PRAKTIKUM I
DATA ACQUISITION, REPRESENTATION, AND STORAGE

SUB-CAPAIAN PEMBELAJARAN MATA KULIAH:

Mahasiswa mampu menjelaskan (C2), mendiskusikan (A2), dan mempraktekkan (P3) representasi dari image, audio, dan text dengan menggunakan python dengan benar.

POKOK BAHASAN:

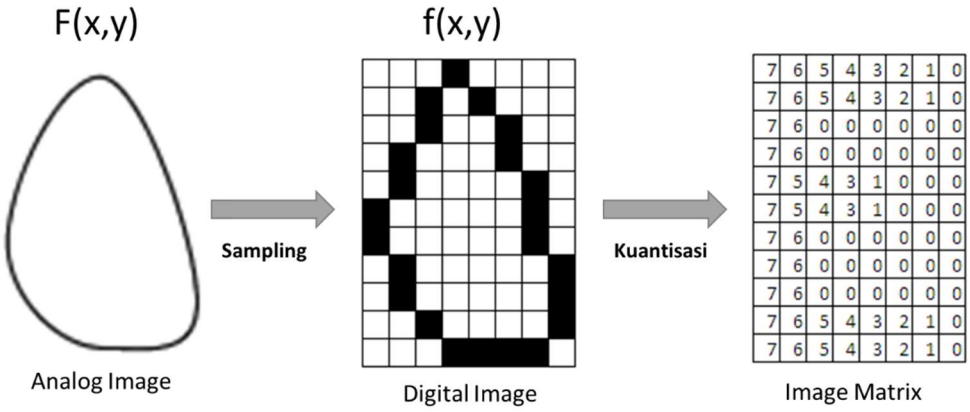
1. Data acquisition, representation, and storage pada Image, Audio, dan Text

1.1. Tujuan Praktikum

Setelah melakukan praktikum, mahasiswa diharapkan untuk dapat:

1. Menjelaskan dan menerapkan akuisisi dan reprsentasi data baik image, audio, dan text dengan menggunakan bahasa pemrograman python
2. Memaksimalkan kreativitasnya dalam menerjemahkan algoritma ke dalam bahasa python dengan memanfaat konsep akuisisi dan representasi data dalam penyelesaian masalah.

1.2. Akuisisi dan Representasi Data Image



Gambar 1. Merubah Citra Analog menjadi Citra Digital

Citra digital dapat diperoleh dari proses digitalisasi citra analog sebagaimana Gambar 1. Ada 2 proses untuk merubah citra analog menjadi citra digital yaitu:

a. Sampling

   Memberi grid pada citra analog, sehingga didapatkan koordinat ruang (x,y). Tiap grid disebut dengan pixel.

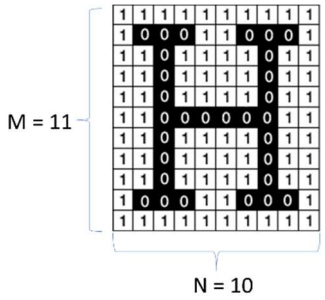
b. Kuantisasi

   Proses memberi nilai pixel berdasarkan intensitas citra/gray level/level keabuan.

   Tingkat keabuan / gray level pixel dalam integer dinyatakan dalam selang [0, L-1]. Proses kuantisasi membagi tingkat keabuan [0, L-1] menjadi G buah level.

$$G = 2^m,\ m = bilangan\ bulat\ positif$$

Ukuran pixel atau resolusi dari citra asli bergantung pada alat yang digunakan untuk mengambil citra. Mengambil citra dengan menggunakan kamera digital tentu saja menghasilkan resolusi yang berbeda jika mengambil citra dengan menggunakan scanner.



Gambar 2. Representasi Citra Digital

Jika kita asumsikan citra digital f(x,y) merupakan citra yang memiliki baris M sebanyak 11, dan kolom N sebanyak 10, sebagaimana Gambar 2, maka nilai dari masing-masing koordinat (x,y) diperoleh dari hasil kuantisasi.

