

DTSense AI & Data Science

Data Science II Academy
Artificial Intelligence and Machine Learning



Data Science II Academy

Week 4

Module 1:

Natural Language Processing

Module 2:

Image Processing

Alvin Rindra Fazrie, M.Sc.

I Nyoman Prayana Trisna, M.Cs.



Germany, Indonesia



info@dt-sense.com



+49 178 1980320



© 2020 DTsense. Belajar
Artificial Intelligence & Data
Science

Table of Contents

DATA SCIENCE II ACADEMY SYLLABUS.....	2
1) TEXT MINING	5
2) REGULAR EXPRESSION	5
3) NLTK	9
4) COMMON TEXT MINING PREPROCESSING	10
I) TOKENIZATION	10
II) CASE FOLDING.....	11
III) STOPWORD REMOVAL	11
IV) NON-ASCII CHARACTERS REMOVAL.....	12
V) PUNCTUATION REMOVAL	13
VI) DIGIT REMOVAL.....	13
VII) STEMMING AND LEMMATIZATION.....	14
5) DOCUMENT VECTOR AND BAG-OF-WORDS.....	15
6) TRAINING TEXT CLASSIFIER MODEL.....	19
7) IMAGE PROCESSING.....	21
I) IMAGE.....	22
II) SCIKIT-IMAGE.....	23
III) NUMPY FOR IMAGES.	26
8) IMAGE PROCESSING TECHNIQUES.....	26
I) THRESHOLDING.....	26
II) FILTERING.....	27
III) CONTRAST ENHANCEMENT.....	30
IV) TRANSFORMATION.....	31
V) EDGE DETECTION.....	32
VI) CORNER DETECTION.....	33
VII) FACE DETECTION.....	34
9) REFERENCES.....	35

Data Science II Academy Syllabus

	Artificial Intelligence & Machine Learning
Day 1	<p>Introduction to the program structures</p> <p>Introduction to Data Science: Understanding Data Science Concept</p> <p>Installation:</p> <ul style="list-style-type: none"> - Python - Integrated Development Environment (eg: Jupyter Notebook) <p>Data Analytics with Python</p> <p>Brief introduction & recap of data structure libraries for data analytics:</p> <ul style="list-style-type: none"> - NumPy - Pandas <p>Exploratory Data Analysis</p> <ul style="list-style-type: none"> - Non-graphical univariate - Graphical univariate - Non-graphical multivariate - Graphical multivariate
Day 2	<p>Introduction to Artificial Intelligence and Machine Learning</p> <p>Introduction to AI:</p> <ul style="list-style-type: none"> - What is AI? - AI Application in real life - Machine Learning as part of AI <p>Introduction to Machine Learning:</p> <ul style="list-style-type: none"> - Machine Learning Vs Conventional AI agent. - Machine Learning types & their terminology. <p>Get to know with Machine Learning Libraries</p> <ul style="list-style-type: none"> - Sklearn - Scipy
Day 3	<p>Machine Learning: Supervised Learning (Part I)</p> <p>Model Selection for Supervised Learning</p> <ul style="list-style-type: none"> - Holdout - Cross-validation <p>Feature Selection</p> <ul style="list-style-type: none"> - Filter-based method - Wrapper-based method

	<p>Classification: Probabilistic Based</p> <ul style="list-style-type: none"> - Naïve Bayes Classifier <p>Classification: Tree based</p> <ul style="list-style-type: none"> - Decision Tree Classifier - Random Forest Classifier
Day 4	<p>Machine Learning: Supervised Learning (Part II)</p> <p>Classification: Vector Space Based</p> <ul style="list-style-type: none"> - K-NN Classifier - SVM + Kernel Function <p>Classification: Neural Network Based</p> <ul style="list-style-type: none"> - MLP + Activation Function <p>Metric Evaluation for Classification</p> <ul style="list-style-type: none"> - Confusion Matrix and its terminology - Overall Performance - Class-based Performance
Day 5	<p>Machine Learning: Supervised Learning (Part III)</p> <p>Regression</p> <ul style="list-style-type: none"> - Linear Regression - SVM Regression - MLP (Neural Based Model) <p>Metric Evaluation for Regression</p> <ul style="list-style-type: none"> - Absolute Error - Squared Error
Day 6	<p>Machine Learning: Unsupervised Learning</p> <p>Introduction to Unsupervised Learning</p> <ul style="list-style-type: none"> - Its distinctions from supervised learning - When to implement unsupervised learning? <p>Clustering</p> <ul style="list-style-type: none"> - Partition-based clustering - Hierarchical-based clustering - Evaluation for Clustering: Silhouette Score <p>Association Rules</p> <ul style="list-style-type: none"> - The rules measurements: support, confidence, lift, leverage, conviction - Apriori method <p>Dimensionality Reduction</p>

Day 7	<h3>Natural Language Processing</h3> <p>Common NLP Preprocessing</p> <ul style="list-style-type: none"> - Case folding - Stopword removal - Stemming / Lemmatization - Non-ASCII removal - Punctuation Removal <p>Vectorizer & Bag of Words</p> <ul style="list-style-type: none"> - Concept in bag of word vector - Term Frequency - Inverse Document Frequency <p>Building NLP Model</p>
Day 8	<h3>Image Processing with Scikit-Image</h3> <p>Introduction to Image Processing with Scikit-Image</p> <ul style="list-style-type: none"> - Identifying image colors: RGB or Gray Scale - Flipping out the images - Thresholding <p>Images Filters & Transformation</p> <ul style="list-style-type: none"> - Filtering images to reduce noises. - Transforming images: aliasing, rotating, rescaling <p>Images & Face Detections</p> <ul style="list-style-type: none"> - Identifying image's edges & perspectives - Face detection: is someone there? <p>Wrap up!</p> <p>Evaluations</p> <p>Discussion to the final project</p>

Natural Language Processing

1) Text Mining

Text mining adalah salah satu *task* dalam AI yang sejatinya bertujuan untuk mengambil data maupun informasi dari suatu naskah (teks). Data atau informasi yang diekstrak dapat berupa potongan informasi seperti kategori teks, atau informasi dalam bentuk utuh seperti ringkasan berita. Pada pertemuan kali ini, kita membahas tentang *text mining* sebagai aplikasi dari *supervised learning*, utamanya adalah klasifikasi.

Dengan klasifikasi, informasi yang dapat diambil berupa kategori dari suatu naskah. Beberapa *task* dalam *text mining* yang masuk sebagai klasifikasi adalah sebagai berikut:

- Analisis sentimen, yang menandakan apakah suatu teks berisi komentar positif, negatif, atau netral terhadap suatu fokus topik
- Deteksi spam, yang biasa digunakan dalam memfilter pesan elektronik (baik SMS atau email)
- Klasifikasi berita, untuk membedakan berita yang diliput berdasarkan kategori berita
- Deteksi suatu *tweet* apakah spam atau bukan
- Deteksi hoax dari suatu berita

Pada pertemuan kali ini, kita akan mencoba untuk melakukan analisis sentimen dari suatu teks. Sebelumnya, kita akan membahas tentang ekstraksi *string* dengan *regular expression*. Kita juga akan berkenalan terlebih dahulu dengan salah satu *library* Python yang paling sering digunakan dalam *text mining*, yaitu NLTK.

2) Regular Expression

Regular expression (atau *regex*) adalah urutan karakter khusus tertentu yang membantu dalam pencocokan suatu *string* di dalam *string*. *Regex* menggunakan sintaks khusus yang diatur dalam pola *regex*. Dalam Python, *regex* disimpan dalam *library* *re*.

```
1 import re
```

Dalam *regex*, semua karakter kecuali *control character* (+ ? . * ^ \$ () [] { } | \) merepresentasikan dirinya sendiri. Artinya, apabila dalam pola *regex* terdapat karakter

huruf A, maka karakter tersebut merepresentasikan karakter A dalam *string*. Apabila *control character* ingin digunakan untuk merepresentasikan diri sendiri dalam pola *regex*, maka dapat menggunakan garis miring (\).

```

1 text = 'my name is alfa and your name is too'
2 pattern = r'name'
3 obj = re.findall(pattern, text)
4 obj

['name', 'name']

```

Dalam gambar di atas, *library re* digunakan sebagai *library* untuk *regex*. Method *re.findall* digunakan untuk mendapatkan semua *string* dalam *string* yang *match* dengan pola yang diberikan. Untuk pola, sebelum pemberian tanda petik, maka diberikan karakter *r* terlebih dahulu seperti contoh di atas. Hasil *return* dari *method* ini adalah *list* yang berisi *string* yang sesuai. Pada contoh di atas, terdapat dua *string* yang cocok dengan pola yang diberikan.

Pembentukan pola tidak hanya menggunakan karakter biasa, namun juga *control character*. Tabel di bawah ini menjelaskan kegunaan dari tiap *control character*:

Pola	Kegunaan
^	Merepresentasikan awal dari baris <i>string</i>
\$	Merepresentasikan akhir dari baris <i>string</i>
.	Merepresentasikan semua karakter (kecuali <i>newline</i>)
(...)	Menggabungkan semua karakter dalam kurung sehingga seakan bertindak sebagai satu karakter
[...]	Merepresentasikan karakter tunggal apapun yang berada dalam kurung
[^...]	Merepresentasikan karakter tunggal apapun yang tidak berada dalam kurung
X*	Merepresentasikan 0 atau lebih kemunculan karakter X
X+	Merepresentasikan 1 atau lebih kemunculan karakter X

X?	Merepresentasikan 0 atau 1 kemunculan karakter X
X{3}	Merepresentasikan 3 kemunculan karakter X
X{3,}	Merepresentasikan 3 atau lebih kemunculan karakter X
X{3,5}	Merepresentasikan 3 sampai 5 kemunculan karakter X
X Y	Merepresentasikan karakter X atau Y
\w	Merepresentasikan karakter yang berupa huruf
\W	Merepresentasikan karakter yang bukan huruf
\d	Merepresentasikan karakter yang berupa angka
\D	Merepresentasikan karakter yang bukan angka
\s	Merepresentasikan karakter yang berupa spasi
\S	Merepresentasikan karakter yang bukan spasi
\n	Merepresentasikan <i>newline</i>
\t	Merepresentasikan <i>tab</i>

Berikut adalah beberapa contoh penggunaan *regex* dalam mengambil suatu pola.

