filkom.ub.id

DIGITAL TALENT SCHOLARSHIP 2019

Big Data Analytics

Visualisasi: Data Tidak Terstruktur (Mix)

Oleh: Imam Cholissodin | imamcs@ub.ac.id, Putra Pandu Adikara, Sufia Adha Putri
Asisten: Guedho, Sukma, Anshori, Aang dan Gusti
Fakultas Ilmu Komputer (Filkom) Universitas Brawijaya (UB)

digitalent.kominfo.go.id

Pokok Pembahasan

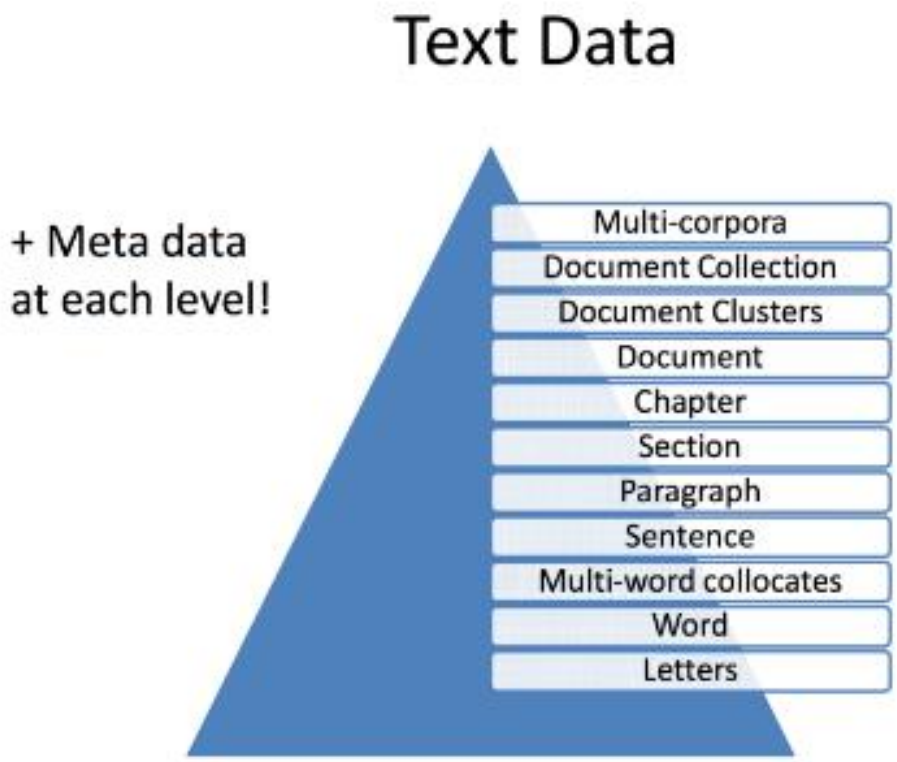
- Lecture’s Objective
- Data visualization
- Use case for data visualization
- Tugas

Lecture's Objective

- Mempelajari cara penjelasan **analisis** statistika kualitatif dan lebih luas **data tidak terstruktur** (**berbasis text**) **menggunakan gambar** untuk mempermudah pemahaman.
- Setelah mengikuti sesi ini, peserta dapat mengetahui cara memvisualisasi hasil analisis dari data teks atau numerik kualitatif dan menerapkan pada kasus yang sesuai.

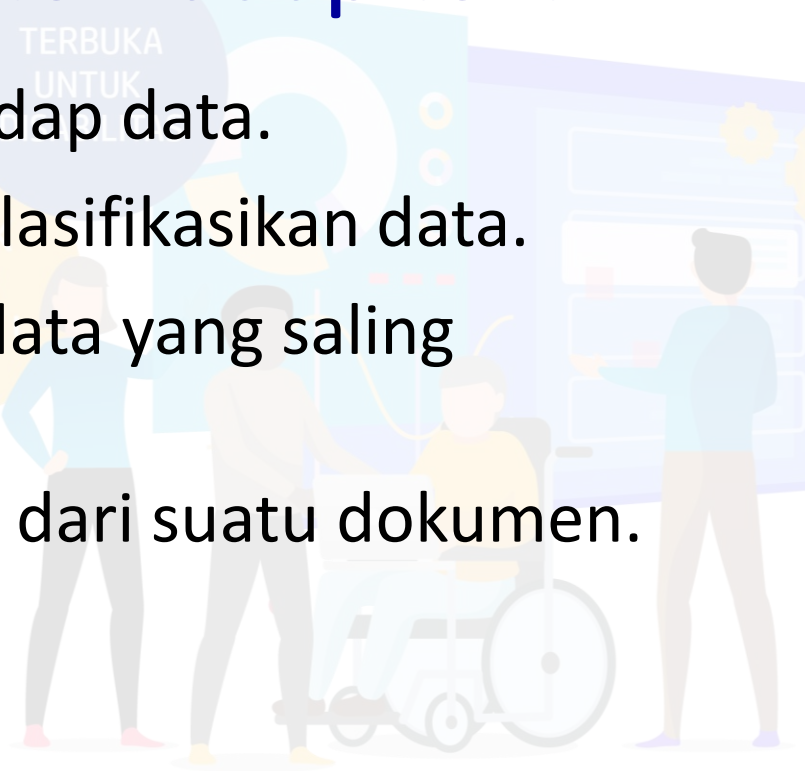
Text Analysis & Visualization

- Visualisasi ini digunakan untuk menganalisis data tidak terstruktur (berbasis teks)



Apa yang dapat dilakukan dengan visualisasi dan analisis terhadap text?

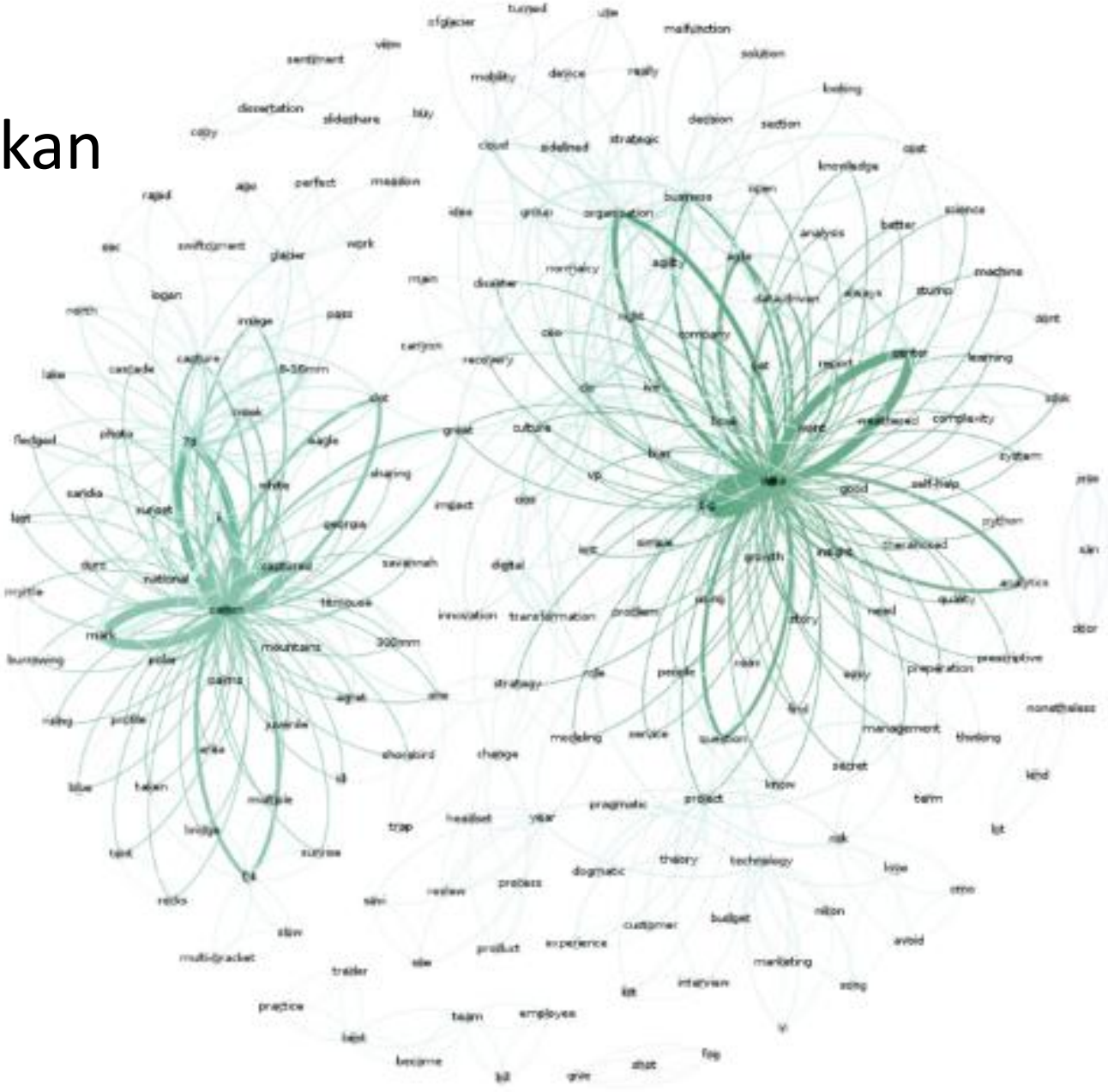
- Mempermudah analisis terhadap data.
- Mengelompokkan dan mengklasifikasikan data.
- Memahami dan memetakan data yang saling berhubungan
- Mencari konten yang menarik dari suatu dokumen.