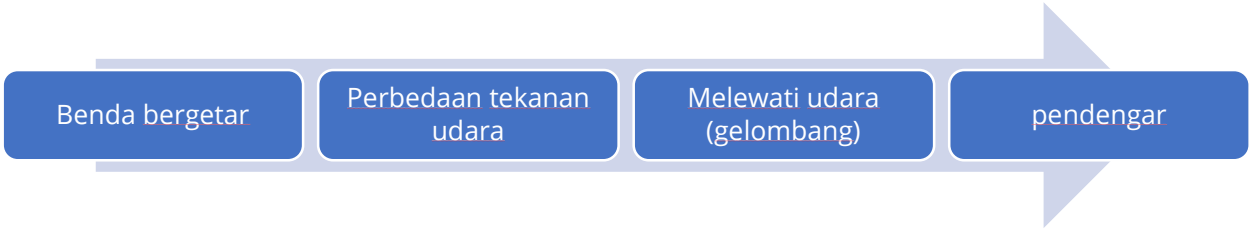
Suatu citra digital berukuran M x N dan setiap pixel memiliki kedalaman b bit, maka kebutuhan memori untuk merepresentasikan citra adalah:

$$M \times N \times b$$

Untuk citra berwarna, 256 level digunakan untuk setiap warna. Citra berwarna terdiri dari 3 kanal (channel) warna: Red, Green, dan Blue. Kombinasi ketiga warna tersebut menghasilkan persepsi warna-warna yang kita lihat.

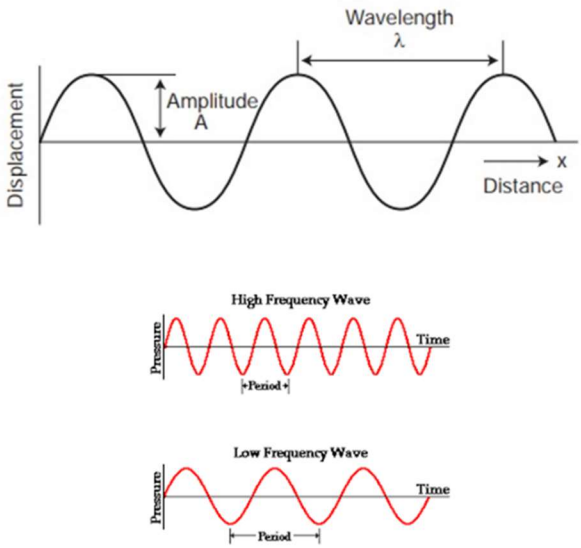
1.3. Akuisisi dan Representasi Data Audio

Bunyi terjadi karena adanya getaran dari suatu objek atau benda. Getaran ini menyebabkan udara di sekitarnya bergetar. Getaran ini sampai ke telinga manusia yang kemudian menyebabkan gendang telinga bergetar. Otak menginterpretasikan getaran ini sebagai bunyi. Bagaimana proses bunyi terdengar oleh pendengar bisa digambarkan melalui siklus pada Gambae 3. Telinga manusia normal mampu mendengar bunyi yang memiliki frekuensi 20 hingga 20.000 Hz.



Gambar 3. Bagaimana Bunyi atau Audio Terdengar

Selama bergetar, perbedaan tekanan terjadi di udara sekitarnya. Pola osilasi yang terjadi dinamakan sebagai “GELOMBANG” (wave) sebagaimana digambarkan pada Gambar 4.



Gambar 4. Gelombang Bunyi

Ada 3 sifat penting gelombang:

a. *Wavelength* / panjang gelombang /periode gelombang

   Jarak antara titik gelombang dan titik ekuivalen pada fasa berikutnya.

b. Amplitudo

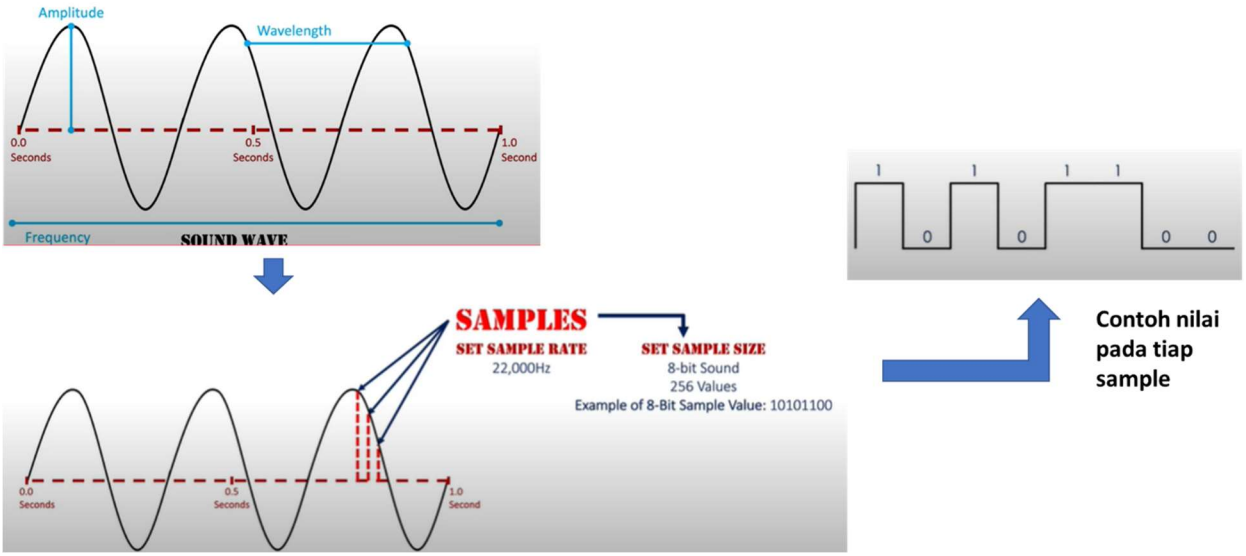
   Tinggi atau rendahnya gelombang. Gelombang yang lebih tinggi diinterpretasikan sebagai volume yang lebih tinggi dan sebaliknya. Satuan dari amplitudo adalah decibel (db).