- Contoh di bawah ini adalah pola yang mengambil kata “Python” maupun “python”

```

1 text = 'Python python Lython'
2 pattern = r'[Pp]ython'
3 obj = re.findall(pattern, text)
4 obj

['Python', 'python']

```

- Contoh di bawah ini adalah pola yang mengambil 2 buah digit, yang diikuti “a” kemudian diikuti dua buah huruf

```

1 text = '88abc 29acp 9axxA'
2 pattern = r'\d\da\w\w'
3 obj = re.findall(pattern, text)
4 obj

['88abc', '29acp']

```

- Contoh di bawah ini adalah mencari satu karakter apapun yang diikuti “ma” dan diikuti karakter apapun. Perlu dicatat pada *array* kedua di mana “8ma” juga diikutkan, karena spasi dianggap sebagai karakter.

```

1 text = 'dimas 8ma *man 00mas'
2 pattern = r'.ma.'
3 obj = re.findall(pattern, text)
4 obj

['imas', '8ma ', '*man', '0mas']

```

Untuk *control character* yang berupa perulangan (yaitu * + dan ?), pencarian dilakukan secara menyeluruh. Artinya, seluruh pola dicocokkan sampai semua karakter dalam *string*. Hal ini disebut sebagai *greedy repetition*. Di bawah ini adalah contoh *greedy repetition*.

```

1 text = 'AkuAlaku'
2 pattern = r'A.+u'
3 obj = re.findall(pattern, text)
4 obj

['AkuAlaku']

```

Pada contoh di atas, polanya adalah mencari *string* yang diawali “A” dan diakhiri “u” dengan 1 atau lebih karakter apapun di tengahnya. Padahal, “Aku” ataupun “Alaku” juga memenuhi pola di atas. Hal ini terjadi karena *greedy repetition* dari *regex* yang menyebabkan “u” yang memenuhi adalah “u” yang paling terakhir. Apabila ingin berupa *non-greedy repetition*, maka akan mendapatkan “u” yang paling pertama. Hal ini dilakukan dengan menambahkan “?” setelah *control character* perulangan (yaitu * + dan ?). Berbeda dengan sebelumnya, di mana hasilnya hanya satu, dengan *non-greedy repetition* kita mendapatkan dua *string*. Hal ini karena memang ada dua *string* yang *non-greedy* dan memenuhi pola tersebut.

```

1 text = 'AkuAlaku'
2 pattern = r'A.+?u'
3 obj = re.findall(pattern, text)
4 obj

['Aku', 'Alaku']

```

Apabila sebuah pola *regex* mengandung tanda kurung, baik buka "(" atau tutup ")" maka hasil *return*-nya adalah sebuah *array of tuple*, di mana *tuple*-nya berisi *string* yang cocok dengan pola di dalam tiap tanda kurung. Hal ini seperti pada contoh di bawah, di mana kita ingin mencari kata ganti kepemilikan serta namanya.

```
1 text = "My name is Alfa. Your name is Beta. Name is good"
2 pattern = r"(.*?) name is (.*)\.\s?"
3 obj = re.findall(pattern, text)
4 obj

[('My', 'Alfa'), ('Your', 'Beta')]
```

Perlu dicatat bahwa pola di sini menggunakan *non-greedy repetition* sehingga bukan “name is” yang terakhir digunakan, tetapi yang terdekat. Contoh lainnya dengan kasus yang sama diperlihatkan pada gambar di bawah ini, di mana format dua kalimatnya dibuat berbeda.

```
1 text = "My name is Alfa. Your name's Beta. Name is good"
2 pattern = r"([Mm]y|[Yy]our) name('s| is) (.*)\."
3 obj = re.findall(pattern, text)
4 obj

[('My', ' is', 'Alfa'), ('Your', "'s", 'Beta')]
```

Pada gambar di atas, dapat dilihat *string* kedua pada tiap *tuple* bukanlah *string* yang dicari. Hal ini karena baik “s” ataupun “ is” adalah opsional (salah satu bisa dipilih), sehingga harus dibungkus dengan tanda kurung. Untuk mencari hasil seperti pada contoh pertama, kita dapat melakukannya dengan iterasi.

3) NLTK

NLTK adalah salah satu library dalam Python. NLTK adalah singkatan dari *Natural Language Toolkits*. NLTK berisi banyak fitur yang sangat bermanfaat dalam proses *text mining*. NLTK sendiri tidak secara *default* berada dalam Python, sehingga harus di-*install* terlebih dahulu melalui *pip*. Setelah itu, belum semua *package* dalam NLTK langsung ada, sehingga harus di-*download* terlebih dahulu seperti cara di bawah ini. Pada gambar di bawah, *package* bernama *punkt* adalah *package* yang di-*download*.

```
1 import nltk
2 nltk.download('punkt')
```

Beberapa *method* dalam NLTK yang berguna dalam proses *text mining* adalah sebagai berikut:

- Tokenisasi, yaitu proses memecah suatu *string* menjadi bentuk yang lebih kecil. *Word tokenize* berarti memecah *string* menjadi kumpulan kata, sedangkan *sentence tokenize* memecah *string* ke dalam kalimat-kalimat

- *Stemming* dan *lemmatization*, yaitu mengubah kata ke dalam bentuk dasarnya.
- *Regex parser*, yaitu mengekstrak suatu *string* dari pola *regex* yang diberikan.
- *Part of speech tagging*, yaitu melakukan pelabelan terhadap kata berdasarkan fungsi kata dalam kalimat (seperti kata kerja, kata benda, dan lain sebagainya)

Selain itu, NLTK juga menyediakan resource lainnya seperti *dataset*, kamus, maupun daftar kata-kata *stopword*. Namun, NLTK sendiri sebagian besar *support* ke Bahasa Inggris, dengan *support* ke Bahasa Indonesia yang minim.

4) Common Text Mining Preprocessing

Dalam kasus *text mining*, banyak proses yang harus dilakukan sebelum siap digunakan dalam model (baik *supervised* maupun *unsupervised learning*). NLTK, *sklearn*, maupun fungsi bawaan *string* dalam Python sendiri mendukung terhadap proses-proses ini. Adapun masing-masing preprocessing yang umum digunakan dijabarkan pada subbab selanjutnya. Perlu dicatat bahwa tidak semua proses ini harus dilakukan dalam *text mining*.

I. Tokenization

Tokenisasi adalah salah satu proses yang hampir pasti dilakukan dalam *text mining*. Seperti yang dijelaskan sebelumnya, tokenisasi adalah memecah sebuah teks menjadi bentuk yang lebih kecil, baik berupa kalimat ataupun kata.

Proses tokenisasi sendiri terdapat dalam NLTK, menggunakan *method sent_tokenize* untuk tokenisasi menjadi kalimat dan *word_tokenize* untuk tokenisasi menjadi kata.

```
1 sent_tokenize(text)
['Lorem ipsum dolor sit amet, consectetur adipiscing elit.',
 'Mauris vel diam quis mi molestie laoreet eget in ex.',
 'Suspendisse mattis cursus arcu vel porta.',
 'Maecenas tincidunt elit magna, quis ornare lorem iaculis et.']

1 word_tokenize(text)
['Lorem',
 'ipsum',
 'dolor',
 'sit',
 'amet',
 ',',
 'consectetur',
 'adipiscing']
```

Terdapat perbedaan antara *word_tokenize* dengan *string.split()*, di mana bila menggunakan *split*, maka tanda baca tidak dipisahkan, melainkan menjadi bagian yang dianggap sebagai kata.

```
1 text = "Lorem ipsum. Dolor sit amet."  
2  
1 word_tokenize(text)  
['Lorem', 'ipsum', '.', 'Dolor', 'sit', 'amet', '.']  
  
1 text.split()  
['Lorem', 'ipsum.', 'Dolor', 'sit', 'amet.']}
```

II. Case Folding

Case folding adalah merubah karakter huruf pada suatu *string* menjadi seragam kapitalisasinya. Case folding dapat berupa mengubah seluruh karakter menjadi huruf kecil (*lowercase*) maupun huruf besar (*uppercase*). Dalam *preprocessing* pada *text mining*, biasanya yang digunakan adalah *lowercase*. Mengubah suatu *string* menjadi *lowercase* ditunjukkan pada contoh di bawah ini.

```
1 text = "Lorem ipsum. Dolor sit amet."  
2 text.lower()  
  
'lorem ipsum. dolor sit amet.'
```

Tujuan dari *case folding* adalah menyeragamkan suatu kata yang sebenarnya sama, namun dibaca berbeda oleh komputer karena perbedaan karakter. Penyeragaman ini berguna saat nanti melakukan vektorisasi karena mengurangi dimensi dari vektor (dibahas selanjutnya). Sebagai contoh, kata “sawah” dan “Sawah” sebenarnya sama namun komputer mendeteksi perbedaan karakter, sehingga dianggap berbeda. Dengan adanya *case folding*, permasalahan ini dapat dihindari.

