorasi Data

1. Mencari Informasi Data

Metode .info() memberikan ringkasan singkat tentang DataFrame. Ini sangat penting untuk langkah awal analisis data karena menampilkan informasi berikut:

1) Jumlah baris dan kolom
2) Nama-nama kolom.
3) Jumlah data non-null (data yang tidak kosong) per kolom.
4) Tipe data (Dtype) dari setiap kolom.

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1502 entries, 0 to 1501
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   sumber      1502 non-null   object
 1   tanggal     1502 non-null   object
 2   kategori    1502 non-null   object
 3   judul       1502 non-null   object
 4   isi         1502 non-null   object
 5   link        1502 non-null   object
 6   jumlahkata  1502 non-null   int64
dtypes: int64(1), object(6)
memory usage: 82.3+ KB
```

2. **Memeriksa nilai NaN dan Duplikat**
   Memastikan bahwa data tidak memiliki nilai NaN dan nilai duplikat. Bila terdapat nilai NaN maka perlu ditangani terlebih dahulu.

```
df.isnull().sum()
```

	0
sumber	0
tanggal	0
kategori	0
judul	0
isi	0
link	0
jumlahkata	0

dtype: int64

```
df.duplicated().sum()
```

```
np.int64(0)
```

2.2 Data Preparation / Data cleaning

1. **Case Folding (mengubah huruf kapital menjadi huruf kecil)**
   Dalam mengubah huruf kapital menjadi huruf kecil, kita bisa menggunakan library yang sudah pada pyhton dengan menggunakan method *lower()* dan disimpan dengan *function* kita buat sendiri dengan nama *caseFolding*.

   Untuk memastikan bahwa *function* dibuat sudah bekerja dengan baik, kita uji coba dengan kalimat yang ada pada variabel contoh.

```
def caseFolding(text):
  text = text.lower()
  return text


contoh = "Terima kasih, Kak! Kamu sangat baik sekali kepadaku hari ini."
print(f'original: {contoh}')
print(f'case folded: {caseFolding(contoh)}')
```

```
original: Terima kasih, Kak! Kamu sangat baik sekali kepadaku hari ini.
case folded: terima kasih, kak! kamu sangat baik sekali kepadaku hari ini.
```

2. Punctuation Removal

Dalam tahap pembersihan data kali ini kita menggunakan library re (*reguler expression)* yang fungsinya untuk mencari dan mencocokkan string dengan pola tertentu, memungkinkan untuk memvalidasi, mencari, mengekstrak, dan mengganti bagian dari teks.

Dalam kode *reguler expression* dibawah berfungsi untuk menghapus karakter selain huruf dan angka. Hasilnya akan menghapus tanda baca dan simbol

```
import re

def punctuationRemoval(text):
  text = re.sub(r'[^\w\s]', '', text)
  return text


contoh = "Terima kasih, Kak! Kamu sangat baik sekali kepadaku hari ini."
print(f'original: {contoh}')
print(f'punctuation removed: {punctuationRemoval(contoh)}')
```

```
original: Terima kasih, Kak! Kamu sangat baik sekali kepadaku hari ini.
punctuation removed: Terima kasih Kak Kamu sangat baik sekali kepadaku hari ini
```

3. Menerapkan Function

Bila *function* yang kita buat sudah bekerja dengan baik, maka tinggal kita terapkan ke dalam dataframe. Kolom **isi** yang akan kita gunakan untuk dilakukan pembersihan lalu hasilnya kita simpan ke kolom yang baru kita buat yaitu **clean.**