c. Frekuensi

Jumlah gelombang/getaran yang terjadi dalam 1 detik. Frekuensi diukur dalam Hertz. Semakin banyak getaran, semakin besar frekuensinya dan sebaliknya. Gelombang dengan frekuensi lebih tinggi memiliki pucak yang lebih dekat satu sama lain sehingga gelombang dengan frekuensi yang lebih tinggi memiliki panjang gelombang yang lebih pendek. Frekuensi gelombang suara dikaitkan dengan persepsi kita tentang nada (pitch) suara itu. Gelombang suara berfrekuensi tinggi dianggap sebagai suara bernada tinggi, sedangkan gelombang suara berfrekuensi rendah dianggap sebagai suara bernada rendah.

Gelombang bunyi yang dihasilkan oleh microphone, instrumen musik, dll merupakan gelombang bunyi analog. Gelombang bunyi ini tidak bisa diproses oleh komputer. Agar komputer bisa memprosesnya, gelombang bunyi analog harus diubah menjadi gelombang bunyi digital. Tahap-tahap untuk mengubah gelombang bunyi analog ke gelombang bunyi digital adalah sebagai berikut:

a. Melakukan sample pada sinyal analog dengan cara menentukan sampling rate. Sampling rate adalah berapa kali sampel (irisan) diambil dari gelombang bunyi. Contoh rekomendasi sample rates: 44.100 Hz untuk Suara kualitas CD (dalam 1 detik mengambil 44.100 sample dari gelombang bunyi).
b. Menentukan sample size. Sample size adalah jumlah bits per sampel (banyaknya data direkam pada tiap sampel). Sampling size umunya adalah 8-bit dan 16-bit. Semakin banyak bit, maka semakin banyak suara yang tersedia. 8 bit dapat mewakili kemungkinan 256 nilai.



Gambar 5. Mengubah Sinyal Analog Menjadi Digital

1.4. Akuisisi dan Representasi Data Text

Beberapa representasi teks yang sering digunakan adalah:

a. One-Hot Encoding

Memberikan nilai 0 untuk semua elemen dalam vector kecuali untuk satu elemen, dimana memilki nilai 1. Jumlah array pada One Hot Encoding berdasarkan jumlah kata dalam korpus.

korpus: "I ate an apple"

langkah:

1. Buat index posisi kata dari kalimat yang diberikan, index biasanya urut sesuai dengan abjadnya.

I	ate	an	apple
0	3	1	2

2. Buat Vektor One-Hot Encoding

	0	1	2	3
I	1	0	0	0
ate	0	0	0	1
an	0	1	0	0
apple	0	0	1	0

sehingga untuk keseluruhan korpus → [[1 0 0 0] [0 0 0 1] [0 1 0 0][0 0 1 0]] dengan ukuran 4x4

b. Bag-of-words (BOW) – CountVectorizer

BOW menempatkan kata dalam 'tas' (bag) dan menghitung frekuensi kemunculan setiap kata. Tidak memperhitungkan urutan kata atau informasi leksikal untuk representasi teks.

Contoh:

Data1 = the red dog

Data2 = cat eat dog

Data3 = dog eat food

Data4 = red cat eat

Hasil CountVectorizer:

	the	red	dog	cat	eats	food
Data1	1	1	1	0	0	0
Data2	0	0	1	1	1	0
Data3	0	0	1	0	1	1
Data4	0	1	0	1	1	0

c. TF-IDF

TF-IDF adalah Term Frequency-Inverse Document Frequency. Pada TF-IDF, Bobot yang diberikan untuk setiap kata tidak hanya bergantung pada frekuensi kata, tetapi juga seberapa sering kata tersebut berada di seluruh korpus. Kata-kata yang sering muncul (stopwords) memiliki bobot rendah.