- Contoh Dokumen yang akan di analisis

	id	Title	Content	Date	Permalink	Categories
0	8938	Building a Data Culture	<a href="http://ericbrown.com/wp-content/uploa...	20141118	http://ericbrown.com/building-data-culture.htm	Big Data|People
1	8943	Note to Self - Don't say "Data Driven" Anymore	<a href="http://ericbrown.com/wp-content/uploa...	20141120	http://ericbrown.com/dont-say-data-driven-anym...	Big Data|Leadership
2	8948	Foto Friday - Titmouse on the Feeder	I captured this Titmouse on the backporch feed...	20141121	http://ericbrown.com/foto-friday-titmouse-feed...	Foto Friday
3	8952	The Cloud - Gateway to Enterprise Mobility	This post is brought to you by the <a...	20141121	http://ericbrown.com/cloud-gateway-enterprise-...	Information Technology|Strategy|Technology|The...
4	8957	The Agile Data Center	<a href="http://ericbrown.com/wp-content/uploa...	20141124	http://ericbrown.com/agile-data-center.htm	Data Center|Information Technology|The New CIO

Graph yang merepresentasikan text analysis



Contoh Text Analysis

Hasil Text Analysis & Visualization untuk mendapatkan dan **mencari keywords** dari setiap data yang tersedia

	keywords	title	pubdate
0	data,big,culture	Building a Data Culture	20141118
1	data,data-driven,company	Note to Self - Don't say "Data Driven" Anymore	20141120
2	canon,captured,titmouse	Foto Friday - Titmouse on the Feeder	20141121
3	mobility,organization,device	The Cloud - Gateway to Enterprise Mobility	20141121
4	center,data,agile	The Agile Data Center	20141124

Pengukuran

- Ada beberapa unit dasar pengukuran yang dapat kita kumpulkan
- Pengukuran paling dasar adalah jumlah dari setiap kata

13	the	2	for	1	seen	1	hurried	1	after
11	it	2	dear	1	seemed	1	hot	1	actually
9	to	2	conversations	1	see	1	hole	1	'without
8	of	2	by	1	say	1	hedge	1	'oh
8	her	2	book	1	remarkable	1	hear	1	'and
8	a	2	be	1	reading	1	having		
7	she	2	as	1	quite	1	have		
7	and	2	across	1	pop	1	getting		
5	was	1	would	1	pleasure	1	get		
5	rabbit	1	worth	1	pink	1	fortunately		
4	very	1	wondered	1	picking	1	flashed		
4	or	1	white	1	peeped	1	field		
4	in	1	whether	1	own	1	feet		
4	alice	1	what	1	over	1	feel		
3	with	1	well	1	ought	1	eyes		
3	when	1	way	1	once	1	either		
3	that	1	use	1	oh	1	down		
3	so	1	up	1	occurred	1	do		
3	out	1	under	1	nor	1	did		
3	had	1	twice	1	no	1	day		
3	but	1	trouble	1	never	1	daisy		
3	at	1	took	1	natural	1	daisies		
3	'	1	tired	1	much	1	curiosity		
2	watch	1	this	1	making	1	could		
2	waistcoat	1	think	1	made	1	considering		
2	time	1	there	1	looked	1	close		
2	thought	1	then	1	late	1	chain		
2	sister	1	take	1	large	1	burning		
2	ran	1	suddenly	1	just	1	beginning		
2	pocket	1	stupid	1	itself	1	before		
2	pictures	1	started	1	its	1	bank		
2	on	1	sleepy	1	is	1	all		
2	nothing	1	sitting	1	into	1	afterwards		
2	mind	1	shall	1	i	1	after		

```
import nltk
from collections import Counter

tokens = nltk.word_tokenize(text)
counts = Counter(tokens)
sorted_counts = sorted(counts.items(), key=lambda count:
count[1], reverse=True)

sorted_counts
```

```
[(',', 2418),
 ('the', 1516),
 ("'", 1129),
 ('.', 975),
 ('and', 757),
 ('to', 717),
 ('a', 612),
 ('it', 513),
 ('she', 507),
 ('of', 496),
 ('said', 456),
 ('!', 450),
 ('Alice', 394),
 ('I', 374),...]
```

Dengan menggunakan Python, package nltk, dan Counter, kita dapat melakukan tokenisasi dan menghitung banyak kemunculan setiap kata

Tokenisasi

- Memecahkan teks menjadi sebuah daftar dari kata

```
pride.txt:

Mrs. Bennet deigned not to make any reply,
but, unable to contain herself,
began scolding one of her daughters.
```

```
Mrs.
Bennet
deigned
not
to
make
any
reply
```

Clean-up

- Rutin clean-up yang selalu berguna ketika bekerja dengan teks walaupun kita tidak akan selalu menggunakan semua rutinnya, tergantung dengan proses yang dibutuhkan.
- Berikut adalah beberapa rutin clean-up:
  - Remove punctuation ('!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~-')
  - Remove stop word (seperti “the”, “a”, “an”, “in”)
  - Normalize the case (menormalisasi menjadi huruf kecil
  - Remove fragments (seperti apostrophe n’t, ‘s)
  - Stemming (menjadi kata dasar)

Remove Punctuation

- Python memiliki class String yang memungkinkan kita untuk mengambil kumpulan pungtuasi
- Pungtuasi seperti *'!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~-'*

```
from string import punctuation

def remove_tokens(tokens, remove_tokens):
    return [token for token in tokens if token not in
remove_tokens]

no_punc_tokens = remove_tokens(tokens, punctuation)
```

Remove Stop Word

- Stop word adalah kata kata (seperti “the”, “a”, “an”, “in”) yang search engine telah di program untuk mengabaikan, baik ketika mencari dari dan menampilkan hasil pencarian
- Biasanya tidak menghasilkan sesuatu yang berarti dan berhubungan dengan konteks

```
from nltk.corpus import stopwords
stops = stopwords.words('english')

# stop words look like:
# [u'i', u'my', u'myself', u'we', u'our',
# u'ours', u'you'...]
filtered_tokens = remove_tokens(no_punc_tokens, stops)
```

Normalize Case

- Membuat semua kata menggunakan huruf kecil
- Karena bisa saja terdapat dua kata yang sama, namun peda penulisan huruf kecil, maka akan dianggap berbeda
- Kita ingin agar kata tersebut dianggap sama, sehingga kita dapat menormalisasi menjadi huruf kecil semua

```
def lowercase(tokens):
    return [token.lower() for token in tokens]

lowercase(no_punc_tokens)
```

Remove Fragments

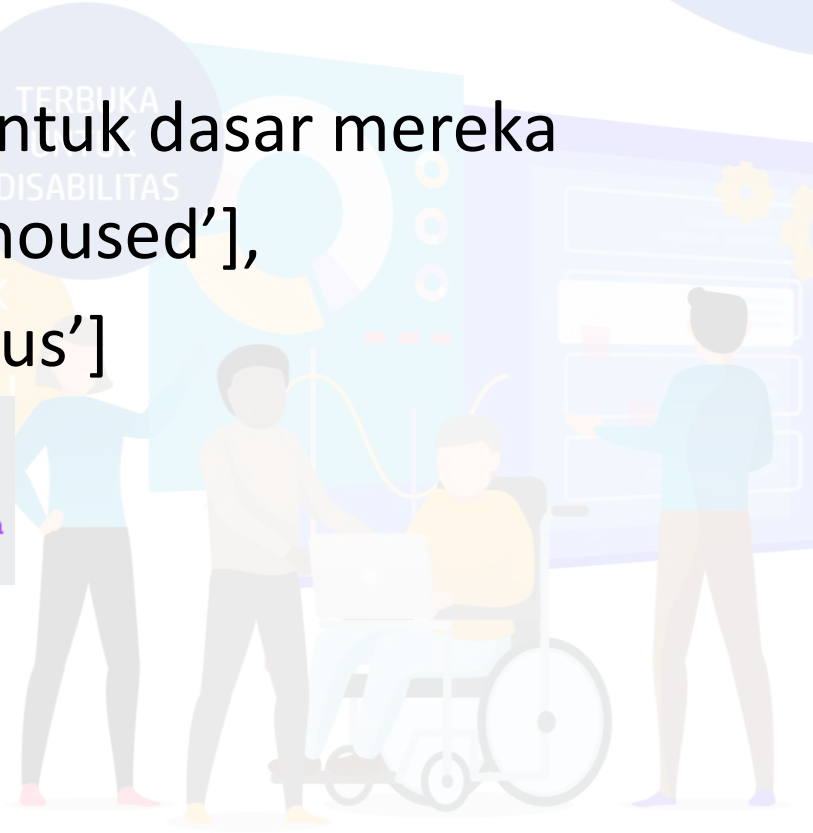
- Seperti penggunaan apostrophe
- Seperti n’t, ‘s

```
def remove_word_fragments(tokens):
    return [token for token in tokens if "'" not in token]

no_frag_tokens = remove_word_fragments(filtered_tokens)
```

Stemming

- Konversi kata kata menjadi bentuk dasar mereka
- regular = ['house', 'housing', 'housed’],
- stemmed = ['hous', 'hous', 'hous’]

```
from nltk.stem import PorterStemmer
stemmer = PorterStemmer()

stemmed_tokens = [stemmer.stem(token) for token in
no_frag_tokens]
```



Setelah melakukan proses clean-up, hasil perhitungan menjadi:

```
[('said', 462),
 ('alic', 396),
 ('littl', 128),
 ('look', 102),
 ('one', 100),
 ('like', 97),
 ('know', 91),
 ('would', 90),
 ('could', 86),
 ('went', 83),
 ('thought', 80),
 ('thing', 79),
 ('queen', 76),
 ('go', 75),
 ('time', 74),
 ('say', 70),
 ('see', 68),
 ('get', 66),
 ('king', 64),...]
```

```
[(',', 2418),
 ('the', 1516),
 ("'", 1129),
 ('.', 975),
 ('and', 757),
 ('to', 717),
 ('a', 612),
 ('it', 513),
 ('she', 507),
 ('of', 496),
 ('said', 456),
 ('!', 450),
 ('Alice', 394),
 ('I', 374),
 ('was', 362),
 ('in', 351),
 ('you', 337),
 ('that', 267),
 ('--', 264),...]
```

before

HUBUNGAN KATA

- Kata bermunculan bersamaan, mereka pasti memiliki sejenis hubungan antara satu dengan yang lain
- Beberapa teknik yang dapat digunakan: Concordance, N-Grams, Co-occurence