III. Stopword Removal

Stopword removal adalah proses menghilangkan kata-kata *stopword*. *Stopword* sendiri adalah kata-kata yang sekiranya tidak memiliki makna utama dalam kalimat, atau tidak dapat mengartikan kalimat bila berdiri sendiri. Kata hubung dan kata sambung adalah contoh dari *stopword*. Dalam Python, proses *stopword* dilakukan tanpa menggunakan *library* bawaan. Namun, dalam proses *stopword removal*, setiap *string* harus ditokenisasi terlebih dahulu, sehingga dapat dibantu dengan NLTK. Dalam beberapa bahasa, NLTK juga

menyediakan daftar *stopword*, namun belum ada untuk Bahasa Indonesia. Contoh *stopword removal* terdapat pada gambar di bawah ini.

```

1 from nltk.corpus import stopwords
2 from nltk import word_tokenize
3
4 text = "Yonanda lives in Tokyo. She likes that place so much"
5 stop_words = stopwords.words('english')
6 filtered_text = []
7 for word in word_tokenize(text):
8     if word not in stop_words:
9         filtered_text.append(word)
10 filtered_text
[ 'Yonanda', 'lives', 'Tokyo', '.', 'She', 'likes', 'place', 'much']

```

Atau bila menggunakan *list comprehension*, maka akan seperti ini.

```

1 text = "Yonanda lives in Tokyo. She likes that place so much"
2 stop_words = stopwords.words('english')
3 filtered_text = [w for w in word_tokenize(text) if not w in stop_words]
4 filtered_text
[ 'Yonanda', 'lives', 'Tokyo', '.', 'She', 'likes', 'place', 'much']

```

Stopword removal berguna menghilangkan kata-kata yang tidak memiliki makna bila berdiri sendiri. Ini bermanfaat dalam pembentukan vektor, sehingga vektor yang terbentuk memiliki dimensi lebih kecil (akan dibahas selanjutnya).

IV. Non-ASCII Characters Removal

ASCII adalah singkatan dari *American Standard Code for Information Interchange*. Sederhananya, ASCII adalah karakter yang bisa dibaca oleh kebanyakan perangkat. Sehingga, pengilangan karakter non-ASCII diharapkan membantu proses tersebut. Karakter ASCII dalam Python disimpan dalam *string.printable*. Bila diperhatikan, *newline* (\n) dan tab (\t) juga termasuk di dalamnya, sehingga tidak dihilangkan dalam proses *non-ASCII characters removal*.

```

1 from string import printable
2 printable
'0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!#$%&`()*+,-./:;<=>?@[\\]^_`{|}~ \t\n\r\x0b\x0c'

```

Untuk menghilangkan karakter non-ASCII, maka kita dapat memanfaatkan fungsi bawaan Python yaitu *filter* dan *lambda*. Hal ini dicontohkan pada gambar di bawah ini.

```

1 text = "Ω is omega. β is beta"
2 filter(lambda x: x in printable, text)
<filter at 0xe288bf8d68>

```

Hasil *return* dari *filter* adalah sebuah *iterable* object, sehingga perlu digabungkan untuk mendapatkan hasil dalam *string*.

```
1 text = "Ω is omega. β is beta"
2 filtered_text = ''.join(filter(lambda x: x in printable, text))
3 filtered_text
' is omega. is beta'
```

V. Punctuation Removal

Punctuation removal atau penghilangan tanda baca adalah proses lain yang bersifat opsional untuk menghilangkan tanda baca dari kalimat. Hal ini dilakukan karena tanda baca sendiri tidak berarti apa-apa jika berdiri sendiri sehingga dapat dihilangkan sebelum kalimat divektorkan.

Punctuation removal memiliki cara yang mirip dengan *stopword removal*, dengan mengasumsikan bahwa tanda baca adalah *stopword*. Di dalam Python, tanda baca disimpan dalam *string.punctuation*.

```
1 from string import punctuation
2
3 text = "You know, I love it!"
4
5 filtered_text = [w for w in word_tokenize(text) if not w in punctuation]
6 filtered_text
['You', 'know', 'I', 'love', 'it']
```

Karena caranya yang mirip, *stopword removal* dan *punctuation removal* dapat dikombinasikan dalam satu *list comprehension*. Perlu dicatat bahwa proses *punctuation removal* di atas ini akan menghilangkan tanda baca yang berdiri sendiri, bukan tanda baca yang berada dalam kata seperti “jum’at” atau “berkaca-kaca”.

VI. Digit Removal

Sesuai namanya, digit removal adalah proses menghilangkan kata atau token yang berupa angka keseluruhan (semisal tahun). Proses ini juga opsional. Dalam Python, suatu *string* memiliki *method isdigit* untuk mengetahui apakah seluruh *string* adalah angka atau bukan.

Proses *digit removal* dapat dilakukan dengan tokenisasi dan *list comprehension*, seperti contoh di bawah ini.

```

1 text = "WW2 was in 1939"
2 filtered_text = [w for w in word_tokenize(text) if not w.isdigit()]
3 filtered_text

['WW2', 'was', 'in']

```

Perlu diperhatikan bahwa “2” pada “WW2” tidak ikut hilang karena dianggap sebagai satu kata utuh. Hal ini akan tergantung dari proses yang diharapkan. Namun, bila “2” dalam “WW2” ingin dihapuskan, maka dapat menggunakan filter dan lambda seperti pada *non-ASCII removal*.

```

1 text = "WW2 was in 1939"
2 filtered_text = ''.join(filter(lambda x: not x.isdigit() and x not in punctuation, text))
3 filtered_text

'WW was in '

```

Penggunaan tokenisasi dan *list comprehension* digunakan dengan cara memecah *string* menjadi token, sehingga *return*-nya berupa *list* atau kalimat tertokenisasi. Di sisi lain, penggunaan *filter* dan *lambda* memecah *string* per karakter. Bila diikuti dengan *join*, maka *return*-nya menjadi suatu *string*, bukan *list*.

VII. Stemming and Lemmatization

Stemming dan *lemmatization* adalah proses mengubah suatu kata ke dalam bentuk dasarnya. Stemming dan lemmatization memiliki perbedaan di mana stemming hanya mengurangi karakter hingga menjadi bentuk dasar, sedangkan lemmatization mengurangi dan mengubah karakter hingga menjadi bentuk dasar. Sebagai contoh, dari kata “*happiness*” akan menjadi “*happi*” bila dilakukan *stemming*, sedangkan *lemmatization* akan merubah kata tersebut menjadi “*happy*”.

Dalam Python, baik *stemming* dan *lemmatization* berada pada *nltk.stem*, seperti pada contoh di bawah ini. Terlihat perbedaan signifikan antara hasil *stemming* dan *lemmatization*.

```

1 from nltk.stem import WordNetLemmatizer
2 lemmatizer = WordNetLemmatizer()
3 print("flies :", lemmatizer.lemmatize("flies"))

flies : fly

1 from nltk.stem import PorterStemmer
2 stemmer = PorterStemmer()
3 print("flies :", stemmer.stem("flies"))

flies : fli

```

WordNetLemmatizer dan *PorterStemmer* adalah *method* yang berupa estimator, sehingga harus diinisiasi terlebih dahulu. Nama-nama ini mengacu pada algoritma yang digunakan untuk proses *stemming* dan *lemmatization*. Sehingga, masih ada *method* lain dalam melakukan *stemming* dan *lemmatization* pada NLTK dengan algoritma yang berbeda.

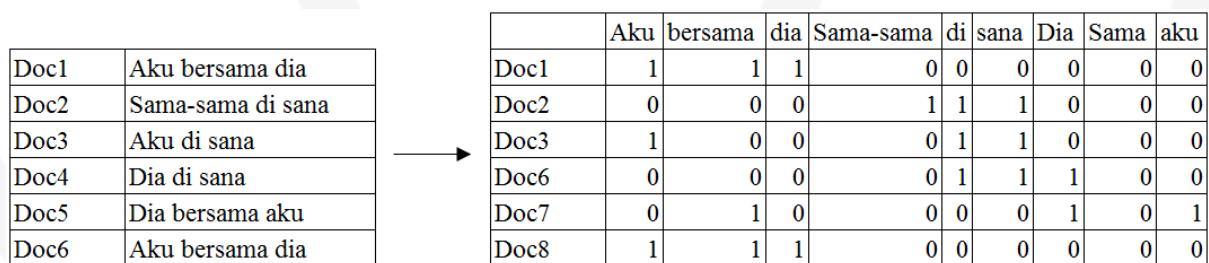
Baik *stemming* dan *lemmatization* digunakan untuk menyeragamkan kata yang memiliki akar kata yang sama, seperti *rock* dan *rocks*, *fly* dan *flies*, dan sebagainya. Hal ini akan mereduksi ukuran dari vektor yang terbentuk.

5) Document Vector and Bag-of-Words

Setelah mengalami beberapa *preprocessing* seperti yang dibahas sebelumnya, maka tidak berarti data berupa teks atau *string* langsung dapat digunakan dalam proses *machine learning*. Seperti yang kita lihat sebelumnya, bahwa proses *machine learning* selalu dimulai dari data berupa tabel dengan tiap nilai di dalamnya adalah numerik atau angka. Oleh karena itu, suatu teks perlu diubah dalam bentuk vektor. Teks yang diubah dalam bentuk vektor biasa disebut sebagai *document vector* atau vektor dokumen.

Secara sederhana, *document vector* adalah representasi sebuah teks dalam vektor di mana tiap *index* dalam vektor merepresentasikan suatu kata, dan tiap nilai dalam index tersebut merepresentasikan bobot dari kata yang dimaksud di dalam dokumen teks.