```
df['clean'] = df['isi'].apply(caseFolding)
df['clean'] = df['clean'].apply(punctuationRemoval)
df.head()
```

	sumber	tanggal	kategori	judul	isi	link	jumlahkata	clean
0	kompas.com	2015/07/01	Teknologi	Ponsel Huawei Honor 4C Dibanderol Rp 2,2 Juta	JAKARTA, KOMPAS.com Ponsel Android Huawei Hon...	http://tekno.kompas.com/read/xml/2015/07/01/18...	315	jakarta kompascom ponsel android huawei honor...
1	kompas.com	2015/07/01	Teknologi	Asosiasi: RPP E-commerce Tidak Sesuai Hasil Di...	JAKARTA, KOMPAS.com - Sejak 2013, wacana tenta...	http://tekno.kompas.com/read/xml/2015/07/01/16...	419	jakarta kompascom sejak 2013 wacana tentang r...
2	kompas.com	2015/07/01	Teknologi	Pemesan "iPhone Jadi Sabun" Karyawan Pesaing L...	JAKARTA, KOMPAS.com Danis Darusman, pelanggan ...	http://tekno.kompas.com/read/xml/2015/07/01/15...	265	jakarta kompascom danis darusman pelanggan laz...

4. Tokenisasi

4.1 Membuat Function Tokenisasi

Dalam tokenisasi ini menggunakan *method split* yang merupakan original dari python untuk dilakukan pemisahan perkata dengan spasi sebagai acuannya. Setalah itu dapat diuji coba *fuction* tersebut.

```
def tokenize(text):
  return text.split(" ")

contoh = "Terima kasih, Kak! Kamu sangat baik sekali kepadaku hari ini."
print(f'original: {contoh}')

clean = caseFolding(contoh)
clean = punctuationRemoval(clean)
print(f'tokenized: {tokenize(clean)}')
```

```
original: Terima kasih, Kak! Kamu sangat baik sekali kepadaku hari ini.
tokenized: ['terima', 'kasih', 'kak', 'kamu', 'sangat', 'baik', 'sekali', 'kepadaku', 'hari', 'ini']
```

4.2 Menerapkan Tokenisasi

Bila *function* yang kita buat sudah bekerja dengan baik, selanjutnya merapkan ke dalam dataframe. Kolom **clean** yang sudah dilakukan pembersihan digunakan untuk tokenisasi lalu hasilnya disimpan ke dalam kolom yang baru kita buat yaitu **tokens** yang berisi list kata-kata**.**

```
df['tokens'] = df['clean'].apply(tokenize)
df.head()
```

	sumber	tanggal	kategori	judul	isi	link	jumlahkata	clean	tokens
0	kompas.com	2015/07/01	Teknologi	Ponsel Huawei Honor 4C Dibanderol Rp 2,2 Juta	JAKARTA, KOMPAS.com Ponsel Android Huawei Hon...	http://tekno.kompas.com/read/xml/2015/07/01/18...	315	jakarta kompascom ponsel android huawei honor...	[jakarta, kompascom, , ponsel, android, huawei...
1	kompas.com	2015/07/01	Teknologi	Asosiasi: RPP E-commerce Tidak Sesuai Hasil Di...	JAKARTA, KOMPAS.com - Sejak 2013, wacana tenta...	http://tekno.kompas.com/read/xml/2015/07/01/16...	419	jakarta kompascom sejak 2013 wacana tentang r...	[jakarta, kompascom, , sejak, 2013, wacana, te...

2.3 Visualisasi

1. **Frekuensi Kata**

   Untuk menghitung kata diperlukan library collections dan *function* Counter untuk menghitungnya. Sebelum itu diperlu menggabungkan semua kata dari seluruh dokumen (all_tokens), lalu menggunakan dilanjutkan *function* Counter untuk menghitung frekuensi kemunculan setiap kata unik (term_freq), dan akhirnya menampilkan hasilnya dalam bentuk tabel Pandas DataFrame (tf_df) untuk analisis Term Frequency (TF). Agar uratan dari yang terbesar ke yang terkicel *paramater* ascending dibuat *false.*

```
from collections import Counter

all_token = []
for tokens in df['tokens']:
  for token in tokens:
    all_token.append(token)

term_freq = Counter(all_token)
term_freq_df = pd.DataFrame(term_freq.items(), columns=['term', 'freq'])
term_freq_df_sorted = term_freq_df.sort_values(by='freq', ascending=False)
term_freq_df_sorted.head()
```

	term	freq
133	yang	6450
10	di	5824
126	dan	4915
2		4264
35	ini	2744

2. **Rank**

   Untuk mengurutkannya berdasarkan rank, untuk menyimpan hasilnya kita buat kolom baru pada dataframe yaitu Rank. Lalu untuk first berfungsi bila ada jumlah frekuensi yang sama maka nilai rank tetap unique.