Concordance

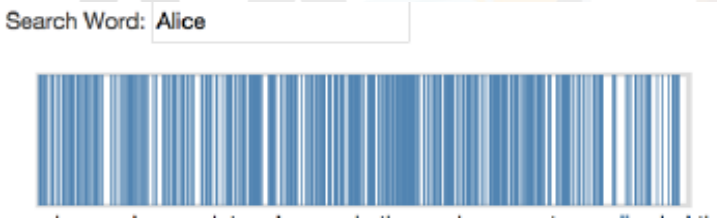
- Memungkinkan kita untuk melihat keyword pada konteks
- Nltk memiliki memiliki class Text yang akan merubah token menjadi sebuah searchable object untuk dapat dimanipulasi

```
my_text = nltk.Text(tokens)
my_text.concordance('Alice')
```

```
                                 Alice was beginning to get very tired of s
hat is the use of a book , ' thought Alice 'without pictures or conversations ?
so VERY remarkable in that ; nor did Alice think it so VERY much out of the way
looked at it , and then hurried on , Alice started to her feet , for it flashed
 hedge . In another moment down went Alice after it , never once considering ho
ped suddenly down , so suddenly that Alice had not a moment to think about stop
she fell past it . 'Well ! ' thought Alice to herself , 'after such a fall as t
own , I think -- ' ( for , you see , Alice had learnt several things of this so
tude or Longitude I 've got to ? ' ( Alice had no idea what Latitude was , or L
 . There was nothing else to do , so Alice soon began talking again . 'Dinah 'l
ats eat bats , I wonder ? ' And here Alice began to get rather sleepy , and wen
dry leaves , and the fall was over . Alice was not a bit hurt , and she jumped
 not a moment to be lost : away went Alice like the wind , and was just in time
 but they were all locked ; and when Alice had been all the way down one side a
```

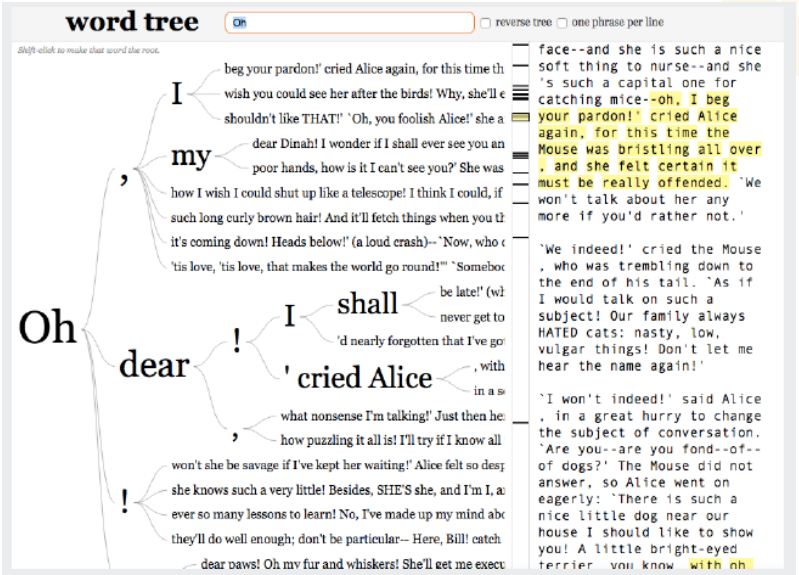
Concordance Plots

- Memberitahukan dimana suatu konteks ditemukan pada teks
- Persegi panjang merepresentasikan teks, dan garis vertical menunjukkan dimana suatu konteks ditemukan pada teks dokumen



Concordance: Word Tree

- Melakukan hal yang sama seperti yang dilakukan concordance, namun dia menggabungkan kata berikut yang sama, seperti membuat hirarki teks.



N-Grams (Collocation)

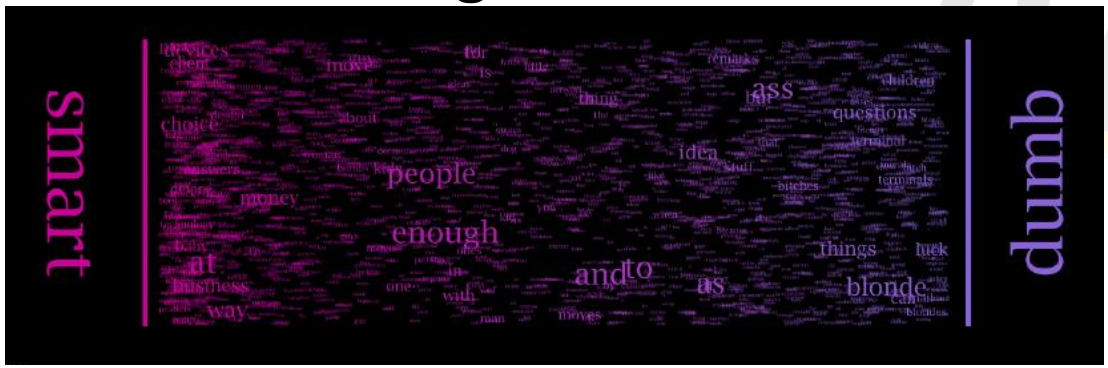
- Melihat satu kata tetapi melihat dua atau tiga atau empat kata bisa lebih informatif

```
from nltk.collocations import BigramCollocationFinder
finder = BigramCollocationFinder.from_words(filtered_tokens)

# built in bigram metrics are in here
bigram_measures = nltk.collocations.BigramAssocMeasures()

# we call score_ngrams on the finder to produce a sorted list
# of bigrams. Each comes with its score from the metric, which
# is how they are sorted.
finder.score_ngrams(bigram_measures.raw_freq)
```

Kumpulan kata yang lebih sering muncul bersama

```
[(('said', 'the'), 0.007674512539933169),
 (('of', 'the'), 0.004700179928762898),
 (('said', 'alice'), 0.004259538060441376),
 (('in', 'a'), 0.0035618551022656335),
 (('and', 'the'), 0.002900892299783351),...]
```

- Visualisasi Google Bi-Gram data



- Daripada menggunakan raw_freq, dengan menggunakan likelihood_ratio dapat menghasilkan kombinasi yang lebih menarik

```
finder.score_ngrams(bigram_measures.likelihood_ratio)
```

```
[(('mock', 'turtle'), 781.0917141765854),
 (('said', 'the'), 597.9581706687363),
 (('said', 'alice'), 505.46971076855675),
 (('march', 'hare'), 461.91931122768904),
 (('went', 'on'), 376.6417465508724),
 (('do', "n't"), 372.7029564560615),
 (('the', 'queen'), 351.39319634691446),
 (('the', 'king'), 342.27277302768084),
 (('in', 'a'), 341.4084817025905),
 (('the', 'gryphon'), 278.40108569878106),...]
```

Co-occurence

- Kita telah berbicara tentang kata-kata yang berdekatan satu sama lain tetapi kata-kata juga dapat dipisahkan dalam semacam pola yang mungkin sering muncul

```
my_text = nltk.Text(tokens)
my_text.findall('<.*><of><.*>')
```

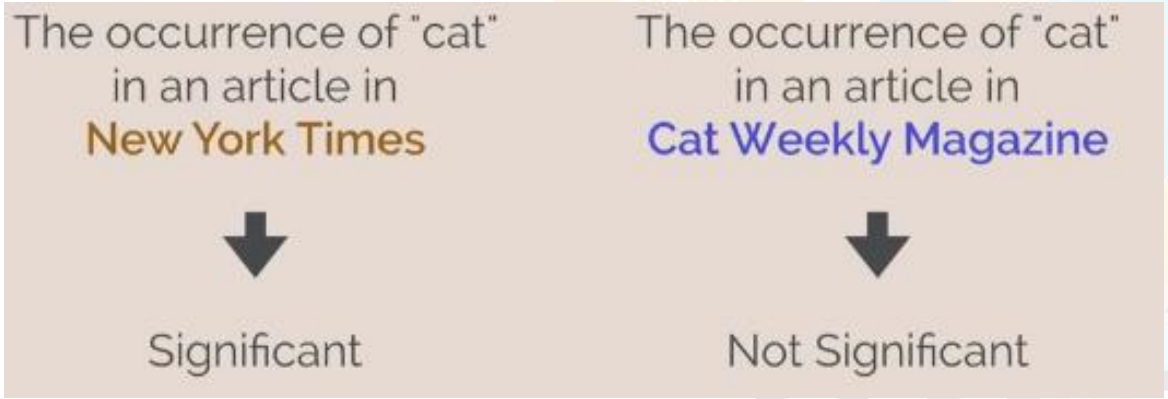
```
tired of sitting; and of having; use of a; pleasure of making; trouble
of getting; out of the; out of it; plenty of time; sides of the; one
of the; fear of killing; one of the; nothing of tumbling; top of the;
centre of the; things of this; name of the; saucer of milk; sort of
way; heap of sticks; row of lamps; made of solid; one of the; doors of
the; any of them; out of that; beds of bright; be of very; book of
rules; neck of the; sort of mixed; flavour of cherry-tart; flame of a;
one of the; legs of the; game of croquet; fond of pretending; enough
of me; top of her; way of expecting; Pool of Tears; out of sight; pair
of boots; roof of the; ashamed of yourself; gallons of tears;
pattering of feet; pair of white; help of any; were of the; any of
them; sorts of things; capital of Paris; capital of Rome; waters of
the; burst of tears; tired of being; one of the; cause of this; number
of bathing; row of lodging; pool of tears; be of any; out of this;
tired of swimming; way of speaking; -- of a; one of its; knowledge of
history; out of the; end of his; subject of conversation; -- of --;
```

KUMPULAN DARI DOKUMEN

- Seperti email, artikel berita, publikasi, dll
- Biasanya kita akan mencoba mencari tahu apa yang dibicarakan oleh kumpulan tersebut secara keseluruhan, apakah ada pola tertentu atau terdapat pengelompokkan dokumen-dokumen ini dan bagaimana mereka dibandingkan satu sama lain