Sebagai contoh, diberikan suatu gambar di bawah ini. Tiap teks memiliki nama Doc1 sampai Doc6. Tabel pada sebelah kiri adalah bentuk dalam teks, sedangkan tabel sebelah kanan adalah bentuk dalam vektor.



Doc1	Aku bersama dia
Doc2	Sama-sama di sana
Doc3	Aku di sana
Doc4	Dia di sana
Doc5	Dia bersama aku
Doc6	Aku bersama dia

	Aku	bersama	dia	Sama-sama	di	sana	Dia	Sama	aku
Doc1	1	1	1	0	0	0	0	0	0
Doc2	0	0	0	1	1	1	0	0	0
Doc3	1	0	0	0	1	1	0	0	0
Doc6	0	0	0	0	1	1	1	0	0
Doc7	0	1	0	0	0	0	1	0	1
Doc8	1	1	1	0	0	0	0	0	0

Pada gambar di atas, kita dapat lihat bahwa tiap kata direpresentasikan dalam kolom, dan tiap baris merepresentasikan teks yang berbeda. Apabila dituliskan, maka Doc3 adalah suatu teks “Aku di sana” yang memiliki bentuk vektor [1, 0, 0, 0, 1, 1, 0, 0, 0]. Perlu diperhatikan pada gambar di atas bahwa kata “Aku” dan “aku” direpresentasikan dalam

index yang berbeda. Begitu pula “bersama” dan “sama”, sekalipun kedua kata ini memiliki akar kata yang sama. Di sini *preprocessing* berkонтibusi untuk mengurangi dimensi dari vektor. Tabel pada sebelah kanan tanda panah ini sering disebut sebagai *bag-of-words*, sedangkan [1, 0, 0, 0, 1, 1, 0, 0, 0] disebut sebagai document vector dari Doc3

Bobot dari tiap kolom pada *bag-of-words* dapat ditentukan sesuai model yang ingin dibangun:

1. Menggunakan nilai *boolean*, sehingga apabila suatu kata terdapat dalam dokumen, maka nilai dalam *index* katanya adalah satu. Jika sebaliknya maka nol.
2. Menggunakan nilai frekuensi, sehingga apabila suatu kata terdapat dalam dokumen, maka nilai dalam *index* katanya sesuai dengan kemunculan kata tersebut dalam dokumen. Bernilai nol bila kata tersebut tidak ada dalam dokumen. Hal ini disebut sebagai *term frequency* (tf)
3. Menggunakan nilai frekuensi yang dengan dengan invers dari kemunculan kata dalam seluruh dokumen. Keseluruhan dokumen sering disebut sebagai *corpus*. Hal ini disebut sebagai *term frequency – inverse document frequency* (tf-idf). Bila dituliskan, tf-idf dari kata *i* pada dokumen *j* pada *corpus* yang berisi *N* buah dokumen ditulis seperti berikut:

$$w_{i,j} = tf_{i,j} \times \log \frac{N}{df_i}$$

$tf_{i,j}$ menyimbolkan frekuensi kata *i* dalam dokumen *j*, sedangkan df_i menghitung berapa banyak dokumen yang memuat kata *i*. Penggunaan *idf* bertujuan untuk mengurangi bobot dari suatu kata apabila kata tersebut terlalu sering muncul di dalam keseluruhan dokumen. Bila kata *i* pada dokumen *j* sering dimuat dalam dokumen lain, dapat dikatakan bahwa kata *i* kurang memiliki makna signifikan yang membedakan dokumen *j* dengan dokumen lainnya.

Selain bentuk pembobotan di atas, masih terdapat beberapa cara lainnya, seperti kombinasi antara nilai boolean dengan *idf*, atau menggunakan normalisasi pada *tf*-nya. Namun, tf-idf adalah kombinasi yang paling sering digunakan dalam pembobotan *bag-of-word*.

Dalam Python, pembobotan *tf-idf* disediakan dalam *sklearn*, utamanya *sklearn.feature_extraction.text.TfidfVectorizer*. Method ini perlu diinisiasi terlebih dahulu, sehingga *objek* dari *method* ini berupa estimator. Oleh karena estimator, maka perlu

dilakukan *fit* untuk membentuk *bag-of-words*. Proses *fit* dilakukan dengan bagian *training data* setelah dipisahkan dengan *testing data*. Sebagai contoh, diberikan suatu data seperti berikut.

	review	sentiment
0	This is good product	positive
1	This is also fine.	positive
2	Good quality for this one.	positive
3	This product is good, I guess.	positive
4	But, this is bad.	negative
5	This one is bad too!	negative
6	It is horrible!	negative
7	The quality of this one is bad.	negative

Pada data ini, kita akan membagi data secara *holdout* dengan *train_test_split* dari *sklearn*.

```

1 from sklearn.model_selection import train_test_split
2
3 X_train, X_test, y_train, y_test = train_test_split(data['review'],
4                                                    data['sentiment'],
5                                                    test_size=0.25)

```

Kemudian, proses dilanjutkan dengan pembentukan *bag-of-words* dengan tf-idf.

```

1 from sklearn.feature_extraction.text import TfidfVectorizer
2
3 vectorizer = TfidfVectorizer(lowercase=True,
4                             binary=False,
5                             use_idf=True,
6                             max_features=None)
7 vectorizer.fit(X_train)

TfidfVectorizer()

```

Dalam *TfidfVectorizer*, terdapat beberapa parameter yang bisa dimodifikasi, diantaranya:

- *lowercase*, yang bernilai *True* apabila ingin semua teks dalam data menjadi *lowercase*.
- *binary*, yaitu parameter *boolean* yang mengubah seluruh frekuensi kata yang tidak nol menjadi satu bila bernilai *True*.

- *use_idf*, yang menentukan apakah idf digunakan dalam pembobotan vektor. Bila tidak ingin menggunakan idf, berikan nilai *False* pada parameter ini.
- *max_features*, yang menerima nilai *integer* N, dan akan mengambil N kata teratas berdasarkan frekuensinya pada pembentukan *bag-of-words*. Bila bernilai *None*, maka seluruh kata digunakan dalam pembentukan *bag-of-words* (tidak hanya top N)

Setelah estimatorenya dipanggil dan dilakukan *fit*, maka vectorizer ini siap digunakan dalam mengubah *training data* maupun *testing data* ke dalam vektor dengan method *transform* seperti di bawah ini.

```

1 X_train_vector = vectorizer.transform(X_train)
2 X_train_vector
<6x14 sparse matrix of type '<class 'numpy.float64'>'  

    with 28 stored elements in Compressed Sparse Row format>

1 X_test_vector = vectorizer.transform(X_test)
2 X_test_vector
<2x14 sparse matrix of type '<class 'numpy.float64'>'  

    with 6 stored elements in Compressed Sparse Row format>

```

Hasil kembalian dari *transform* adalah *sparse matrix* (matriks yang nilainya didominasi oleh 0). Untuk melihat hasilnya dalam bentuk *array*, maka gunakan *toarray* sebagai *method* dari *sparse matrix* tersebut.

```

1 X_train_vector.toarray()
array([[0.          , 0.          , 0.          , 0.53465962, 0.        ,  

       0.          , 0.39566011, 0.          , 0.          , 0.        ,  

       0.63328133, 0.          , 0.          , 0.39566011],  

       [0.          , 0.          , 0.56919132, 0.39405822, 0.        ,  

       0.          , 0.          , 0.          , 0.          , 0.46674503,  

       0.          , 0.46674503, 0.          , 0.29161193],  

       [0.55318292, 0.67460155, 0.          , 0.          , 0.        ,  

       0.          , 0.34561641, 0.          , 0.          , 0.        ,  

       0.          , 0.          , 0.          , 0.34561641],  

       [0.          , 0.          , 0.          , 0.          , 0.        ,  

       0.66482563, 0.34060794, 0.66482563, 0.          , 0.        ,  

       0.          , 0.          , 0.          , 0.          , 0.        ],  

       [0.3847574 , 0.          , 0.          , 0.          , 0.        ,  

       0.          , 0.24038788, 0.          , 0.46920815, 0.3847574 ,  

       0.          , 0.3847574 , 0.46920815, 0.24038788],  

       [0.          , 0.          , 0.          , 0.42315952, 0.61122624,  

       0.          , 0.31314754, 0.          , 0.          , 0.        ,  

       0.50121426, 0.          , 0.          , 0.31314754]])
```

Selain *fit* dan *transform*, estimator tf-idf ini juga memiliki method *get_feature_names* untuk mengetahui kolom-kolom berupa kata yang digunakan dalam *bag-of-words*.

```
1 vectorizer.get_feature_names()  
['bad',  
 'but',  
 'for',  
 'good',  
 'guess',  
 'horrible',  
 'is',  
 'it',  
 'of',  
 'one',  
 'product',  
 'quality',  
 'the',  
 'this']
```

Bila diperhatikan, *stopword* seperti *the*, *this*, *of*, *is*, dan lainnya masih ada di dalam *bag-of-word* ini. Bila dilakukan *preprocessing*, maka kata-kata ini akan dihilangkan dan mengurangi dimensi vektor. Pengurangan dimensi vektor berakibat semakin cepatnya proses komputasi saat proses *training* model.

Kita dapat mengkombinasikan nama-nama kolom ini dengan *sparse matrix* untuk membentuk *bag-of-words* dalam bentuk *DataFrame*. Dengan data teks yang sudah dalam bentuk vektor, maka proses klasifikasi sentimen siap dilakukan.

