Natural Language Processing
Text Scraping dan n-Gram LM (Praktikum)

Ahmad Rio Adriansyah, M.Si.

STT Terpadu Nurul Fikri

Script Python untuk Scraping

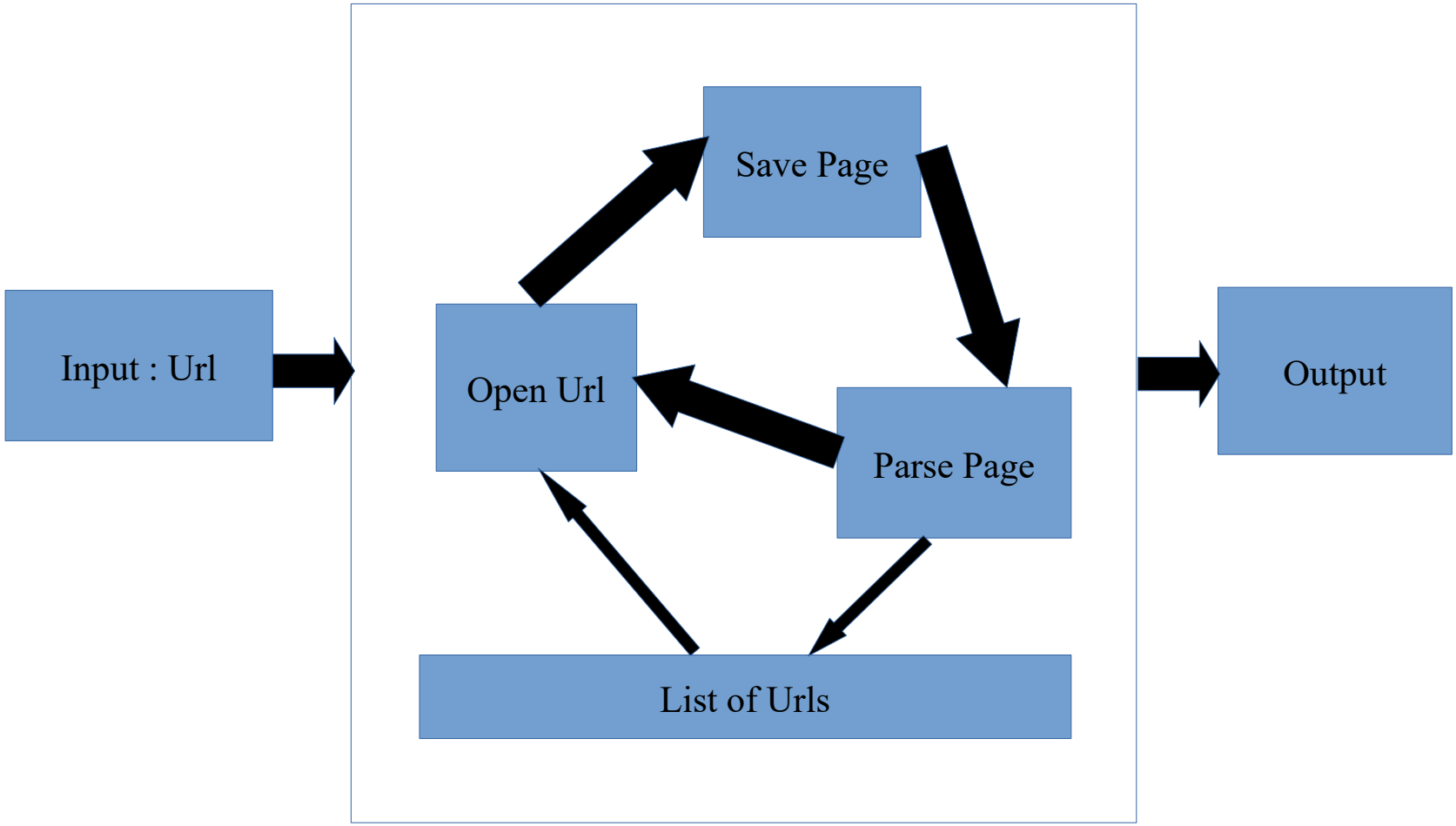
- Teknik yang digunakan untuk mengambil data (teks, gambar, video, atau file) dari website secara otomatis
- Bisa menggunakan :
  - Add-on pada Browser (Web Scraping, Scraper, Data Scraping, Scrapbook, dll)
  - Script sederhana
  - Framework/Software untuk scraping (Scrapy, Grab, Yakuza, FMiner)