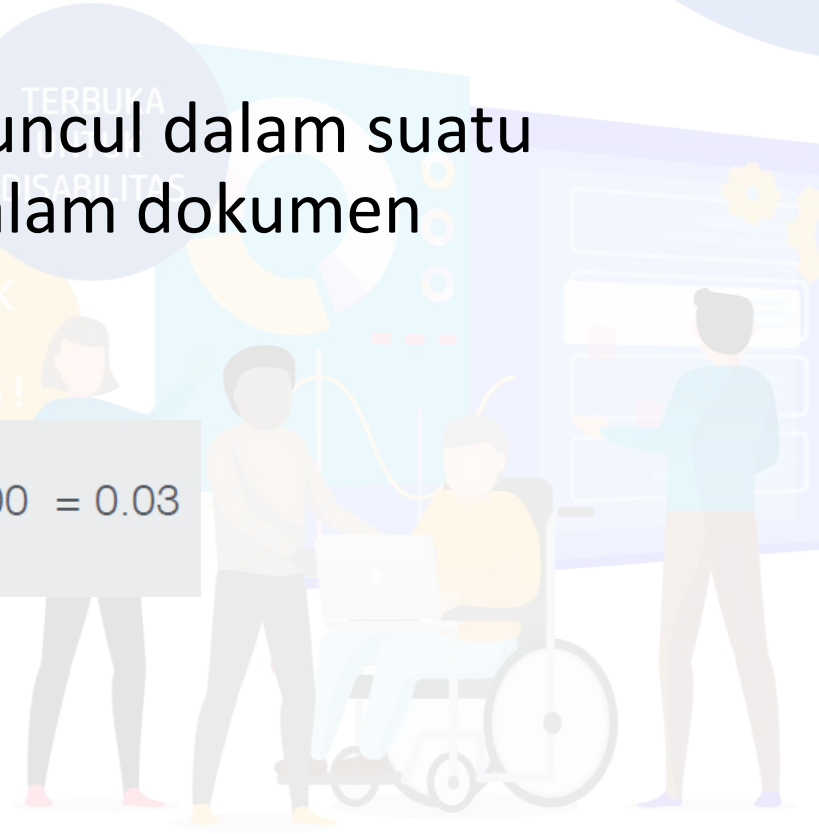
TF-IDF (Term Frequency-Inverse Document Frequency)

- Menemukan signifikansi dari kata pada dokumen



Term Frequency (TF)

- TF = Berapa kali suatu kata muncul dalam suatu dokumen / banyaknya kata dalam dokumen

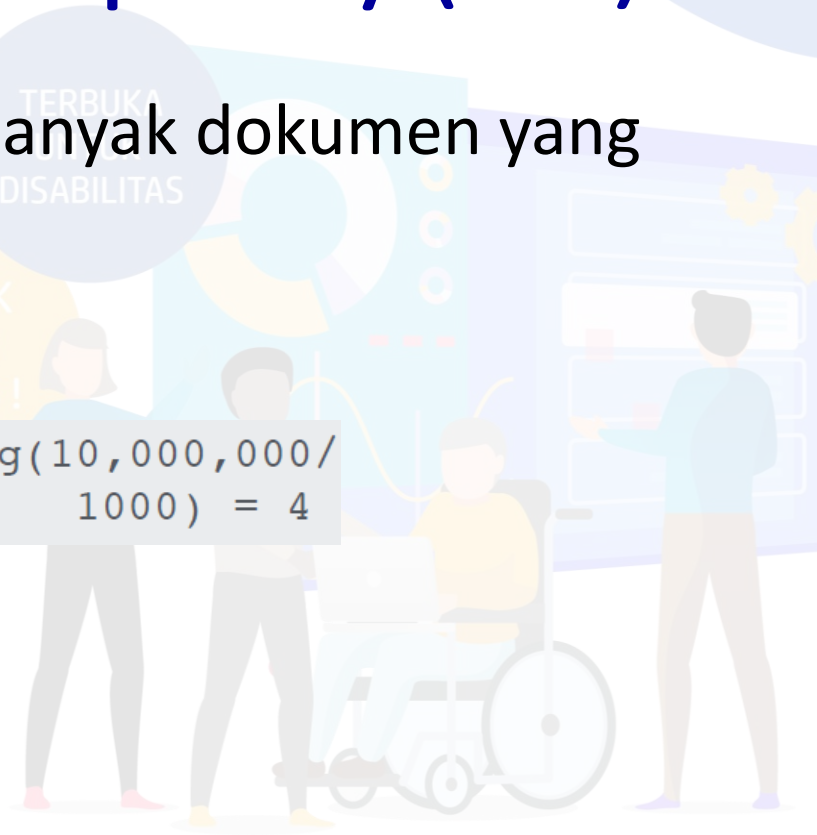


1 document
100 words 3 / 100 = 0.03
3 "cat"

Inverse Document Frequency (IDF)

- IDF = log(banyak dokumen / banyak dokumen yang memiliki kata tersebut)

10 million documents 1000 containing "cat"	log(10,000,000/ 1000) = 4



- Jika saya memiliki 10.000 dokumen dan semuanya memiliki ‘cat’ didalamnya maka ‘cat’ tidak signifikan
- Jika hanya beberapa yang memiliki ‘cat’ maka nilainya akan bervariasi

10,000 documents, 100 documents containing "cat"

```
(3 / 100) * ln(10,000 / 100) = 0.13815510557964275
```

10,000 documents, 1 document containing "cat"

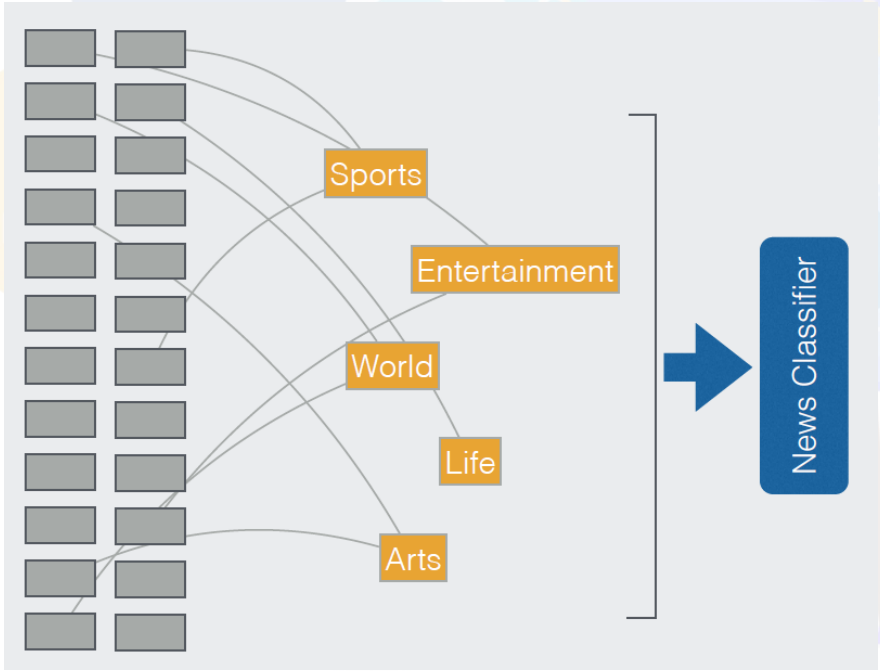
```
(3 / 100) * ln(10,000 / 1) = 0.2763102111592855
```

10,000 documents, all 10,000 documents containing "cat"

```
(3 / 100) * ln(10,000 / 10,000) = 0.0
```

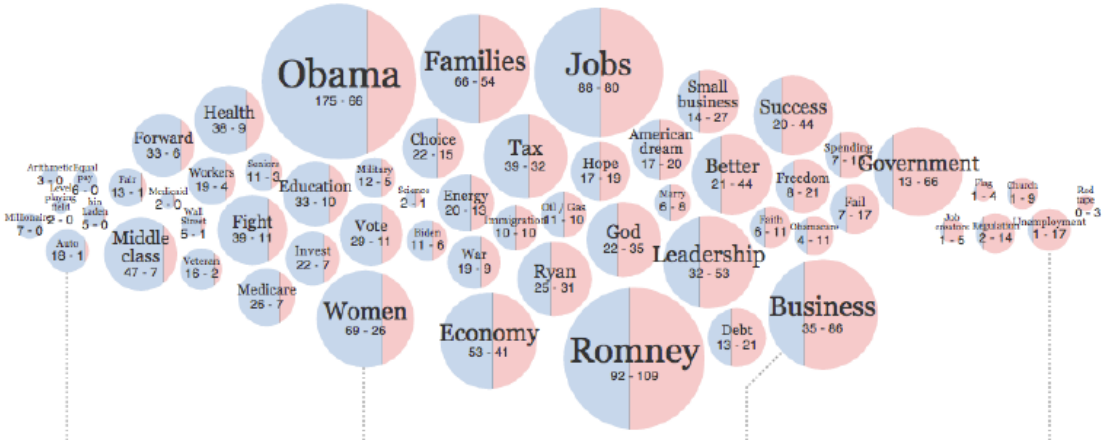
Pengelompokkan

- kita memiliki koleksi dari dokumen yang diberi label dengan bentuk tertentu
- Kotak abu-abu adalah artikel, dan memiliki suatu label