1.5. Percobaan Praktikum Akuisisi dan Representasi Data Image

Untuk lebih memahami materi akuisisi dan representasi data image pada praktikum I ini, mari kita mencoba mempraktikannya.

a. Misalkan kita mempunya citra bernama 'Lenna.png', maka untuk membuka file tersebut maka kita bisa menggunakan library OpenCV CV2, sebagai berikut:

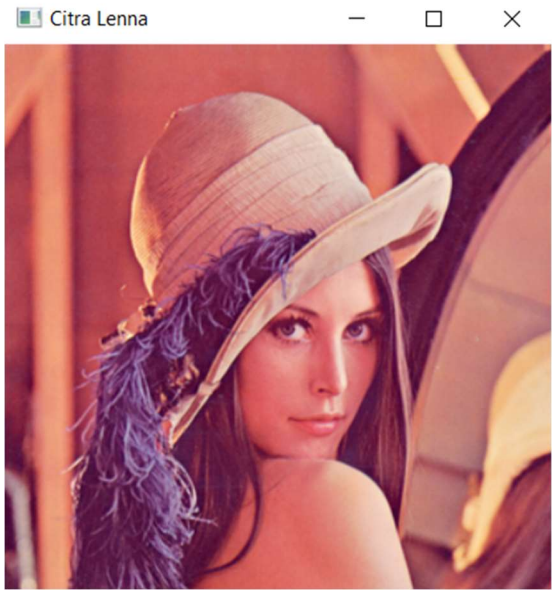
```
#membaca citra
img = cv2.imread("./lenna.png")
```

Beberapa format data yang didukung oleh cv2

Untuk menampilkan citra tersebut, anda bisa mengetikkan kode dibawah ini:

```
cv2.imshow("Citra Lenna", img)
cv2.waitKey(0)
```

Berikut adalah hasilnya:



b. Untuk mengetahui ukuran dan matriks dari citra tersebut gunakan kode dibawah ini.

```
print("Ukuran Citra Warna: ", img.shape)
print("Matriks dari Citra Warna pada baris 0 dan kolom 0: ", img[0,0])
```

Hasilnya adalah sebagai berikut:

```
Ukuran Citra Warna:  (330, 330, 3)
```

Citra tersebut berukuran (330, 330, 3) yang artinya citra terbagi menjadi 330 baris dan 330 kolom, dimana pada tiap baris dan kolom (pixel) terdapat 3 kanal (channel), dengan urutan Blue, Green, Red, dimana masing-masing kanal memiliki nilai tersendiri.

Sedangkan untuk Hasil Nilai Matriks dari Citra Warna adalah sebagai berikut:

```
Matriks dari Citra Warna pada baris 0 dan kolom 0:  [124 137 226]
```

Pada baris ke-0 dan kolom ke-0, Nilai kanal Blue adalah 124, kanal Green adalah 137, dan Kanal Red adalah 226.

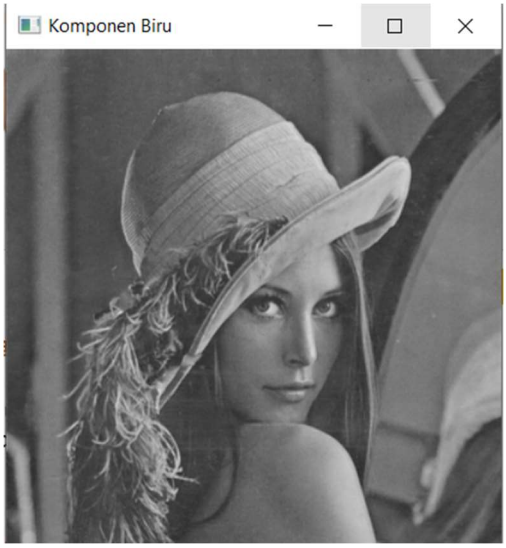
c. Seperti kita tahu, Citra berwarna terdiri dari 3 kanal Red, Green, Blue. Kita bisa memisahkan suatu citra berwarna menjadi komponen Red, Green, dan Blue. Method dalam OpenCV yang dapat digunakan untuk melakukan pemisahan channel adalah `split()`. Untuk memisahkan ketiga channel, tuliskan kode berikut:

```
(blue, green, red) = cv2.split(img)
```

Dengan menggunakan kode diatas, kita telah berhasil memisahkan ketiga channel. Kemudian kita tampilkan channel Blue dengan menuliskan kode berikut:

```
cv2.imshow("Komponen Biru", blue)
```