6) Training Text Classifier Model

Proses klasifikasi dengan teks yang sudah dibentuk menjadi vektor dapat langsung dilakukan. Pada kasus ini, kita akan mencoba membentuk model dengan SVM. Perlu diingat bahwa yang digunakan untuk *training* adalah prediktor yang sudah dalam bentuk vektor.

```
1 from sklearn.svm import SVC  
2  
3 model = SVC()  
4 model.fit(X_train_vector, y_train)  
  
SVC()
```

Apabila proses *training* sudah selesai, maka kita dapat menguji dengan *testing data* yang sudah dalam bentuk vektor juga.

```
1 from sklearn.metrics import classification_report  
2  
3 y_pred = model.predict(X_test_vector)  
4 print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
negative	1.00	1.00	1.00	1
positive	1.00	1.00	1.00	1
accuracy			1.00	2
macro avg	1.00	1.00	1.00	2
weighted avg	1.00	1.00	1.00	2

Perlu dicatat bahwa hasil ini adalah hasil dengan sample data yang sangat sedikit dan memiliki variasi kata yang hanya 28 kata. Dalam kenyataannya, jumlah kata dan dokumen sangatlah banyak. Ini yang menyebabkan *preprocessing* sangatlah penting untuk mengurangi dimensi vektor, sehingga proses jadi lebih efisien.

Computer Vision - Image Processing

7) Image Processing

Pada modul kali ini kita akan membahas bagaimana melakukan teknik image processing menggunakan library scikit-image. Kita akan belajar bagaimana cara transformasi dan melakukan image manipulation. Sebelumnya kita perlu tahu apa itu Image processing. Image Processing adalah bagian dari subset ilmu dari Computer Vision. Sistem Computer Vision menggunakan algoritma image processing untuk meniru vision atau penglihatan pada manusia. Algoritma image processing mencoba dan melakukan metode untuk melakukan operasi dan mengekstraksi informasi yang berguna dari gambar untuk dianalisa sehingga bisa dijadikan bahan pembuat keputusan.

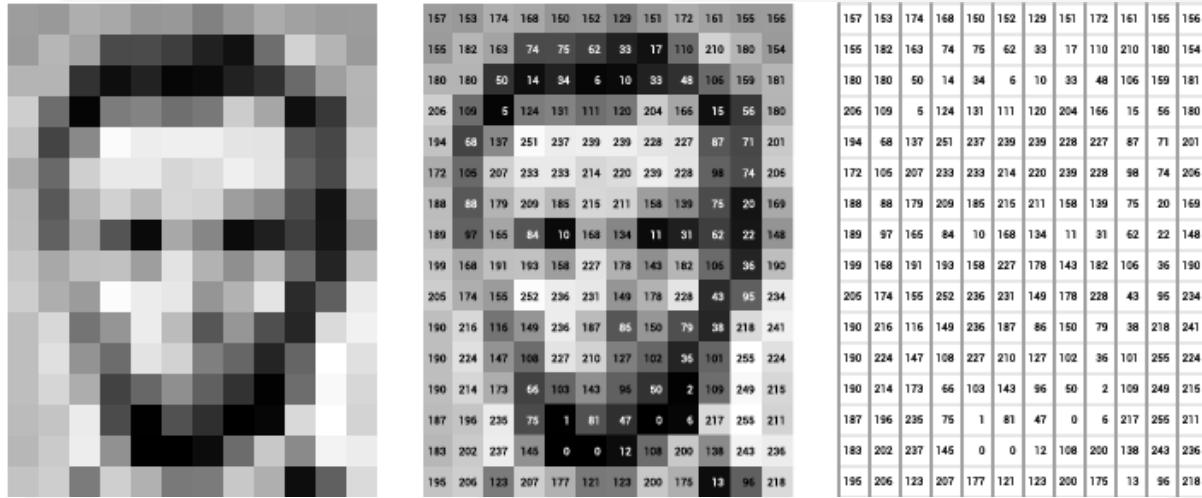
Dengan mendapatkan informasi dari gambar, kita dapat melakukan perhitungan dimana perhitungan ini bisa dijadikan bahan pendukung untuk image analytic seperti mendeteksi sudut, objek, bahkan wajah manusia. Dengan memahami teknik pemrosesan gambar, kita dapat menerapkannya kedalam beberapa aplikasi, contohnya aplikasi untuk merestorasi gambar yang mempunyai pixel yang rendah atau gambar yang rusak, aplikasi untuk melakukan diagnosis penyakit pada foto rekam medis, aplikasi untuk mendeteksi objek yang mencurigakan pada kamera sistem keamanan, Robotik yang menerapkan computer vision, Pendekripsi objek pada Autonomous vehicle dan lainnya.

Tujuan dari image processing dapat dibagi menjadi 5 kategori:

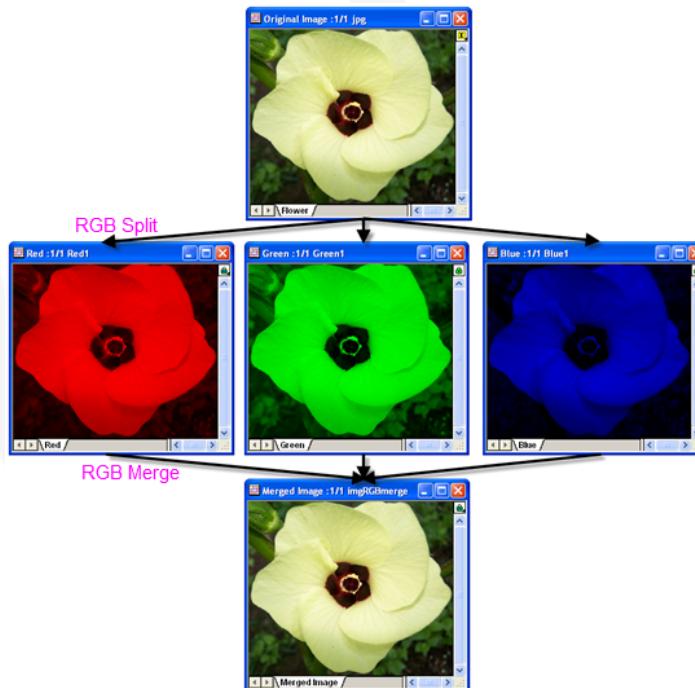
- 1) Untuk visual observation, dimana ditujukan untuk mengobservasi objek yang tidak mudah terlihat.
- 2) Penajaman dan restorasi gambar untuk membuat gambar yang lebih bagus.
- 3) Image Retrieval: Untuk mendapatkan pola atau informasi yang berguna dari gambar.
- 4) Pengukuran pola untuk mengukur suatu objek dari gambar
- 5) Image recognition: untuk mengenali objek dari gambar.

I. Image

Gambar digital adalah sekumpulan / sebuah array / sebuah matrix dari pixel (picture elements) yang tersusun dari columns dan rows yang berisi informasi tentang warna dan intensitasnya.



Gambar hitam putih biasanya bersifat 2 Dimensi dan tidak memiliki informasi warna sedangkan Gambar berwarna bersifat Matrix 3 Dimensi yang memiliki informasi warna RGB Channel atau disebut juga 3 color channel.



			row	0	1	2				
			column	0	.392	.482	.576			
			channel	0	.478	.63	.169	.263	.376	
			0	.580	.79	.263	.44	.306	.376	.451
			1					.376	.478	.561
			2					.443	.569	.674

II. Scikit-Image

Untuk melakukan image processing, kita dapat menggunakan library scikit-image. Scikit-image adalah library pada Python untuk melakukan pemrosesan gambar. Scikit-image menggunakan metode Machine Learning yang sudah disediakan oleh library itu sendiri dan dapat melakukan operasi kompleks pada gambar dengan waktu yang singkat.

Kita bisa memulai untuk installasi scikit-image menggunakan:

```
pip install scikit-image
```

Setelah kita melakukan installasi kita bisa memanggil beberapa API dari module scikit-image:

```
# Import the modules from skimage
from skimage import data, color
import matplotlib.pyplot as plt
```

Dan sebelum kita memulai untuk mengolah gambar, ada baiknya kita membuat user-defined function untuk memudahkan kita visualisasi gambar kita, dimana fungsi dibawah merubah matrix pada numpy ndarray menjadi sebuah gambar.

```
def show_image(image, title='Image', cmap_type='gray'):
    plt.imshow(image, cmap=cmap_type)
    plt.title(title)
    plt.axis('off')
    plt.show()
```

Scikit-image sudah menyediakan beberapa data yang bisa langsung kita gunakan untuk experiment kita. Untuk melihat gambar apa saja yang bisa kita gunakan dari scikit-image, bisa dilihat di link berikut:

<https://scikit-image.org/docs/stable/api/skimage.data.html?highlight=data#module-skimage.data>

Untuk pertama-tama, kita bisa gunakan data.coffee() dan data.coins() dan kita bisa tampilkan gambar gambar kopi dan gambar koin menggunakan fungsi show_image(), dan bisa di perhatikan matrix pada gambar kopi dan koin beserta dimensinya menggunakan attribute .shape:

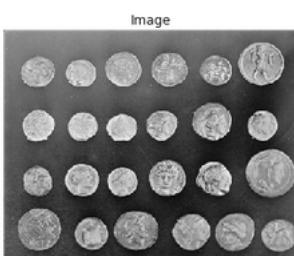
```
gambar_kopi = data.coffee()
gambar_koin = data.coins()

help(show_image)

Help on function show_image in module __main__:

show_image(image, title='Image', cmap_type='gray')
```

```
show_image(gambar_kopi)
show_image(gambar_koin)
```



```
print(gambar_kopi)
print(gambar_koin)
```

```
[[[ 21  13  8]
 [ 21  13  9]
 [ 20  11  8]
 ...
 [228 182 138]
 [231 185 142]
 [228 184 140]]]
```

```
[[ 21  13  7]
 [ 21  13  9]
 [ 20  14  7]
 ...
```

```
: print(data.coffee().shape)
print(data.coins().shape)
```

```
(400, 600, 3)
(303, 384)
```

Kita juga bisa menggunakan parameter yang kedua pada fungsi `show_image()` untuk memberikan judul dan membedakan mana gambar yang RGB dan mana gambar yang grayscale.

Kemudian, kita juga bisa mencoba data gambar yang lain dan mencoba untuk merubah gambar RGB menjadi gambar grayscale dengan method `.rgb2gray()`.