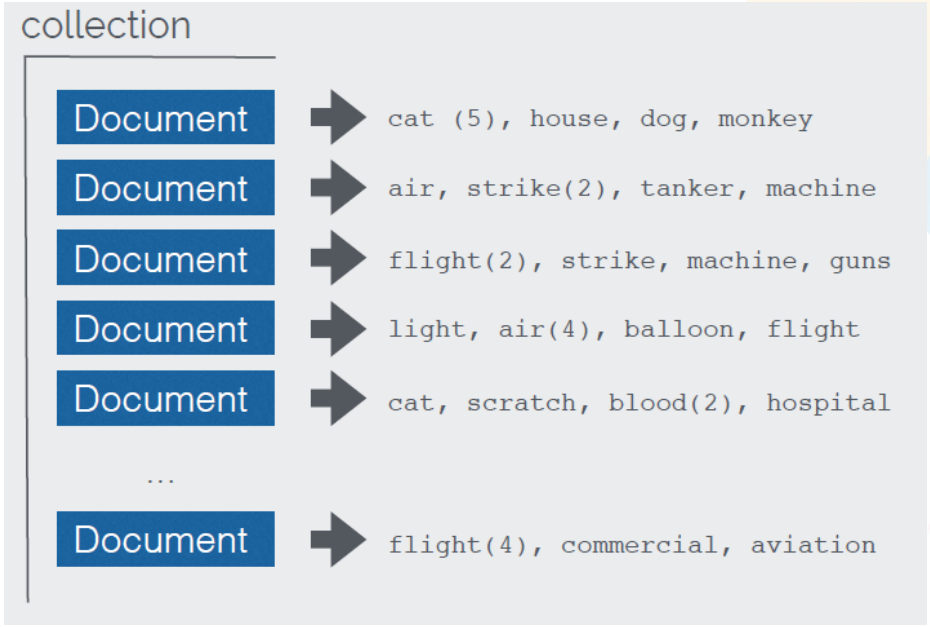


Perbandingan

- Ketika berkaitan dengan koleksi dari dokumen, hal yang harus dapat kita lakukan adalah **membandingkan mereka dengan berbagai cara**
- Pada level dasar, apabila kita hanya membandingkan dua dokumen maka kita hanya perlu **membandingkan** setiap kata dan **banyak masing masing kata** pada setiap dokumen



- Biasanya kita memiliki lebih banyak dokumen, yang dapat kita lakukan adalah menghitung kata-kata yang diinginkan pada masing-masing dokumen

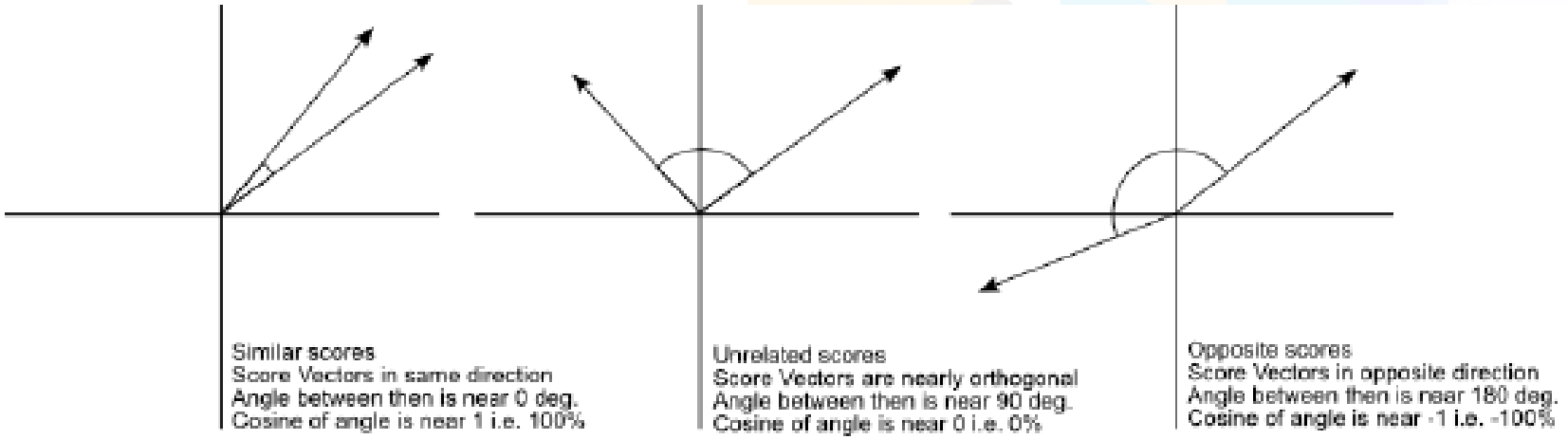


- Kemudian ubah menjadi vector untuk setiap dokumen

```
cat
house
dog
monkey
air
strike
tanker
machine
flight
guns
light
balloon
scratch
blood
hospital
commercial
aviation
```

```
[5,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0]

[0,0,0,0,1,2,1,1,0,0,0,0,0,0,0,0]

[0,0,0,0,0,1,0,1,2,1,0,0,0,0,0,0]

...
```

- Terdapat 3 jenis hasil yang dapat dilihat yaitu mirip, tidak berhubungan dan berkebalikan



$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

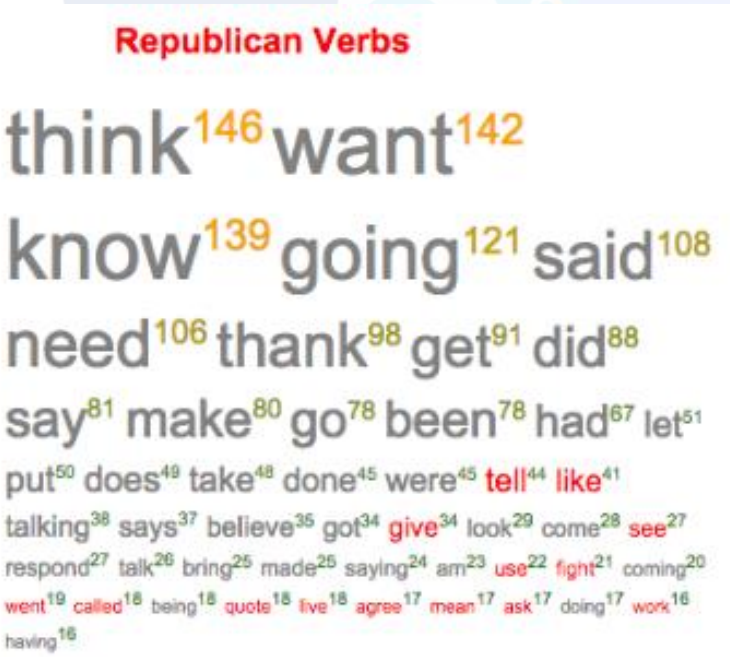
VISUALISASI TEKS

- Eksplorasi pemetaan visual

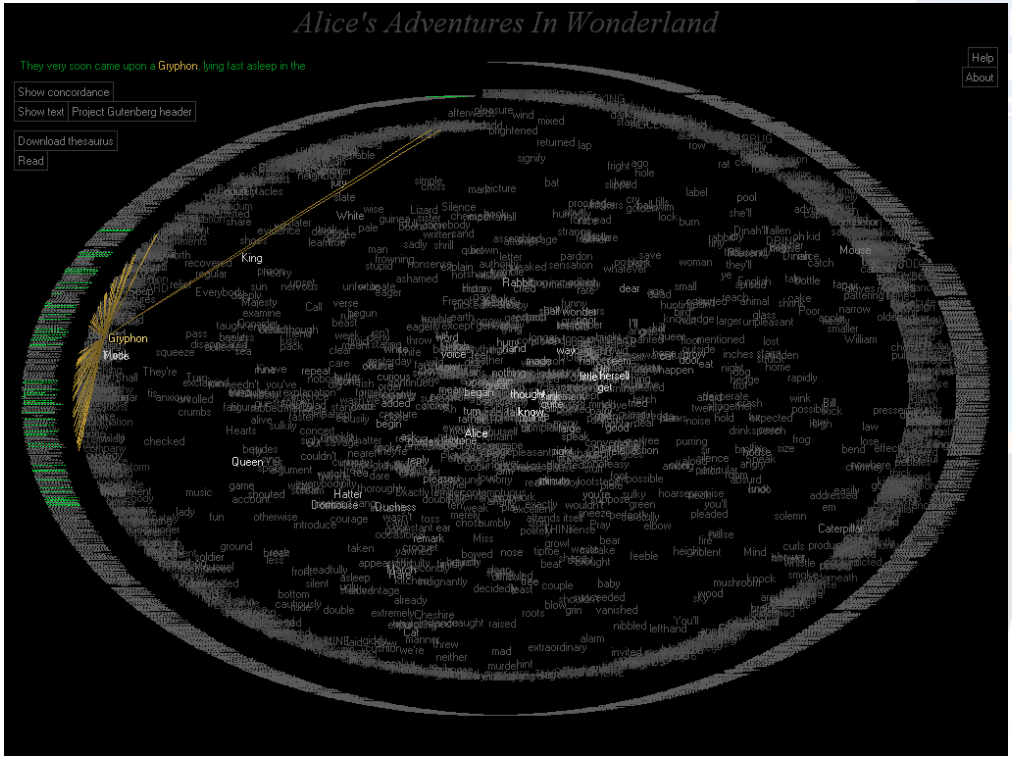
Ukuran : Wordcloud

Ukuran : Multiword Wordcloud

Ukuran : Analytical Wordcloud



Posisi : TextArc



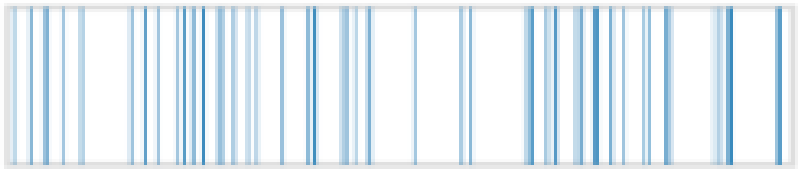
Struktur : Concordance Plot

Search Word: find

wherever you go to on the english coast you find a number of bati

alice_in_wonderland.txt

find was found 21 times.