Script Python untuk Scraping

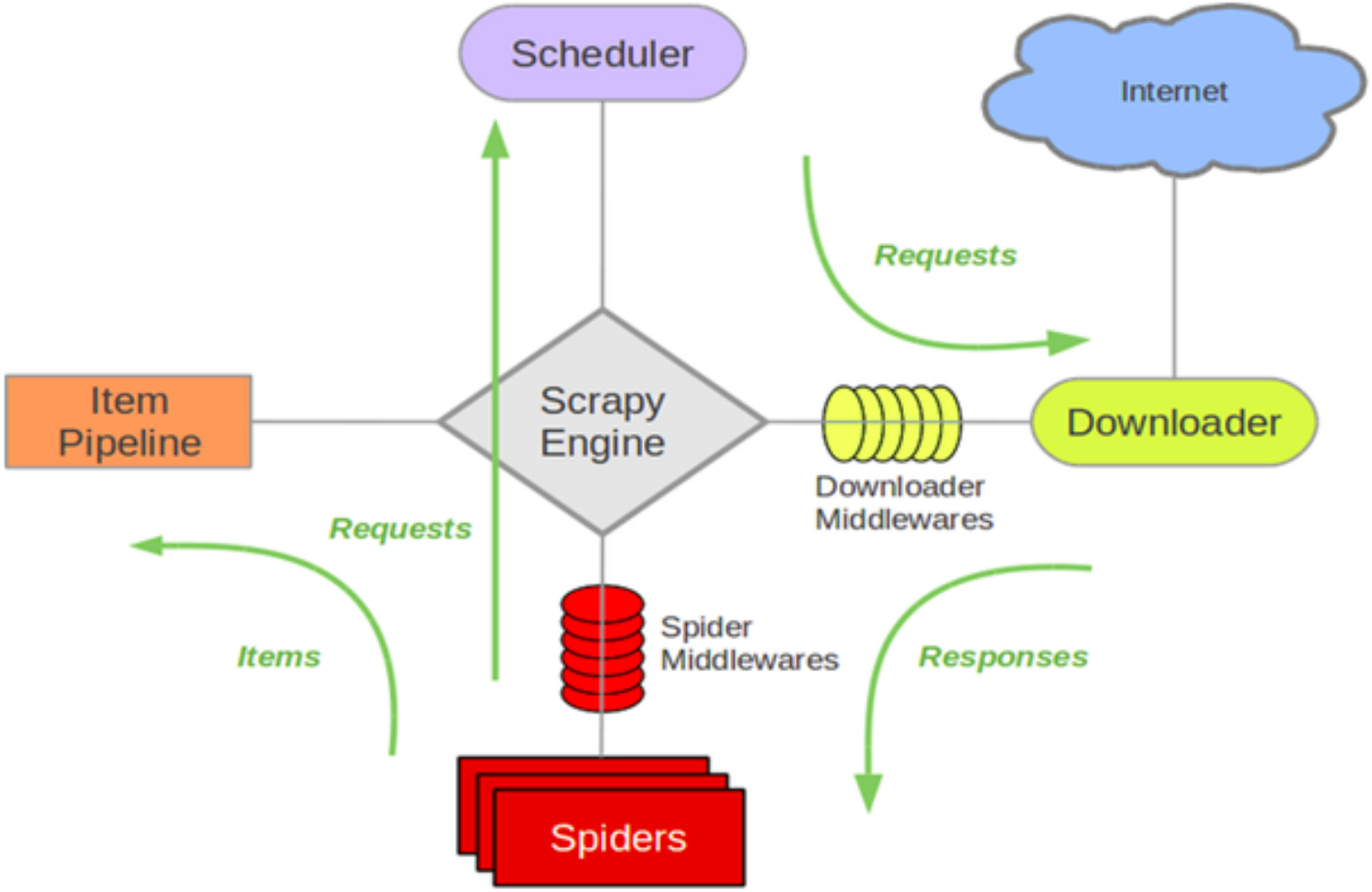
- Library yang dibutuhkan :
  - urllib / urllib2
  - Beautifulsoup