```
# Load the rocket image
rocket = data.rocket()

# Convert the image to grayscale
gray_scaled_rocket = color.rgb2gray(rocket)

# Show the original image
show_image(rocket, 'Original RGB image')

# Show the grayscale image
show_image(gray_scaled_rocket, 'Grayscale image')
```

Original RGB image

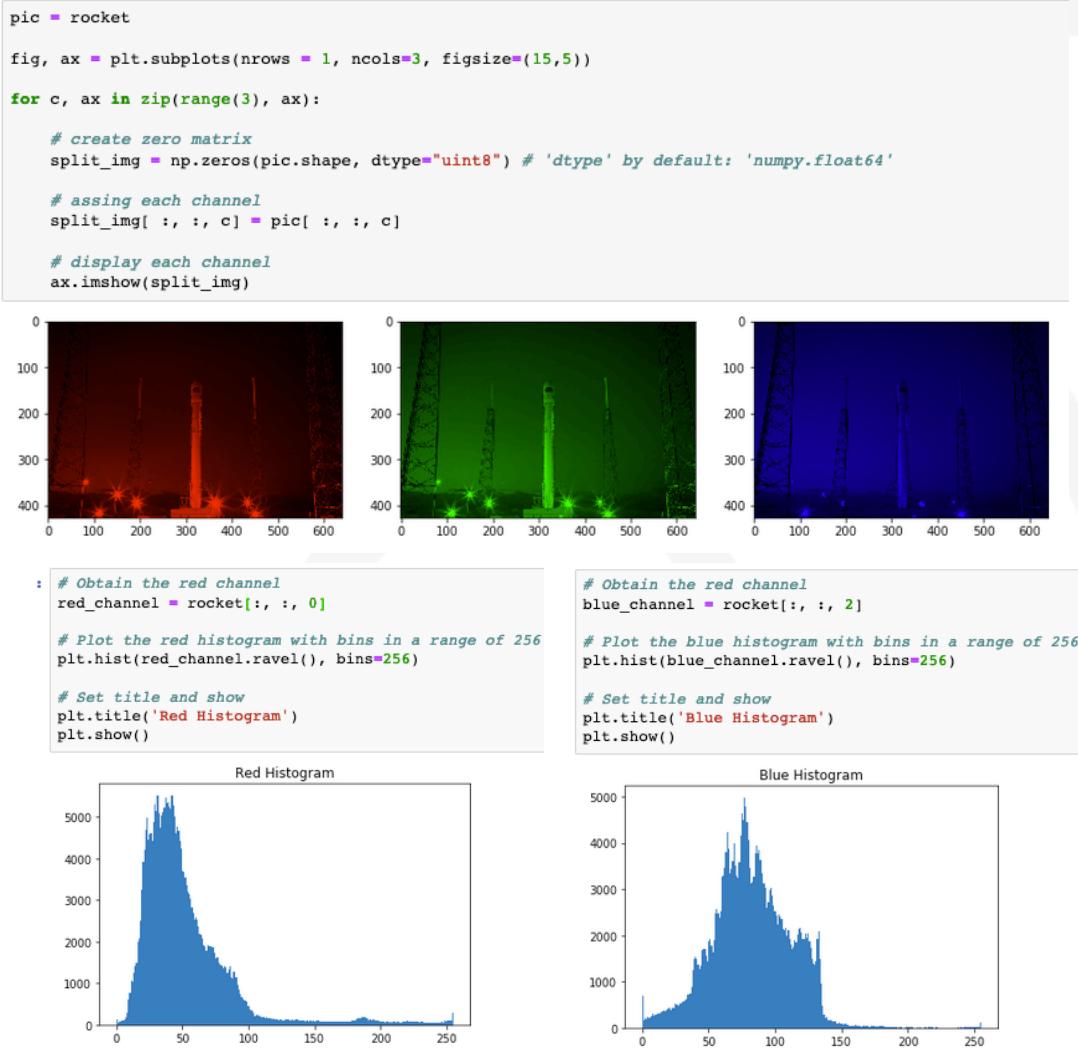


Grayscale image



<code>rocket.shape</code>	<code>gray_scaled_rocket.shape</code>
(427, 640, 3)	(427, 640)
<code>rocket.size</code>	<code>gray_scaled_rocket.size</code>
819840	273280

Untuk melihat lebih lanjut bagaimana 3 channel dipisahkan dan menjadi gambar tersendiri, kita bisa melakukan plotting untuk Red Channel pada index 0, Green Channel pada index 1, dan Blue Channel pada index 2.



Dengan menggunakan histogram plot, kita bisa menganalisa distribusi warna pada masing-masing color channel dimana mendekati 0 adalah lebih ke warna gelap dan mendekati 255 adalah lebih ke warna RGB. Untuk lebih memahami, kita bisa melihat RGB Calculator yang disediakan oleh W3School.

https://www.w3schools.com/colors/colors_rgb.asp



Dari Histogram, kita bisa belajar tentang karakteristik gambar. Histogram biasanya digunakan untuk melakukan teknik thresholding, untuk melihat atau merubah brightness dan contrast, dll.

III. NumPy for Images

Dengan menggunakan Numpy kita dapat melakukan teknik image processing yang simple seperti flipping gambar, mengekstrak fitur dan melakukan Analisa pada gambar. Untuk melakukan import gambar, kita bisa menggunakan fungsi imread dari matplotlib.

```
# Loading the image using Matplotlib
foto_buah = plt.imread('../images/fruits-2.jpg')
type(foto_buah)
numpy.ndarray
show_image(foto_buah)
```

Image

```
import numpy as np
# Flip the image vertically
buah_vertical_flip = np.fliplr(foto_buah)

# Flip the image horizontally
buah_horizontal_flip = np.flipud(buah_vertical_flip)

# Show the resulting image
show_image(buah_horizontal_flip, 'Gambar buah kebalik')
```

Gambar buah kebalik



8) Image Processing Techniques

I. Thresholding

Thresholding adalah metode untuk melakukan teknik image segmentation. Thresholding digunakan untuk mengisolasi elements dan biasanya digunakan untuk aplikasi seperti object detection atau facial recognition. Salah satu penerapan thresholding, biasanya digunakan untuk melakukan partisi pada background dan foreground dari gambar grayscale, dengan membuatnya benar-benar menjadi hitam dan putih.

Kemudian kita akan melakukan perbandingan pada setiap pixel dengan threshold yang diberikan, jika pixel dibawah dari threshold, kita jadikan putih dan jika lebih dari threshold, maka kita akan membuatnya hitam. Ada beberapa algoritma global thresholding yang bisa digunakan, salah satunya adalah `threshold_otsu()` dari module filter untuk mendapatkan nilai yang optimal. Jika background dari sebuah gambar terlihat serupa / uniform, maka global threshold tepat digunakan, tapi jika background lebih beragam local threshold algorithm, `threshold_local()`, memberikan hasil yang lebih baik.

```
# Import threshold and gray convertor functions
from skimage.filters import threshold_otsu
from skimage.color import rgb2gray

# Turn the image grayscale
gray_tools_image = rgb2gray(foto_buah)

# Obtain the optimal thresh
thresh = threshold_otsu(gray_tools_image)

# Obtain the binary image by applying thresholding
binary_image = gray_tools_image > thresh

# Show the resulting binary image
show_image(binary_image, 'Binarized image')
```

Binarized image



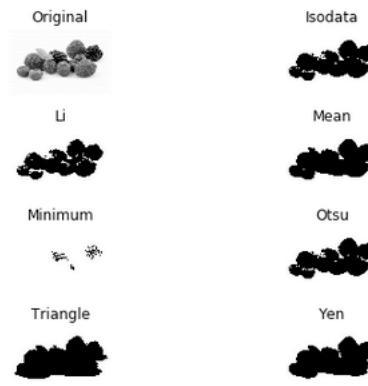
```
# Import the try all function
from skimage.filters import try_all_threshold