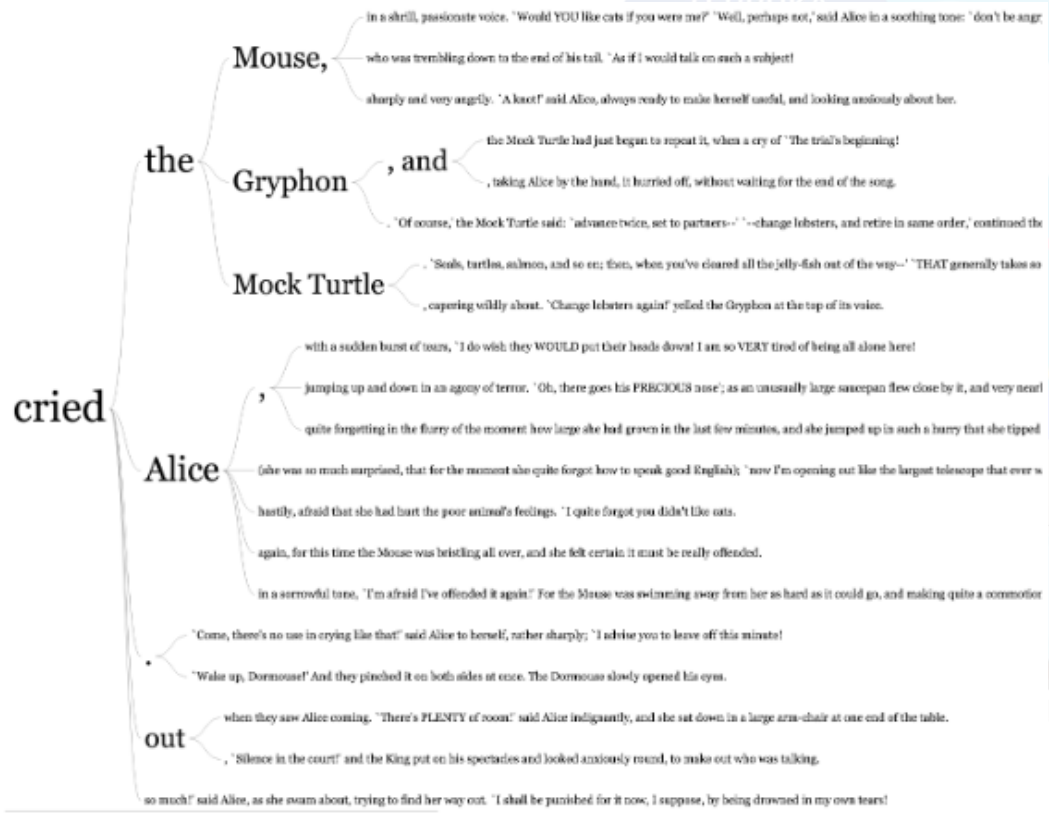


heaven I am going to morrow where I shall find a man who has no

pride_and_prejudice.txt

find was found 58 times.

Struktur : Word Tree



Waktu

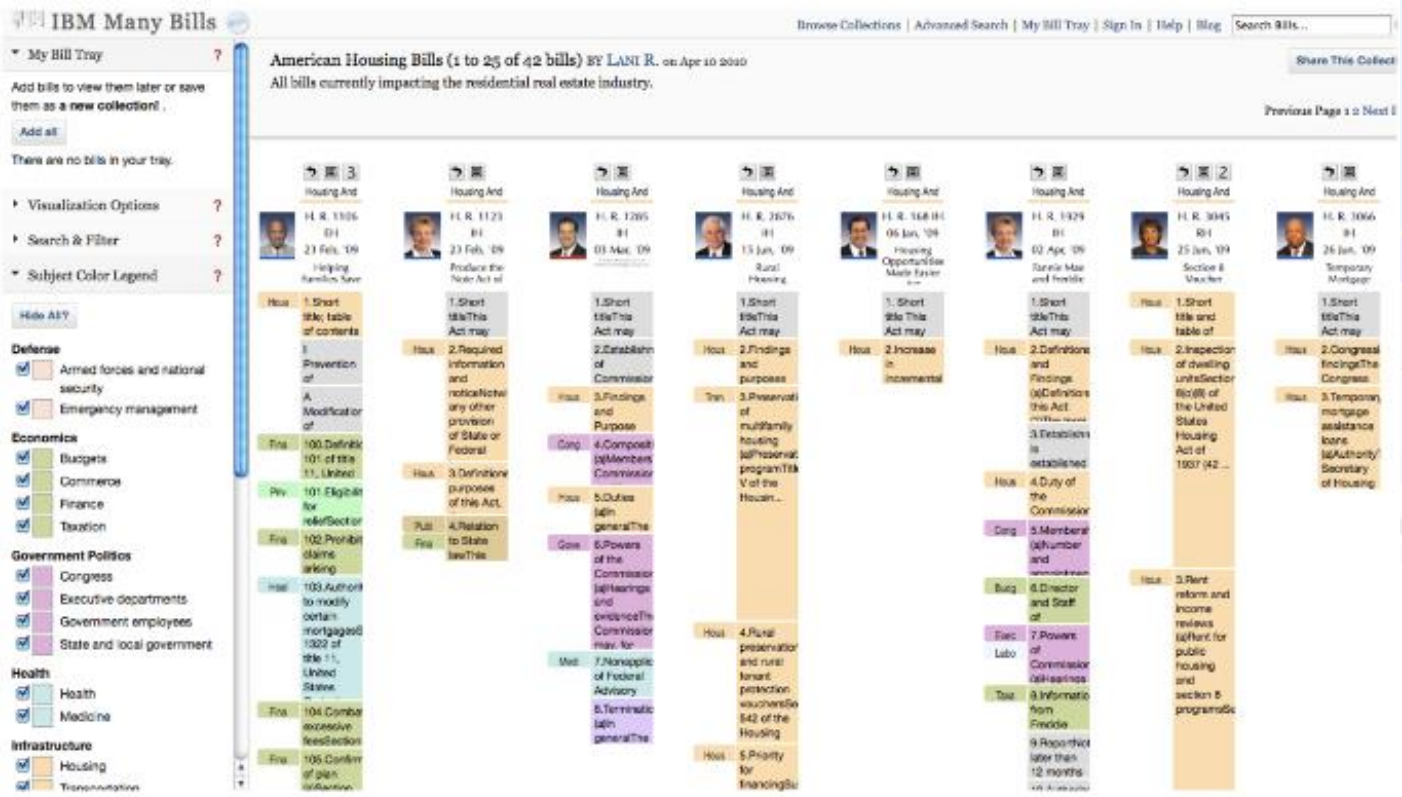

LOL
WIN
OMG
CUTE
FAIL
WTF
Enter terms
0.040000%
0.030000%
0.020000%
0.010000%
0.000000%
2008
2010
2012
2014
ZOOM
2008
2009
2010
2011
2012
2013
2014
2015

USE CASE

Use Case Visualization

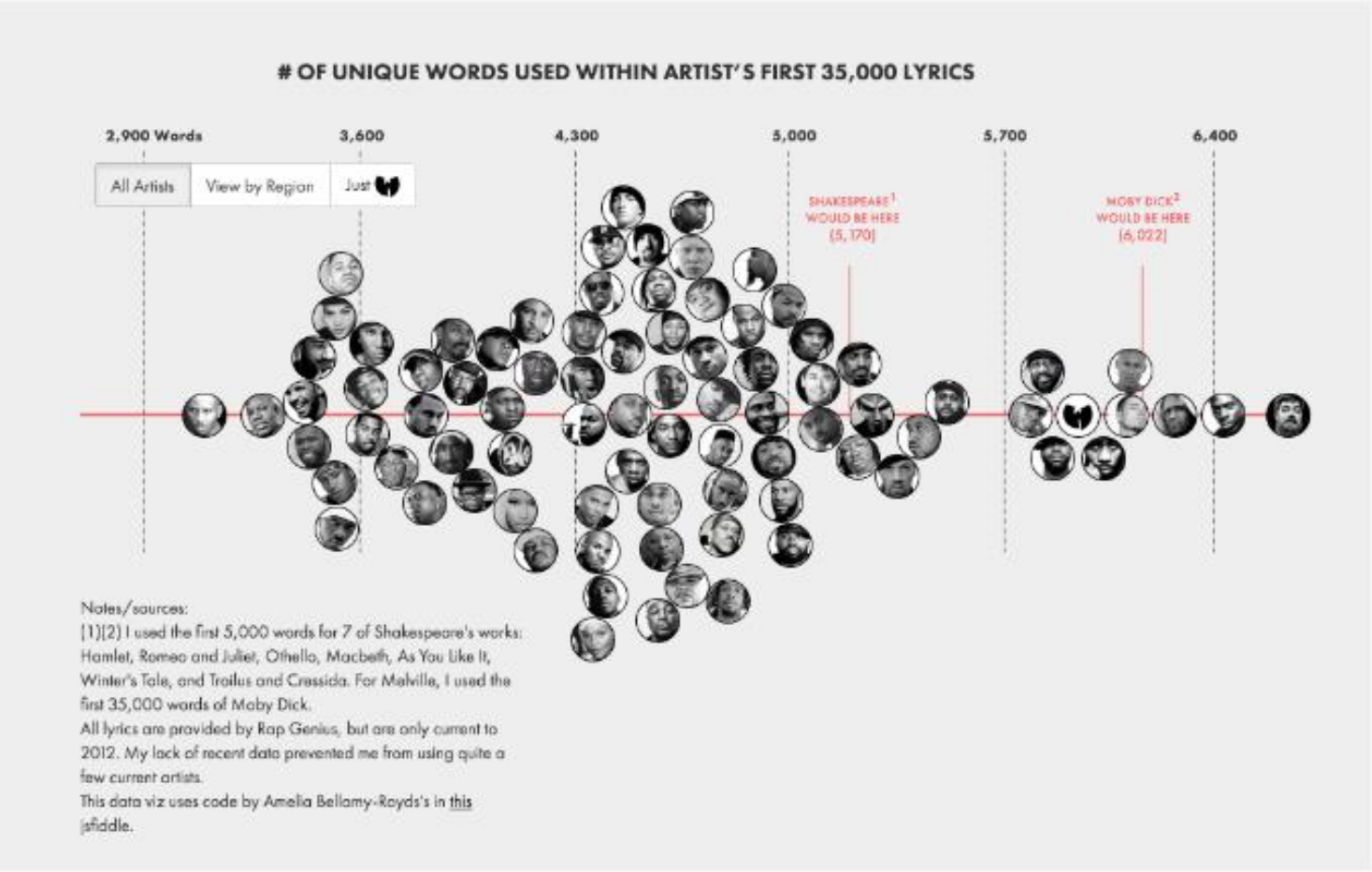
Many Bills

Merupakan produk dari IBM yang memberikan layanan data visualization sebagai visual interface untuk hukum atau hubungan legislatif