(https://www.crummy.com/software/BeautifulSoup/)

Alur



Alur (Scrapy)


Scheduler
Internet
Requests
Item
Pipeline
Scrapy
Engine
Downloader
Downloader
Middlewares
Requests
Items
Spider
Middlewares
Responses
Spiders

Praktikum Crawling (1)

- Buka halaman web melalui python

```
>>> import urllib2
>>> url = “http://www.detik.com/”
>>> page = urllib2.urlopen(url).read()
```

Praktikum Crawling (2)

- Parsing halaman web tersebut agar mudah dibaca dan diidentifikasi

```
>>> from bs4 import BeautifulSoup as BS
>>> soup = BS(page)
>>> print(soup.prettify())
```

Praktikum Crawling (3)

- Identifikasi posisi artikel/informasi yang mau diambil
- Bisa memanfaatkan firebug atau Inspect Element pada browser (Ctlr+Shift+I pada Chrome atau Firefox)
- Misal, artikel pada halaman tersebut tersimpan di dalam tag (<article></article>)

Praktikum Crawling (4)

- Misal, artikel pada halaman tersebut tersimpan di dalam tag (<article></article>)

  >>> artikel = soup.find(“article”)

  >>> print(artikel)

- (saat ini kondisinya artikel sudah didapatkan, tapi masih bersama tag htmlnya)

Praktikum Crawling (5)

- Bersihkan artikel yang sudah didapat

  >>> artikel = artikel.text

  >>> print(artikel)
- (saat ini kondisinya artikel sudah bersih dari tag html)
- Bisa dibersihkan lebih lanjut (perapihan antar paragraf, antar kalimat, penghapusan script, dll)

Praktikum Crawling (6)

- Buat list (array pada python) untuk menyimpan artikel, url target, dan url yang sudah diambil

```
>>> data = []
>>> target = []
>>> crawled = []
```

Praktikum Crawling (7)

- Selama target masih ada isinya, ambil satu url, buka, parsing, simpan

```
>>> while (target) :
...         url = target.pop(0)
...         if (url) in crawled :
...             # buka, parsing
...             ... dst ...
...         data.append(artikel)
...         crawled.append(url)
```

Praktikum Crawling (8)

- Parsing halaman yang dibuka untuk dicari link baru (update daftar link target)

```
…        # parsing
…        links = soup.find_all('a')
…        for link in links :
…            if link not in crawled :
…                target.append(link)
```

Alternatif Teks

Di elen disediakan teks dari situs berita detik.com dengan nama file "**detik.txt**", "**artikel10k.txt**", dan "**artikel1k.txt**".