# Import the rgb to gray convertor function
from skimage.color import rgb2gray

# Turn the fruits_image to grayscale
grayscale = rgb2gray(foto_buah)

# Use the try_all method on the resulting grayscale image
fig, ax = try_all_threshold(grayscale, verbose=False)

# Show the resulting plots
plt.show()
```



II. Filtering

Filtering adalah teknik untuk memodifikasi atau meningkatkan kualitas suatu objek pada gambar. Intinya, dengan filtering kita menerapkan fungsi matematis pada gambar dimana fungsi tersebut mampu untuk mempertegas atau menghilangkan fitur dari gambar, seperti edge/tepi. Fungsi matematis juga bisa bisa digunakan untuk smoothing /memperhalus atau sharpening/mempertajam kualitas gambar.

Beberapa operasi pada image processing melakukan pemrosesan pada suatu gambar dengan hanya mengambil sebagian dari gambar (sectioning). Section ini disebut juga sebagai blocks atau neighborhoods. Operasi image yang melibatkan blocks per blocks ini diterapkan pada teknik filter, histogram equalization untuk memperbaiki contrast dan fungsi morphologi.

Figure 01

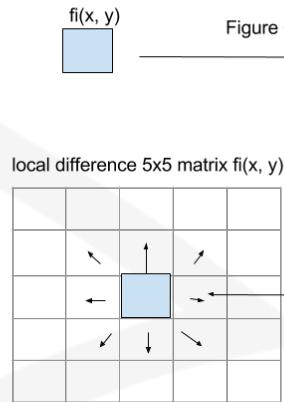
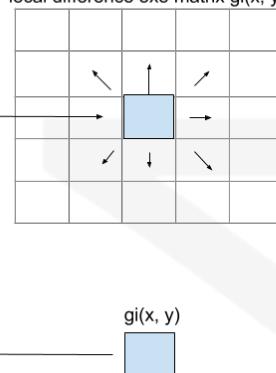


Figure 02
local difference 5x5 matrix $gi(x, y)$



Dengan Filtering kita dapat melakukan edge detection. Teknik ini sangat berguna untuk mencari batas-batas dari object didalam gambar, dan bahkan mampu melakukan segmentasi dan mengekstrak informasi seperti ada berapa koin yang ada pada gambar. Informasi terbesar dan terpenting dari suatu objek tentang bagaimana kondisi dan bentuk pada objek tersebut terletak pada tepi/edgenya.

Salah satu algoritma yang dapat digunakan untuk mendeteksi edge adalah Sobel, Sobel mampu melakukan filter pada tepi dari object dan kita bisa memanggil filter ini dari scikit image pada module filters.

```

foto_sabun = plt.imread('../images/soap_image.jpg')

# Import the color module
from skimage import color

# Import the filters module and sobel function
from skimage.filters import sobel

# Make the image grayscale
foto_sabun_grayscale = color.rgb2gray(foto_sabun)

# Apply edge detection filter
edge_sobel = sobel(foto_sabun_grayscale)

# Show original and resulting image to compare
show_image(foto_sabun, "Original")
show_image(edge_sobel, "Edges with Sobel")

```



Untuk melakukan komparasi pada gambar yang original yang yang sudah di filter, kita bisa membuat user-defined function sebagai berikut:

```
def plot_comparison(original, filtered, title_filtered):
    fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(10, 10), sharex=True,
                                  sharey=True)
    ax1.imshow(original, cmap=plt.cm.gray)
    ax1.set_title('original')
    ax1.axis('off')
    ax2.imshow(filtered, cmap=plt.cm.gray)
    ax2.set_title(title_filtered)
    ax2.axis('off')
```

Kemudian kita bisa belajar salah satu teknik dari filtering, yaitu adalah smoothing, dimana kita bisa membuat gambar menjadi blur dan mengurangi noise dengan menggunakan Gaussian filter. Kita bisa melihat bagaimana kita menerapkan Gaussian filter pada gambar bangunan dan kita bisa membandingkan jika Gaussian filter bisa mengurangi contrast dan membuat blur edges.

```
foto_bangunan = plt.imread('../images/building_image.jpg')

# Import Gaussian filter
from skimage.filters import gaussian

# Apply filter
gaussian_image = gaussian(foto_bangunan, multichannel=True)

plot_comparison(foto_bangunan, gaussian_image, "Reduced sharpness Gaussian")
```



III. Contrast Enhancement

Contrast Enhancement adalah salah satu jenis Image enhancement yang bisa sangat berguna pada beberapa area. Sering kali seperti pada gambar medical images khususnya pada X-ray mempunyai contrast yang sangat rendah, dan membuat kita cukup sulit melihat dan memperhatikan spot/objek penting pada gambar tersebut. Ketika kita bisa menaikkan contrast, maka detail-detail tersebut akan semakin mudah terlihat.

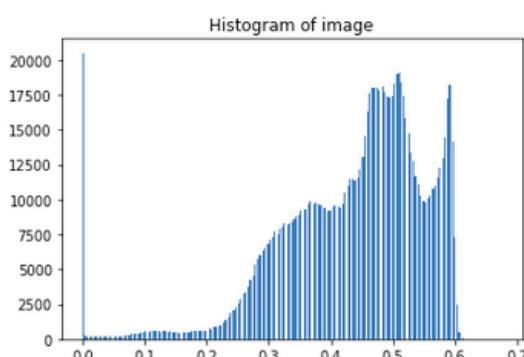
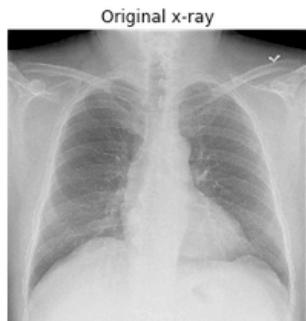
Kita bisa melihat persebaran kontras dengan menggunakan histogram. Kontras bisa diujur dari perbedaan maximum dan minimum intensitas pixel pada gambar. Gambar yang mempunyai contrast rendah biasanya hanya mempunyai perbedaan/persebaran yang dekat diantara pixel gelap dan pixel terangnya. Biasanya antara mempunyai histogram dengan skew condong ke kanan bisa disimpulkan bahwa gambar tersebut mostly terang, dan jika skew condong ke kanan, gambar tersebut condong ke gelap dan jika skew berada di tengah-tengah, maka gambar tersebut lebih condong ke abu-abu.

Dengan meningkatkan level contrast, berarti kita juga meningkatkan persebaran dari intensitas pixel. Persebaran akan semakin di stretching sehingga jarak penuh dari intensitas pixel pada gambar bisa terpenuhi.

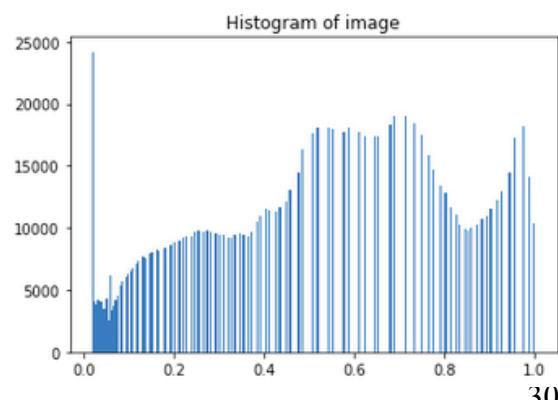
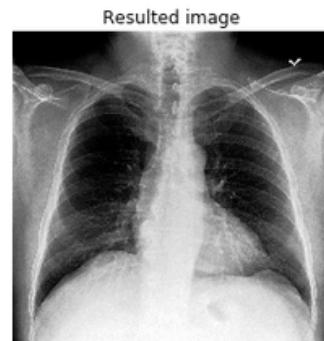
```
foto_xray = plt.imread('../images/chester_xray_image.png')
# Import the required module
from skimage import exposure

# Show original x-ray image and its histogram
show_image(foto_xray, 'Original x-ray')

plt.title('Histogram of image')
plt.hist(foto_xray.ravel(), bins=256)
plt.show()
```



```
# Use histogram equalization to improve the contrast
xray_image_eq = exposure.equalize_hist(foto_xray)
# Show the resulting image
show_image(xray_image_eq, 'Resulted image')
plt.title('Histogram of image')
plt.hist(xray_image_eq.ravel(), bins=256)
plt.show()
```



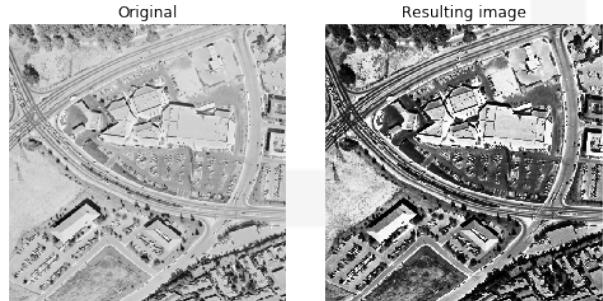
Kita bisa juga mencoba dengan contoh lain, seperti merubah contrast dari gambar peta yang ditangkap satelit misalnya.

```
image_aerial = plt.imread('../images/image_aerial.tif')

# Import the required module
from skimage import exposure

# Use histogram equalization to improve the contrast
image_eq = exposure.equalize_hist(image_aerial)

# Show the original and resulting image
show_image(image_aerial, 'Original')
show_image(image_eq, 'Resulting image')
```



Untuk mendapatkan contrast yang lebih baik pada gambar berwarna, kita bisa menggunakan fungsi Contrast Limited Adaptive Histogram Equalization (CLAHE) yang bisa dipanggil dari module exposure.

```
original_image = data.coffee()

# Apply the adaptive equalization on the original image
adaphist_eq_image = exposure.equalize_adapthist(original_image, clip_limit=0.03)

# Compare the original image to the equalized
plot_comparison(original_image, adaphist_eq_image, '#ImageProcessingWithCLAHE')
```



IV. Transformation

Proses transformasi gambar biasanya diperlukan dalam proses Image Augmentation. Dimana kita bisa memperbanyak gambar dari satu gambar, sehingga kita akan memiliki gambar yang berbeda-beda. Biasanya proses memperbanyak gambar ini sangat diperlukan ketika kita melakukan training dengan menggunakan Machine Learning/Deep Learning, sehingga model bisa mendapatkan data yang lebih banyak untuk dipelajari.

Proses Transformasi ini biasanya meliputi proses rotating, rescaling atau resizing gambar. Sebelumnya kita sudah mencoba untuk melakukan rotasi pada gambar dengan menggunakan library NumPy, sekarang kita akan mencoba menggunakan fungsi pada skimage.