Use Case Visualization (1)

The Largest Vocabulary in Hip Hop



Contoh Source Code

Install nltk agar dapat melakukan tokenize terhadap data yang ditentukan

```
pip install bs4 nltk
```

```
import pandas as pd
import numpy as np

from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer, PorterStemmer
from string import punctuation
from collections import Counter

from collections import OrderedDict
import re
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)

from HTMLParser import HTMLParser
from bs4 import BeautifulSoup
```

```
porter = PorterStemmer()
wnl = WordNetLemmatizer()
stop = stopwords.words('english')
stop.append("new")
stop.append("like")
stop.append("u")
stop.append("it'")
stop.append("'s")
stop.append("n't")
stop.append('mr.')
stop = set(stop)
```

```
# From http://ahmedbesbes.com/how-to-mine-newsfeed-data-and-extract-interactive-insights-in-python.html

def tokenizer(text):

    tokens_ = [word_tokenize(sent) for sent in sent_tokenize(text)]

    tokens = []
    for token_by_sent in tokens_:
        tokens += token_by_sent

    tokens = list(filter(lambda t: t.lower() not in stop, tokens))
    tokens = list(filter(lambda t: t not in punctuation, tokens))
    tokens = list(filter(lambda t: t not in [u"'s", u"n't", u"...", u"''", u'``', u'\u2014', u'\u2026', u'\u2013'], tokens))

    filtered_tokens = []
    for token in tokens:
        token = wnl.lemmatize(token)
        if re.search('[a-zA-Z]', token):
            filtered_tokens.append(token)

    filtered_tokens = list(map(lambda token: token.lower(), filtered_tokens))

    return filtered_tokens
```

Contoh Source Code (2)

```
class MLStripper(HTMLParser):
    def __init__(self):
        self.reset()
        self.fed = []
    def handle_data(self, d):
        self.fed.append(d)
    def get_data(self):
        return ''.join(self.fed)

def strip_tags(html):
    s = MLStripper()
    s.feed(html)
    return s.get_data()
```

Untuk mendapatkan keyword dari data yang tersedia

```
def get_keywords(tokens, num):
    return Counter(tokens).most_common(num)
```

Contoh Source Code (3)

Menggunakan fungsi tokenizer untuk mendapatkan data yang diinginkan kemudian data tersebut dimasukan ke dalam tabel dengan kolom yang didefinisikan

```
def build_article_df(urls):
    articles = []
    for index, row in urls.iterrows():
        try:
            data=row['text'].strip().replace("'", "")
            data = strip_tags(data)
            soup = BeautifulSoup(data)
            data = soup.get_text()
            data = data.encode('ascii', 'ignore').decode('ascii')
            document = tokenizer(data)
            top_5 = get_keywords(document, 5)

            unzipped = zip(*top_5)
            kw= list(unzipped[0])
            kw=",".join(str(x) for x in kw)
            articles.append((kw, row['title'], row['pubdate']))
        except Exception as e:
            print e
            #print data
            #break
            pass
        #break
    article_df = pd.DataFrame(articles, columns=['keywords', 'title', 'pubdate'])
    return article_df
```

Contoh Source Code (4)

Memanggil kembali tabel yang tedapat pada csv (https://github.com/urgedata/pythondata/blob/master/examples/tocsv.csv) dengan kolom yang ditentukan kemudian mengubah nama kolom tersebut

```
df = pd.read_csv('../examples/tocsv.csv')
data = []
for index, row in df.iterrows():
    data.append((row['Title'], row['Permalink'], row['Date'], row['Content']))
data_df = pd.DataFrame(data, columns=['title' ,'url', 'pubdate', 'text' ])
```

```
data_df.tail()
```

Contoh Source Code (5)

Melakukan tokenize dan melakukan penhitungan kata dengan menggunakan fungsi build_article_df

```
article_df = build_article_df(data_df)
```

Memisahkan keyword terhadap keyword yang lain untuk setiap barisnya sehingga setiap baris hanya terdiri dari satu keyword

```
keywords_array=[]
for index, row in article_df.iterrows():
    keywords=row['keywords'].split(',')
    for kw in keywords:
        keywords_array.append((kw.strip(' '), row['keywords']))
kw_df = pd.DataFrame(keywords_array).rename(columns={0:'keyword', 1:'keywords'})
```

Contoh Source Code (6)

Membentuk matriks yang berisi frekuensi dari setiap kata yang muncul pada data dan didapatkan 3 kata yang sering muncul yang dijadikan sebagai keyword

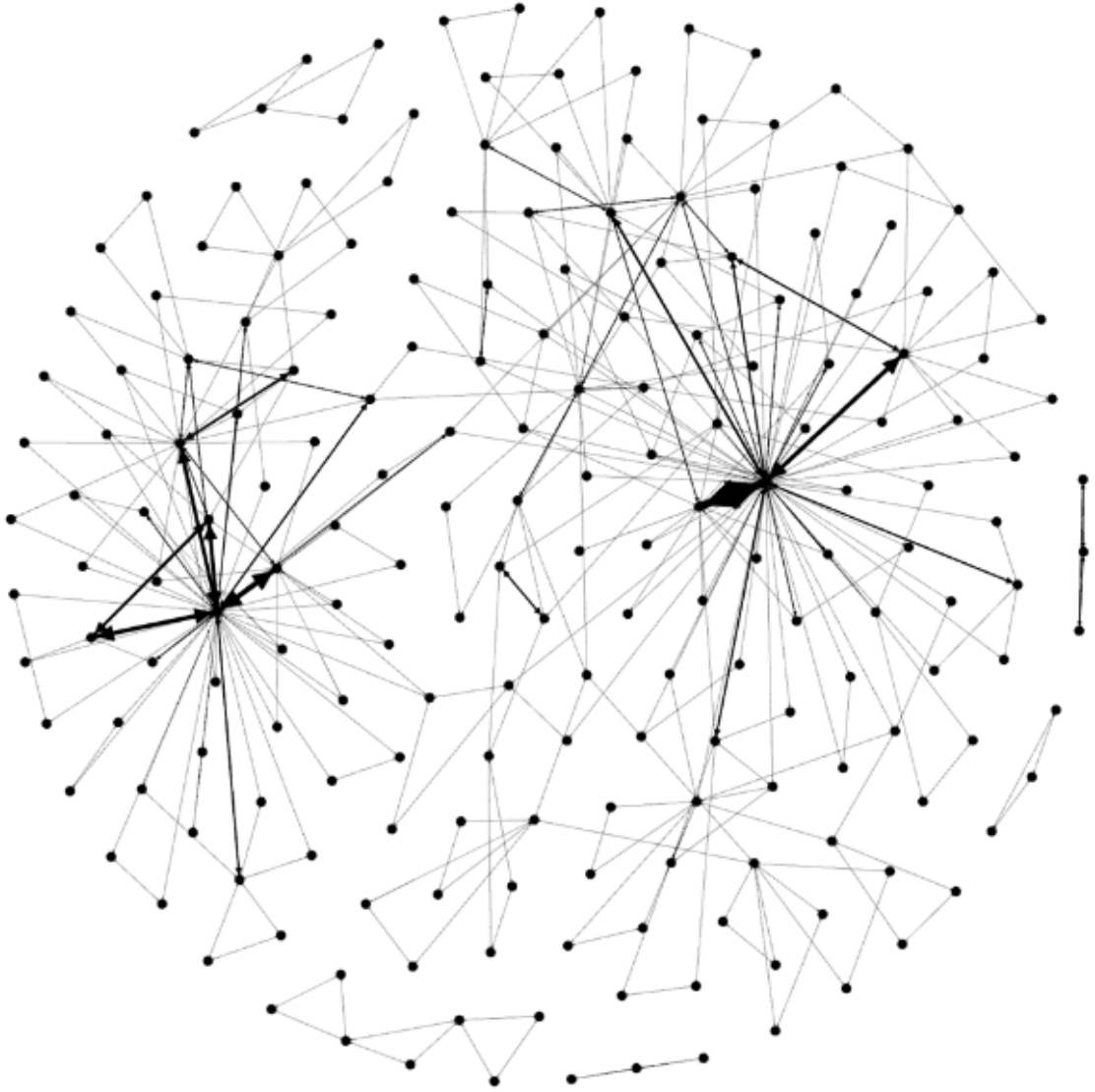
```
document = kw_df.keywords.tolist()
names = kw_df.keyword.tolist()

document_array = []
for item in document:
    items = item.split(',')
    document_array.append((items))

occurrences = OrderedDict((name, OrderedDict((name, 0) for name in names)) for name in names)

# Find the co-occurrences:
for l in document_array:
    for i in range(len(l)):
        for item in l[:i] + l[i + 1:]:
            occurrences[l[i]][item] += 1

co_occur = pd.DataFrame.from_dict(occurrences )
```

Contoh Source Code (6)