Download filenya dengan cara klik kanan lalu pilih save link as (jika dibuka lewat browser terlalu besar)

Filenya masih mengandung informasi url, tag html, dll

Install NLTK dan Tokenizer

- Install NLTK
  - http://www.nltk.org/install.html
  - $ sudo pip install nltk
- Download tokenizer

```
>>> import nltk
>>> nltk.download()
Downloader> d
Identifier> punkt
```

Praktikum n-gram (1)

Persiapan artikel

```
>>> from bs4 import BeautifulSoup as BS
>>> berkas = open("artikel1k.txt","r")
>>> artikel = ""
>>> for teks in iter( lambda: berkas.readline(), ""):
...     artikel = artikel + " " + (teks[teks.find("<html>"):])
>>> artikel = BS(artikel).text
>>> artikel = artikel.replace("\n", " ")
>>> artikel = artikel.replace("\t", " ")
>>> artikel = artikel.replace(".", ". ") #kasih spasi setelah titik
```

Praktikum n-gram (2)

```
>>> import nltk
>>> from nltk.util import ngrams
>>> token = nltk.word_tokenize(artikel)
>>> bigram = ngrams(token,2)
>>> bigram.send(None)
```

Note : Jika dilakukan seperti di atas, maka seluruh artikel akan dianggap 1 kalimat.

Praktikum n-gram (3)

```
>>> from nltk import *
>>> from nltk.util import ngrams
>>> kalimat = sent_tokenize(artikel)
>>> token = []
>>> for klm in kalimat :
...         token.append(word_tokenize(klm))
```

Praktikum n-gram (3)

```
>>> listbigram = []
>>> for tk in token :
...         bigram = ngrams(tk,2, pad_right = True, pad_left = True, left_pad_symbol = '<s>', right_pad_symbol = '</s>')
...         for gram in bigram :
...             listbigram.append(gram)
```

Praktikum n-gram (3)

```
>>> bigram_fd = FreqDist(listbigram)
>>> for key,value in bigram_fd.items() :
...         print(key,value)
>>>
```

Note : Untuk menentukan token dapat juga menggunakan RegEx

Praktikum n-gram (4)

- Buat unigram dan trigramnya juga
- Simpan ketiganya menggunakan pickle

```
>>> import pickle
>>> pickle.dump(bigram_fd,
open('bigram.p', 'wb'))
```

Praktikum

- Setelah kita mendapatkan model bahasanya, kita dapat :
  - Membandingkan dua kalimat mana yang lebih tinggi kemungkinannya
  - Membuat kalimat yang mendekati bahasa tersebut

Peluang dari Kalimat

Praktikum P(w) (1)

- Sekarang kita punya distribusi frekuensi dari korpus artikel berupa unigram_fd, bigram_fd, dan trigram_fd
- Misalnya kita punya dua kalimat berikut :
  - “Saya minta copot itu untuk pidato budaya.”
  - “Saya minta itu dicopot untuk pidato budaya.”
- Kita bisa hitung mana yang paling mungkin benar menggunakan ngram

Praktikum P(w) (2)

>>> kal1 = “Saya minta copot itu untuk pidato budaya.”

>>> kal2 = “Saya minta itu dicopot untuk pidato budaya.”

>>> token = word_tokenize(kal1)

```
#peluang unigram
#P(w1w2...wn) = P(w1).P(w2). ... . P(wn)
p = 1
for kata in token:
    pkata = unigram_fd.freq((kata,))
    print("p('%s')=%f"%(kata, pkata))
    p=p*pkata
```

Praktikum P(w) (3)

- Dari hasil perhitungan, didapatkan peluang kedua kalimat tersebut menggunakan unigram adalah sbb :

No	Token	Peluang
1	Kal1	1.9318113058225432 e-26
2	Kal2	1.159086783493526 e-25

- Untuk bigram dan trigram, silahkan dicoba sendiri merujuk ke slide sebelumnya tentang teori ngram.