```

from skimage.transform import rotate, rescale
gambar_kucing = plt.imread('../images/image_cat.jpg')

# Rotate the image 90 degrees clockwise
rotated_cat_image = rotate(gambar_kucing, -90)
# Rescale with anti aliasing
rescaled_with_aa = rescale(rotated_cat_image, 1/4, anti_aliasing=True, multichannel=True)

# Rescale without anti aliasing
rescaled_without_aa = rescale(rotated_cat_image, 1/4, anti_aliasing=False, multichannel=True)

# Show the resulting images
show_image(rescaled_with_aa, "Transformed with anti aliasing")
show_image(rescaled_without_aa, "Transformed without anti aliasing")

/usr/local/anaconda3/lib/python3.7/site-packages/skimage/transform/_warps.py:105: UserWarning: Th
stant', will be changed to 'reflect' in skimage 0.15.
warn("The default mode, 'constant', will be changed to 'reflect' in "

```

Transformed with anti aliasing



Transformed without anti aliasing



V. Edge Detection

Sebelumnya kita telah berlajar untuk mendeteksi tepi dengan menggunakan Sobel Filtering, ada algoritma lain yang lebih sering digunakan untuk medeteksi tepi yaitu Canny Edge Detection. Algoritma Canny memberikan akurasi yang lebih bagus dalam mendeteksi tepi dan lebih cepat dibandingkan Algoritma Sobel.

```

from skimage.feature import canny

# Apply canny edge detector
canny_edges = canny(gambar_koin)

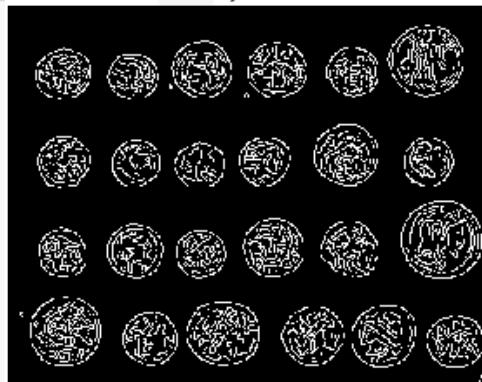
# Apply canny edge detector with a sigma of 1.8
edges_1_8 = canny(gambar_koin, sigma=1.8)

# Apply canny edge detector with a sigma of 2.2
edges_2_2 = canny(gambar_koin, sigma=2.2)

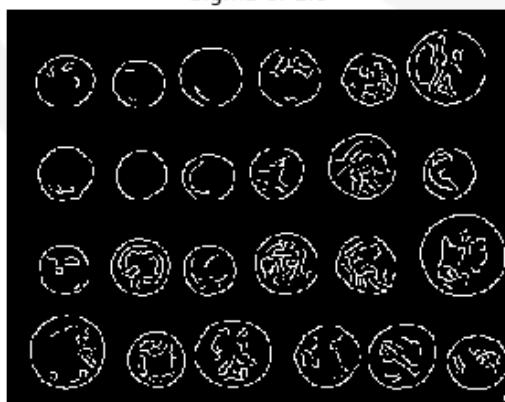
# Show resulting images
show_image(canny_edges, "Canny default")
show_image(edges_1_8, "Sigma of 1.8")
show_image(edges_2_2, "Sigma of 2.2")

```

Canny default



Sigma of 1.8



Sigma of 2.2



Algoritma Canny mempunyai hyperparameter sigma untuk menentukan derajat penerapan Gaussian filter, dimana semakin kecil nilai sigma, maka semakin kecil juga intensitas penerapan Gaussian pada gambar, sehingga semakin sedikit noise yang dikurangi. Secara default, Algoritma Canny menentukan sigma satu dan bisa kita lihat hasilnya, secara default Canny akan mendeteksi edge lebih banyak dibandingkan dengan hyperparameter sigma > 1.

VI. Corner Detection

Dekripsi sudut adalah satu metode untuk menekstrak beberapa fitur sudut didalam gambar. Corner detection sering kali digunakan pada projek-projek motion detection, video tracking, 3D modelling dan object detection. Fitur adalah point of interest yang menyediakan informasi yang sangat bermanfaat dalam computer vision. Fitur ini bisa berupa nilai rotasi, translasi, intensitas, perubahan skala, sampai ke nilai lokasi sudut. Sudut bisa diartikan sebagai pertemuan dari dua tepi / edges. Untuk mendekripsi corner kita bisa menggunakan Harris corner Algorithm. Untuk menggunakan algoritma ini, kita perlu mentransformasi gambar kita dari RGB ke grayscale.

```
sudut_gedung = plt.imread('../images/corners_building_top.jpg')

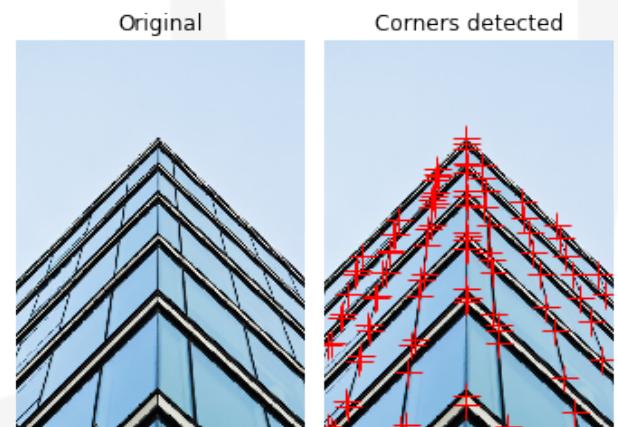
# Import the corner detector related functions and module
from skimage.feature import corner_harris, corner_peaks

# Convert image from RGB-3 to grayscale
building_image_gray = color.rgb2gray(sudut_gedung)

# Apply the detector to measure the possible corners
measure_image = corner_harris(building_image_gray)

# Find the peaks of the corners
coords = corner_peaks(measure_image, min_distance=2)

# Show original and resulting image with corners detected
show_image(sudut_gedung, "Original")
show_image_with_corners(sudut_gedung, coords)
```



VII. Face Detection

Beberapa tahun terakhir, teknologi Face Detection berkembang pesat dan telah menyita perhatian dunia seiring dengan semakin populernya teknologi Deep Learning. Dimana computational power semakin berkembang, penemuan algoritma-algoritma Machine Learning yang semakin robust untuk menghandle data yang sangat besar, juga penemuan fungsi-fungsi pendukung seperti fungsi aktivasi seperti ReLu yang memungkinkan Machine Learning belajar lebih cepat dengan less computational power. Faktor-faktor ini jugalah yang membuat teknologi pada Computer Vision seperti Face Detection semakin advance.

Beberapa platform social media dan juga aplikasi pada smart-phones sudah menerapkan teknologi face detection, seperti facebook yang bisa mengenali wajah dari sebuah foto yang diupload dan memberikan rekomendasi tag. Face Detection juga sangat berguna untuk kasus yang lebih spesifik seperti emotion recognition pada facial expression dari sebuah wajah. Sebelum ke tahap emotion recognition, kita perlu menerapkan face detection terlebih dahulu.

Pada pembelajaran hari ini, kita akan menggunakan Cascade classifier algorithm. Algoritma ini bisa dipanggil dari feature module pada scikit-image dan juga tersedia pada library lainnya seperti OpenCV. Cascade classifier adalah pre-trained algorithm yang berarti kita tidak perlu melakukan training dan bisa langsung melakukan deteksi wajah (prediction). Cascade classifier framework memerlukan xml file dimana seluruh data yang telah di training disimpan. Sebelum melakukan deteksi wajah, ada baiknya kita membuat fungsi pendukung untuk melakukan plotting garis dengan bentuk persegi.

```
from matplotlib import patches

def show_detected_face(result, detected, title="Face image"):
    plt.imshow(result)
    img_desc = plt.gca()
    plt.set_cmap('gray')
    plt.title(title)
    plt.axis('off')
    for patch in detected:
        img_desc.add_patch(
            patches.Rectangle(
                (patch['c'], patch['r']),
                patch['width'],
                patch['height'],
                fill=False, color='r', linewidth=2))
    plt.show()
```

Face image

```

from skimage import data
from skimage.feature import Cascade

gambar_wajah = plt.imread('../images/face_det3.jpg')

# Load the trained file from data
trained_file = data.lbp_frontal_face_cascade_filename()

# Initialize the detector cascade
detector = Cascade(trained_file)

# Detect faces with min and max size of searching window
detected = detector.detect_multi_scale(img = gambar_wajah,
                                         scale_factor=1.2,
                                         step_ratio=1,
                                         min_size=(10, 10),
                                         max_size=(200, 200))

# Show the detected faces
show_detected_face(gambar_wajah, detected)
    
```



9) References

- <https://docs.python.org/3/library/re.html>
- https://www.w3schools.com/python/python_regex.asp
- https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
- https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
- <https://medium.com/@cmukesh8688/tf-idf-vectorizer-scikit-learn-dbc0244a911a>
- <https://www.nltk.org/>
- <https://realpython.com/train-test-split-python-data/>
- <https://scikit-learn.org/stable/modules/svm.html>
- <https://www.nltk.org/api/nltk.stem.html>
- <https://towardsdatascience.com/how-to-use-nlp-in-python-a-practical-step-by-step-example-bd82ca2d2e1e>
- https://scikit-image.org/docs/dev/auto_examples/applications/plot_face_detection.html
- <https://campus.datacamp.com/courses/image-processing-in-python/advanced-operations-detecting-faces-and-features?ex=7>
- https://scikit-image.org/docs/dev/auto_examples/segmentation/plot_thresholding.html
- <https://scikit-image.org/docs/dev/api/skimage.filters.html>
- https://scikit-image.org/docs/dev/api/skimage.filters.rank.html?highlight=contrast#skimage.filters.rank.enhance_contrast