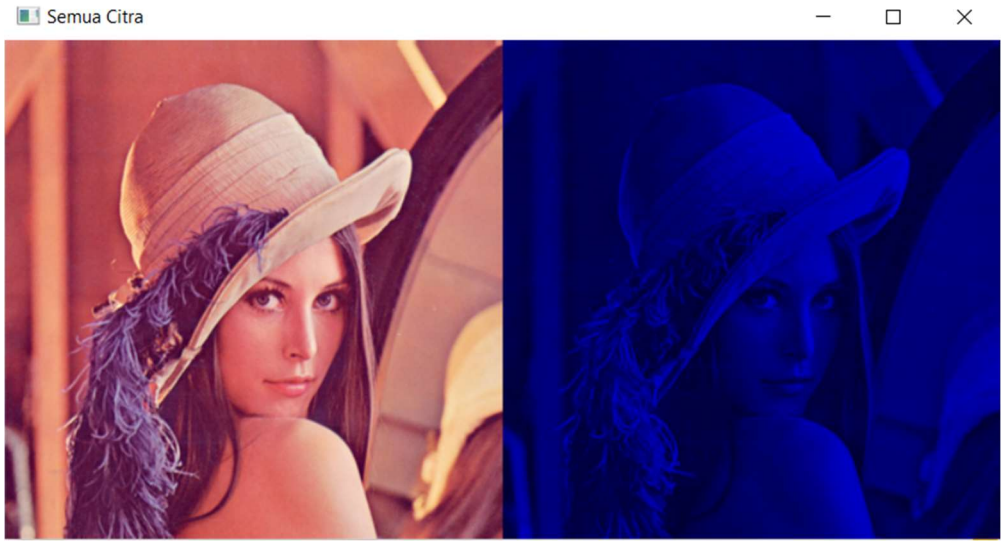
Setelah dipisahkan, komponen Blue akan terlihat seperti gambar dibawah. Karena setiap channel direpresentasikan dalam 1 channel saja, maka tidak terlihat warna Blue dan seperti citra grayscale karena hanya memiliki 1 channel.



Agar channel Blue bisa menampilkan warna sesuai channel-nya maka harus direpresentasikan dalam 3 channel menggunakan method `merge()`. Kita harus membuat matriks berisi nilai '0' yang memiliki ukuran sama dengan ukuran gambar kita (gambar 'lenna.png' memiliki ukuran 330 x 330), kemudian matriks tersebut digunakan untuk mengisi channel Red dan Green (**ingat!** Pada OpenCV, urutan channel adalah Blue, Green, Red). Tuliskan kode berikut:

```
#buat matrix berisi 0 yang berukuran sesuai image asli
zeroMatrix = np.zeros(img.shape[:2], img.dtype)
m = zeroMatrix
blue = cv2.merge([blue, m, m])
```

Output dari kode diatas adalah sebagai berikut:



Tugas Praktikum:

- Modifikasi kode bagian (a) agar dapat menampilkan citra ‘Lenna.png’ dalam grayscale.
- Modifikasi kode bagian (a) agar bisa melakukan crop pada citra ‘Lenna.png’ (terserah dibagian mana saja).
- Modifikasi kode bagian (b) agar dapat menampilkan ukuran citra grayscale dari ‘Lenna.png’ dan nilai matriks dari citra grayscale ‘Lenna.pg’ pada baris ke-0 dan kolom ke-0. Apakah hasilnya berbeda dengan bagian (b), jelaskan alasannya.
- Modifikasi kode bagian (c) untuk menampilkan channel Green dan Red.
- Simpanlah file ‘Lenna.png’ menjadi format JPEG dengan menggunakan method `imwrite` pada OpenCV, apakah terdapat perbedaan nilai array pada file citra asli dan file dengan format JPEG? Jelaskan alasannya.

1.6. Percobaan Praktikum Akuisisi dan Representasi Data Audio

Untuk lebih memahami materi akuisisi dan representasi data audio pada praktikum I ini, mari kita mencoba mempraktikannya.

a. Mencoba membuat sinyal audio berupa gelombang. Sebelum membuat sinyal audio, kita harus menentukan dulu sampling rate, frekuensi, dan panjang. Ketikkan kode berikut untuk membuat sinyal gelombang.