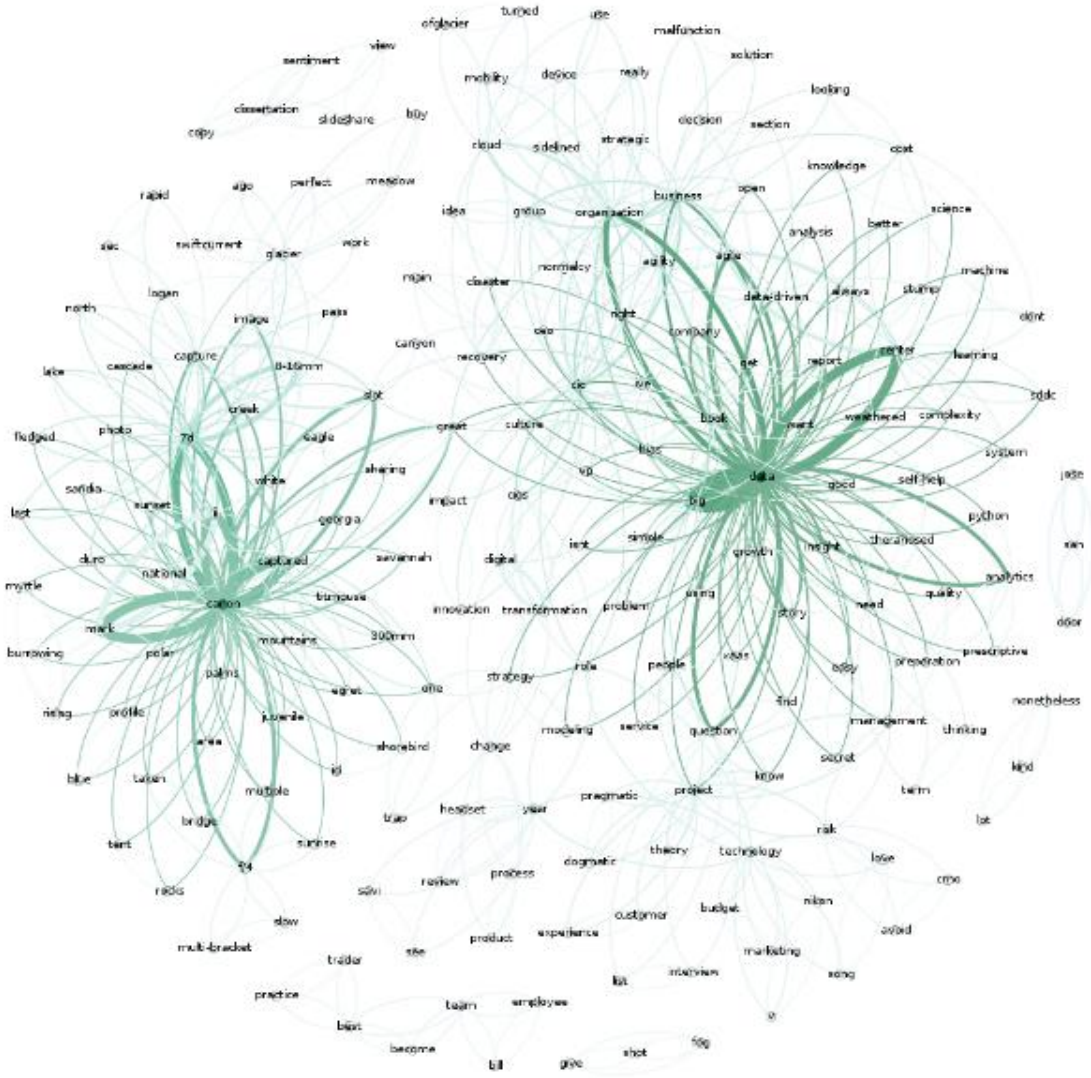
Menyimpan dataframe matriks (https://github.com/urgedata/pythondata/blob/master/Text%20Analytics/out/ericbrown_co-occurancy_matrix.csv) menjadi file CSV yang dapat digunakan untuk memvisualisasikan menjadi bentuk graph melalui software Gephi

```
1 co_occur.to_csv('out/ericbrown_co-occurancy_matrix.csv')
```

Hasil Visualisasi berupa Graph



Hasil Visualisasi berupa Graph (1)



Latihan langsung di Kelas Ke-1 & Pembahasan

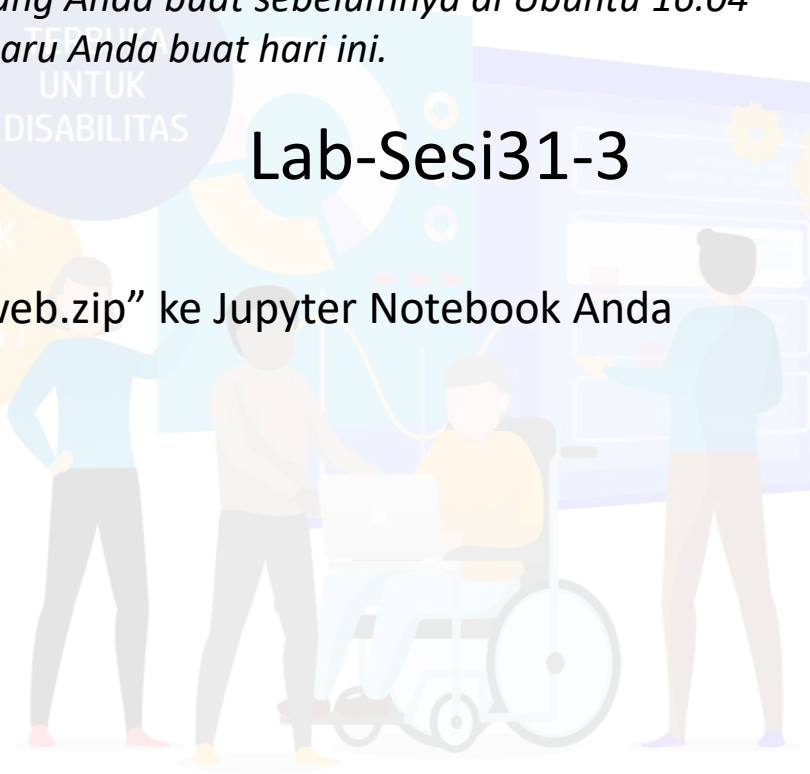
Link kode "http://bit.ly/2Z4RBxT"

Silahkan dicoba dijalankan dengan Jupyter notebook yang Anda buat sebelumnya di Ubuntu 16.04 atau dengan SageMaker notebook (JupyterLab) yang baru Anda buat hari ini.

Lab-Sesi31-1 Lab-Sesi31-2 Lab-Sesi31-3

Petunjuk mengerjakan Lab-Sesi31-1-2-3:

Pindahkan "draft koding python Pert. 31 dari web.zip" ke Jupyter Notebook Anda (*.ipynb)

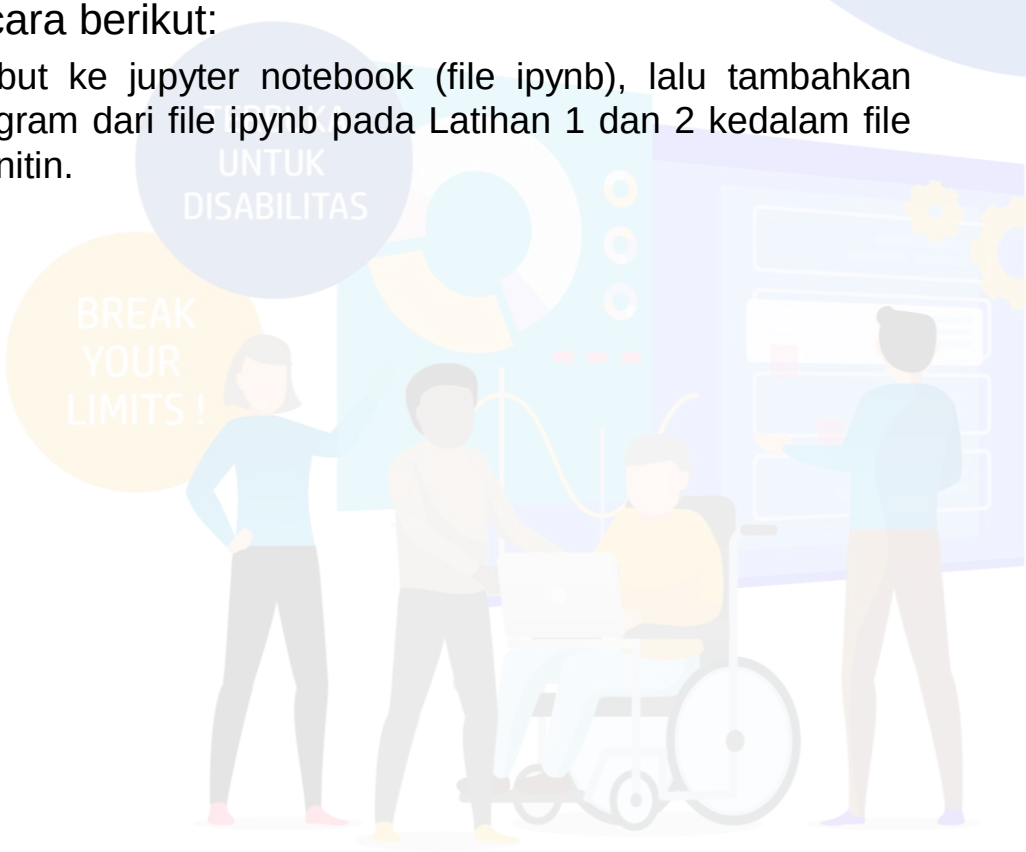


Latihan langsung di Kelas Ke-2 & Pembahasan

- Tidak ada

Tugas Individu

1. Buatlah rangkuman materi dengan cara berikut:
   - Pindahkan koding dari web tersebut ke jupyter notebook (file ipynb), lalu tambahkan penjelasan/comment pada tiap program dari file ipynb pada Latihan 1 dan 2 kedalam file word *.doc/*.docx untuk dicek di turnitin.



filkom.ub.id

DIGITAL TALENT SCHOLARSHIP 2019

Big Data Analytics

Terimakasih

Oleh: Imam Cholissodin | imamcs@ub.ac.id, Putra Pandu Adikara, Sufia Adha Putri
Asisten: Guedho, Sukma, Anshori, Aang dan Gusti
Fakultas Ilmu Komputer (Filkom) Universitas Brawijaya (UB)