```
term_freq_df_sorted['rank'] = term_freq_df_sorted['freq'].rank(method='first', ascending=False).astype(int)
term_freq_df_sorted.head()
```

	term	freq	rank
133	yang	6450	1
10	di	5824	2
126	dan	4915	3
2		4264	4
35	ini	2744	5

3. Mengurutkan Berdasarkan Rank

Untuk mengurutkannya berdasarkan *rank,* untuk menyimpan hasilnya kita buat kolom baru pada dataframe yaitu **Rank.** Lalu untuk *first* berfungsi bila ada jumlah frekuensi yang sama maka nilai tetap unique.

```
tf_df['Rank'] = tf_df['Frequency (ft,d)'].rank(method='first', ascending=False).astype(int)

tf_sorted_rank = tf_df.sort_values(by='Rank', ascending=True)
tf_sorted_rank.head(20)
```

	term	Frequency (ft,d)	Rank
122	yang	6451	1
9	di	5829	2
115	dan	4916	3
31	ini	2749	4
21	dengan	2628	5
26	untuk	2586	6
28	dari	2465	7
16	itu	2437	8
87	akan	2055	9
97	dalam	1750	10

4. Zipf Law

Pertama-tama menghitung nilai k = f × r untuk setiap kata (term) yang di mana f adalah frekuensi kemunculan kata (freq) dan r adalah peringkatnya (rank) berdasarkan urutan frekuensi. Operasi ini digunakan untuk menguji Hukum Zipf, yang menyatakan bahwa hasil perkalian antara frekuensi dan peringkat kata dalam korpus teks cenderung konstan.

```
term_freq_df_sorted['k = f * r'] = term_freq_df_sorted['freq'] * term_freq_df_sorted['rank']
term_freq_df_sorted.head(n=100)
```

	term	freq	rank	k = f * r
133	yang	6450	1	6450
10	di	5824	2	11648
126	dan	4915	3	14745
2		4264	4	17056
35	ini	2744	5	13720
...	...	...	...	...
319	para	360	96	34560
1112	dapat	360	97	34920
10922	republikacoid	352	98	34496
406	lain	350	99	34650
441	jika	345	100	34500

Setaleh itu melakukan visualisasi denggan menggunakan library matplotlib dan dari grafik disebut dapat disimpulkan. Bahwa Grafik menunjukkan kurva yang sangat curam, di mana frekuensi kata turun drastis dari peringkat 1 (kata yang paling sering muncul) hingga mencapai peringkat yang sangat rendah. Hanya sedikit kata teratas (di sumbu r dekat nol) yang memiliki frekuensi kemunculan yang sangat tinggi (mencapai sekitar 6000), sementara mayoritas besar kata memiliki frekuensi yang sangat rendah, hampir mendekati nol.

Dari hal itu maka sesuai dengan pernyataan hukum Zipf's Law :

"frekuensi kata berbanding terbalik dengan peringkatnya"

Grafik terlihat distribusi yang sangat tidak merata (skewed), yang menjadi khas pada data bahasa alami.

```
import matplotlib.pyplot as plt

rank = term_freq_df_sorted['rank']
freq = term_freq_df_sorted['freq']
plt.figure(figsize=(10, 5))
plt.scatter(rank, freq, color='red', s=10)
plt.title("Term Frequency vs Rank")
plt.xlabel("rank")
plt.ylabel("frekuensi (f)")

plt.grid(True)
plt.show()
```