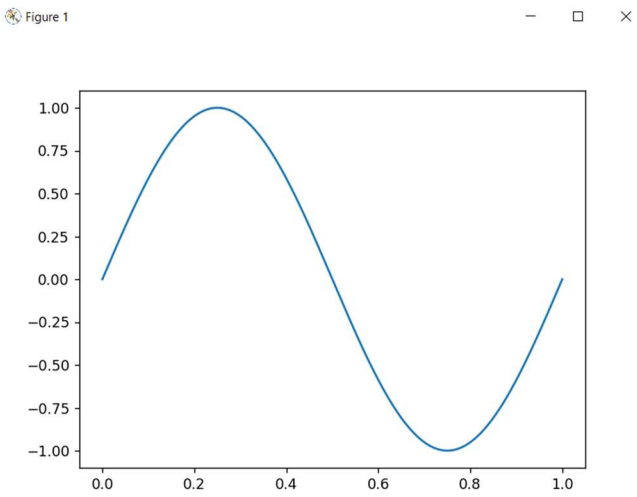
```
# menentukan sampling rate
sr = 44100
# menentukan frequency dalam 1 detik
freq = 1
#menentukan time (sumbu x)
length = 1

#membuat fungsi linear dari 0 s/d time dengan titik berjumlah sr
#1/sr --> stepsize, karena kita mau ada sr titik per detik
t = np.arange(0, length, 1.0/sr)


#membuat wave dengan sin function wave = A*sin(2*pi*freq*t)
signal = np.sin(np.pi*2*freq*t)

#range 1 cycle dalam 1 detik, range 1 sd -1, starting point 0
plt.plot(t, signal)
plt.show()
```

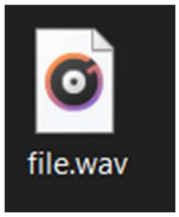
Hasilnya kode diatas adalah gambar gelombang dibawah ini. Gelombang yang dihasilkan memiliki amplitudo antara 1 dan -1, terdapat 1 gelombang dalam 1 detik (frekuensi = 1 Hz).



Lalu simpan sinyal gelombang ke dalam format wav dengan menggunakan kode dibawah ini.

```
wavfile.write("file.wav", sr, signal)
```

Dalam folder kita akan tersimpan file.wav seperti ini:

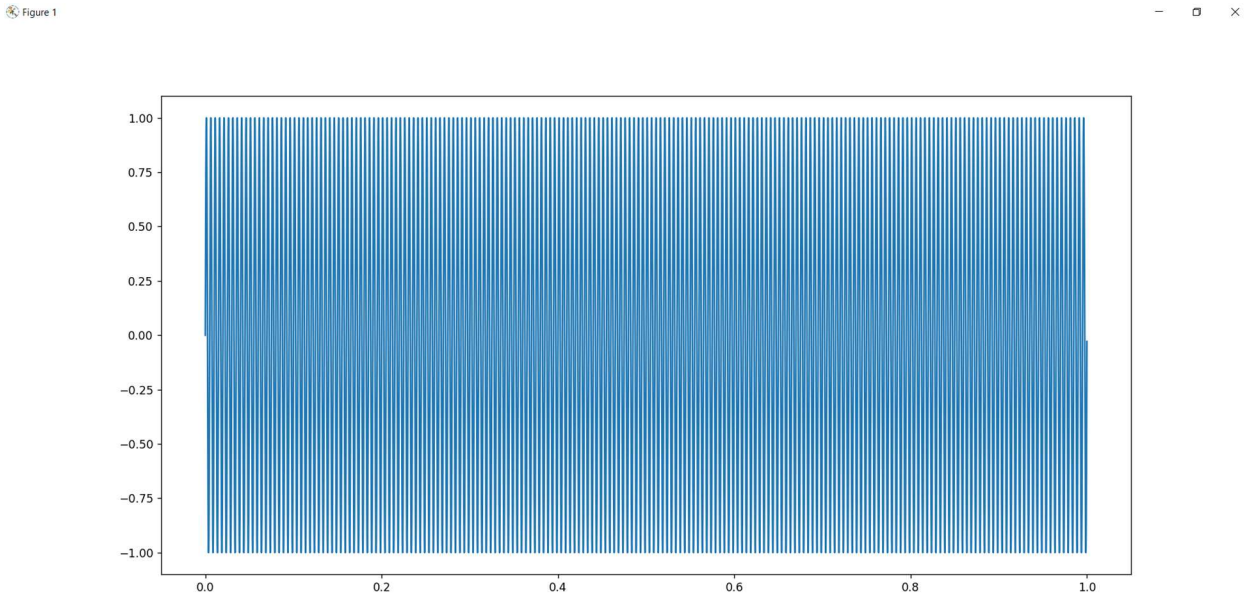


Setelah menyimpan file ke dalam format wav. Coba buka file tersebut dengan menggunakan windows media player atau VLC, apakah terdengar suara?. Ketika anda mencoba mendengar file tersebut tidak ada suara yang muncul karena dalam program kita mengatur frekuensinya adalah 1, padahal telinga manusia hanya bisa mendengar bunyi yang memiliki frekuensi 20 hingga 20.000 Hz.

b. Supaya terdengar manusia, kita harus mengganti frekuensinya. Misalkan, kita mengubah frekuensinya menjadi 200.

```
freq = 200
signal = np.sin(np.pi*2*freq*t)
plt.plot(t, signal)
plt.show()
```

Hasilnya kode diatas adalah gambar gelombang dibawah ini. Gelombang yang dihasilkan memiliki amplitudo antara 1 dan -1, terdapat 200 gelombang dalam 1 detik (frekuensi = 200 Hz).