Generate Kalimat

Praktikum Generate Kalimat (1)

- Untuk menggenerate kalimat (secara acak) menggunakan model ngram, kita dapat menggunakan algoritma berikut :
  - Banyak **kata** dibatasi **n** buah
  - **kalimat** = “ ”
  - **kata** = **<s>**
  - Selama **n>0** dan **kata != </s>** lakukan
    - Ambil secara acak sebuah bigram yang berawal dari **kata**, misalnya (**kata**, **katabaru**)
    - Masukkan **kata** ke dalam **kalimat**
    - **kata** = **katabaru**
    - **n** = **n**-1

Praktikum Generate Kalimat (2)

```
1 import random
2 def generate(batas):
3    print("akan dibuat kalimat dengan maksimal %d kata menggunakan bigram"%batas)
4    kalimat = []
5    kata = "<s>"
6    selesai = False
7    while not selesai:
8       r = random.random()
9       count=0
10      for key,value in bigram_fd.items():
11         if(key[0]==kata):
12            count+=bigram_fd[key]
13      r = r*count
14      acc=.0
15      for key,value in bigram_fd.items():
16         if(key[0]==kata):
17            acc+=bigram_fd[key]
18         if(acc>=r):
19            kata = key[1]
20            kalimat.append(kata)
21            batas-=1
22            print(batas,kata,key)
23            break
24      if(kata=="</s>"):
25         kalimat=kalimat[:-1]
26         selesai = True
27      if(batas<=0):
28         selesai = True
29   return(' '.join(kalimat))
```

Beberapa Hasil Generating

'Total pemadaman maka tepungnya cukup banyak artikel menarik gerobak sampah'

'Pengadilan Negeri Meksiko Enrique Pena dan pergi sama sekali yang'

'Regulasi Pendidikan Provinsi Jatim belum bisa tanyakan ke Kairo'

'Namun siapa di sepanjang sejarah kurban yang dicapai penurunan'

'Salah satu dengan nama anak ke kalangan di twitter'

'Suradi mengatakan bahwa amfetamin yang menarik perhatian dari pihak'

'Kapolda juga .'

'`` Kecelakaan ini masih ditahan dasar Segara Anak pada Kementerian Luar Negeri Jakarta Selatan , Papua ?'

'Yang Saya ditanya tentang Penetapan tersangka) .'

'Kalau dimasukkan dalam pelarian , sangat berbahaya bagi pembangunan di bulan November , tidak memperlihatkan adanya kerusakan untuk riset dengan'

'Mereka menargetkan akhir perang di Kepulauan Riau mengklaim akan mendorong hubungan dekat perbatasan sedikit kemungkinan bisa ikut terbakar , serta'

'Pengibaran bendera merah menyala , stasiun Manggarai bersih kebutuhan pelaksanaan kebijakan lain yang menyabet gelar sebagai anggota dewan pengupahan ,'

Efisiensi Struktur

Menggunakan FreqDist pada bigram akan menghitung frekuensi tiap kemunculan bigramnya

[(a1,b1) : n11 , (a1,b2) : n12 , …]

Kekurangan :

- penyimpanan besar
- perlu menghitung kembali untuk frekuensi bersyaratnya

Efisiensi Struktur

Penyimpanannya diubah menjadi

[a1 : [b1 : n11 , b2 : n12 , …],
a2 : [b1 : n21 , b2 : n22 , …],
…]

Menggunakan ConditionalFreqDist

```
>>> from nltk.probability import ConditionalFreqDist
>>> bigram_cfd = ConditionalFreqDist(listbigram)
```

Generate Kalimat dengan CFD

```
 1 import random
 2 def generate(batas):
 3    print("akan dibuat kalimat dengan maksimal %d kata menggunakan bigram"%batas)
 4    kalimat = []
 5    kata = "<s>"
 6    selesai = False
 7    while not selesai:
 8       r = random.random()
 9       acc=.0
10       for key in bigram_cfd[kata]:
11          acc+=bigram_cfd[kata].freq(key)
12          if(acc>=r):
13             kata = key
14             kalimat.append(kata)
15             batas-=1
16             print(batas,kata,key)
17             break
18       if(kata=="</s>"):
19          kalimat=kalimat[:-1]
20          selesai = True
21       if(batas<=0):
22          selesai = True
23    return(' '.join(kalimat))
```