Lalu simpan sinyal gelombang ke dalam format wav dengan nama file1.wav. Apakah kamu mendengar suara sekarang? Kita dapat mendengar file1.wav karena audio tersebut memiliki frekuensi 200 Hz.

Tugas Praktikum:

- Modifikasi kode bagian (a) agar membuat gelombang suara dengan frekuensi 1000, dan simpan file audio kedalam file2.wav. Apakah suara yang dihasilkan file2.wav berbeda dengan file1.wav (dihasilkan oleh kode (b))? jelaskan alasannya.
- Modifikasi kode bagian (a) dengan mengganti nilai amplitudo menjadi 16, dengan frekuensi 200 Hz, lalu simpan file audio kedalam file3.wav. Apakah terdapat perbedaan bunyi dengan hasil suara dari kode bagian (b) atau file1.wav? jelaskan alasannya.

1.7. Percobaan Praktikum Akuisisi dan Representasi Data Text

Untuk lebih memahami materi akuisisi dan representasi data text pada praktikum I ini, mari kita mencoba mempraktikannya.

a. ASCII

Fungsi ‘ord’ pada python digunakan untuk mengubah karakter menjadi ASCII code, sedangkan fungsi ‘char’ digunakan untuk mengubah ASCII code menjadi karakter.

```
char = 'A'
print(ord(char))
```

Output dari kode diatas adalah **65**

```
ascii = 65
print(chr(ascii))
```

Output dari kode diatas adalah **A**

b. One-Hot Encoding

Kode berikut digunakan untuk melakukan one-hot encoding pada suatu kalimat.

```
from numpy import array
from numpy import argmax
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder

docs = "I ate an apple"

# memisah kalimat menjadi token
split_docs = docs.split(" ")
data = [doc.split(" ") for doc in split_docs]
values = array(data).ravel()

# integer encode
label_encoder = LabelEncoder()
integer_encoded = label_encoder.fit_transform(values)
print(integer_encoded)

# binary encode
onehot_encoder = OneHotEncoder(sparse=False)
integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
onehot_encoded = onehot_encoder.fit_transform(integer_encoded)
print(onehot_encoded)
```

Berikut index dari tiap kata dalam kalimat:

```
[0 3 1 2]
```

Yang berarti:

I	ate	an	apple
0	3	1	2

Dan berikut adalah hasil one-hot encoding untuk kalimat “I ate an apple”

```
[[1. 0. 0. 0.]
 [0. 0. 0. 1.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]]
```

Sehingga [[1 0 0 0] [0 0 0 1] [0 1 0 0] [0 0 1 0]] merupakan one-hot encoding dari masing-masing kata “I ate an apple”.

c. CountVectorizer

Kode berikut digunakan untuk menghitung CountVectorizer pada suatu korpus.

```python
from sklearn.feature_extraction.text import CountVectorizer

text = ["everybody love nlp", "nlp is so cool",
"nlp is all about helping machines process language",
"this tutorial is on basic nlp technique"]

vectorizer = CountVectorizer()

# tokenisasi dan membuat vocab
vectorizer.fit(text)
print(vectorizer.vocabulary_)

# encode dokumen
vector = vectorizer.transform(text)

# hasil encode vektor
print(vector.shape)
print(vector.toarray())
```

Daftar vocabulary atau kata dari dokumen adalah sebagai berikut:

```
{'everybody': 4, 'love': 8, 'nlp': 10, 'is': 6, 'so': 13, 'cool': 3,
'all': 1, 'about': 0, 'helping': 5, 'machines': 9, 'process': 12,
'language': 7, 'this': 15, 'tutorial': 16, 'on': 11, 'basic': 2,
'technique': 14}
```

Ukuran vektor adalah **(4,16)**, karena dalam korpus terdapat **4** dokumen, dimana terdapat **16** kata unik dalam korpus tersebut.

Sehingga, CountVectorizer output pada korpus tersebut adalah sebagai berikut:

```
[[0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0]
 [0 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0]
 [1 1 0 0 0 1 1 1 0 1 1 0 1 0 0 0 0]
 [0 0 1 0 0 0 1 0 0 0 1 1 0 0 1 1 1]]
```

d. TF-IDF

Kode berikut digunakan untuk menghitung TF-IDF pada suatu korpus.

```
from sklearn.feature_extraction.text import TfidfVectorizer
text1 = ['i love nlp', "nlp is so cool",
"nlp is all about helping machines process language",
"this tutorial is on basic nlp technique"]

tf = TfidfVectorizer()
txt_fitted = tf.fit(text1)
txt_transformed = txt_fitted.transform(text1)

idf = tf.idf_
print(dict(zip(txt_fitted.get_feature_names(), idf)))
```

Hasilnya adalah sebagai berikut:

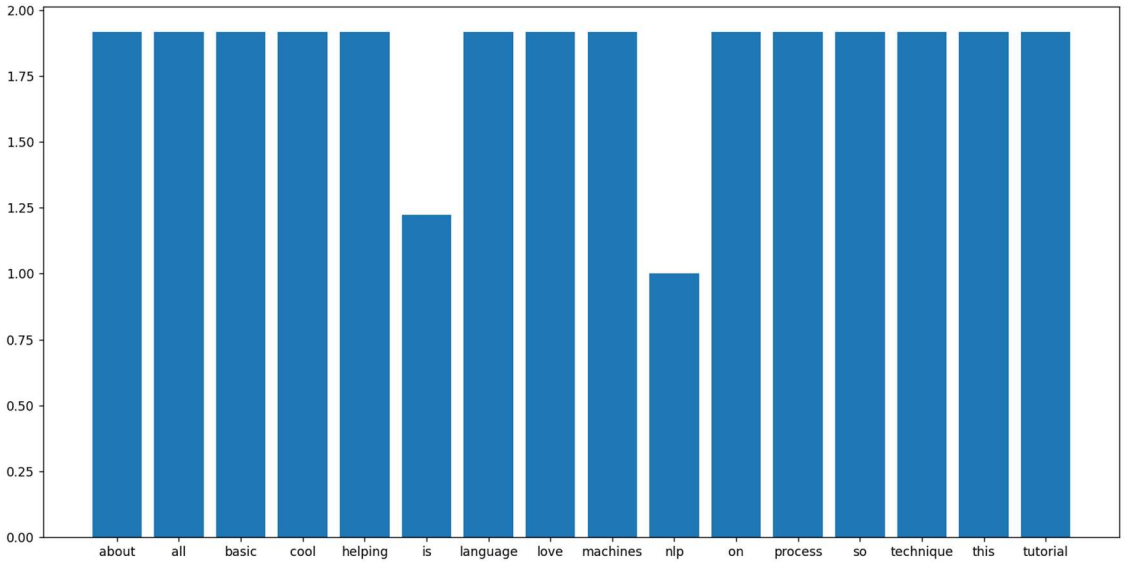
```
{'about': 1.916290731874155, 'all': 1.916290731874155, 'basic':
1.916290731874155, 'cool': 1.916290731874155, 'helping':
1.916290731874155, 'is': 1.2231435513142097, 'language':
1.916290731874155, 'love': 1.916290731874155, 'machines':
1.916290731874155, 'nlp': 1.0, 'on': 1.916290731874155, 'process':
1.916290731874155, 'so': 1.916290731874155, 'technique':
1.916290731874155, 'this': 1.916290731874155, 'tutorial':
1.916290731874155}
```

Kata ‘nlp’ memiliki nilai paling kecil karena kata tersebut merupakan kata yang paling sering muncul dalam korpus, terdapat kata ‘nlp’ pada tiap kalimat. Diikuti dengan stopword ‘is’ dengan nilai `1.2231435513142097`, karena kata ‘is’ muncul di tiga kalimat dalam korpus.

Tugas Praktikum:

- Modifikasi kode bagian (a) agar bisa menampilkan ASCII code untuk kata ‘datA mining’
- Tambahkan kode bagian (b) agar bisa menampilkan kembali kata pertama yang di lakukan one-hot encoding (hint: lakukan inverse dari hasil encode).

  [1 0 0 0] [0 0 0 1] [0 1 0 0] [0 0 1 0]] → yang dibold merupakan hasil one-hot encoding dari kata ‘**I**’, sehingga output dari program nanti adalah **‘I’**
- Download file tugas_text_representation.csv dari hebat e learning, kemudian lakukan CountVectorizer dan TF-IDF pada korpus tersebut. Jelaskan hasil yang didapatkan.
- Modifikasi kode bagian (d) agar bisa menampilkan grafik dari tiap kata. contohnya seperti gambar di bawah.


2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
about
all
basic
cool
helping
is
language
love
machines
nlp
on
process
so
technique
this
tutorial