TEMU KEMBALI INFORMASI

JULIO ADISANTOSO
Departemen Ilmu Komputer IPB

Pertemuan 2

PEMROSESAN TEKS dan INVERTED INDEX

Pokok Bahasan Diskusi Tugas

1. Apa yang dimaksud dengan Analisis Teks secara Otomatis?
2. Apa ide dari Luhn, dan bagaimana hubungannya dengan Hukum Zipf?
3. Bagaimana menjelaskan makna dari Figure 2.1?
4. Apa saja prosedur utama pemrosesan teks?
5. Apa yang dimaksud dengan ”indexing” dan apa saja yang dilakukan?
6. Apa penjelasan terkait dengan konsep pembobotan?

Review: Statistik Teks

- Jumlah Kata: Seberapa besar korpus yang ada (N)
- Jenis kata:
  - Berapa jumlah kata yang unik?
  - Berapa besar perbendaharaan kata pada korpus?
- Token (dapat berupa kata, kalimat, paragraf, atau bagian teks lainnya)
  - Berapa jumlah token pada korpus?
  - Berapa frekuensi dari setiap jenis token?
  - Token apa yang paling sering muncul pada korpus?
  - Bagaimana hubungan antar token?
- Isu: Bagaimana melakukannya (Metode dan Algoritme, Program Komputer)?

Algoritme Frekuensi Kata

Input : Beberapa baris teks atau sebuah dokumen S
Output: Frekuensi setiap kata unik (array)

1: $words = split(delimiters, S)$;
2: **for each** t in words **do**
3: $lowerT = lower(t)$;
4: $res[lowerT]$++;
5: **end for**
6: **return** *res*

Isu selanjutnya:

- Bagaimana program komputernya?
- Menggunakan bahasa pemrograman apa?

Hello World!

C

```
#include <stdio.h>
int main() {
   printf("Hello World!\n");
   return 0;
}
```

Java

```
public class Hello {
   public static void main(String args[]) {
      System.out.println("Hello World!");
   }
}
```

Now in ... Python or PERL

```
print "Hello World!\n"
```

Why Python?

- Python is a scripting language (memiliki kekuatan di bidang pemrosesan teks, seperti halnya PERL) ... hanya membutuhkan interpreter, namun lebih ”elegan” dibanding PERL
- High level language sehingga mudah dipahami, seperti halnya C, Java, PERL, etc ... mirip pseudocode
- Prosedural (seperti C, Pascal, etc), tetapi juga OOP (seperti C++, Java), dan terkadang fungsional (seperti Scheme, Haskell)
- Tersedia banyak library pemrosesan teks ... program menjadi lebih singkat

Python is not just for kidding ...

upss ... scripting

BUT

for serious coding

Contoh Array

Membalik Array

```
def balik(a):
    if a==[]:
        return []
    else:
        return balik(a[1:])+[a[0]]
```

Menggunakan built-in list function

```
def balik(a):
    return a.reverse()
```

Python Memudahkan ... mirip Pseudocode

Quick Sort

```
def sort(a):
    if a==[]:
        return []
    else
        pivot= a[0]
        left = [x for x in a if x < pivot]
        right= [x for x in a[1:] if x >= pivot]
        return sort(left)+pivot+sort(right)
```

Numbers and Strings

- Tipe data: bilangan (int, float), bool, string, list.
- Tidak perlu deklarasi variabel !
- Operator aritmatika: + - * / ** %
- Baca Panduan Singkat Python!

Contoh

```
>>> a=5
>>> a+=4
9
>>> 5/2
2
>>> 5/2.
2.5
```

Contoh

```
>>> s="Hello"
>>> s+" World!"
'Hello World!'
>>> s[0]
H
>>> s[-1]
d
```

Assignments and Comparisons

Assignments

```
>>> a=b=0
>>> a
0
>>> b
0
>>> a, b=3, 5
>>> a+b
8
>>> (a,b)=(3,5)
>>> a+b
8
>>> a,b=b,a
(swap)
```

Comparisons

```
>>> a=b=0
>>> a==b
True
>>> type (3==5)
<class 'bool'>

>>> (1,2)==(1,2)
True
>>> (1,2)==(2,1)
False

>>> 1,2 == 1,2
(1, False, 2)
```

Conditional

Conditional IF

```
>>> x=9
>>> if x<10 and x>=0:
        print(x, "is digit")

9 is digit
>>> False and False or True
True
>>> not True
False
>>> if 4>5:
        print("foo")
    else:
        print("bar")

bar
>>> print("foo" if 4>5 else "bar")
bar
```

For and Range

`for` selalu menggunakan `list` atau sequences

for Loop

```
>>> for i in range(10):
        print(i, end=" ")

0 1 2 3 4 5 6 7 8 9
>>> sum=0
>>> for i in range(10):
        sum+=i

>>> print(sum)
45
>>> for word in ["welcome","to", "python"]:
        print(word, end=" ")

welcome to python
```

String

Mengolah String

```
>>> s="ini adalah program python. \n"
>>> kata=s.split()
>>> kata
['ini', 'adalah', 'program', 'python.']
>>> s.strip()
'ini adalah program python.'
>>> " ".join(kata)
'ini adalah program python.'
>>> s.find("adalah")
4
>>> s.find("program c")
-1
>>> s.replace("adalah", "merupakan")
'ini merupakan program python. \n'
>>> kata=s.replace(".","").split()
>>> kata
['ini', 'adalah', 'program', 'python']
>>>
```

String

Enumerate

```
>>> s="ini adalah program python. \n"
>>> i=0
>>> for kata in enumerate(s.replace(".","").split()):
        i+=1
        print(i, kata, end='\n')

1 (0, 'ini')
2 (1, 'adalah')
3 (2, 'program')
4 (3, 'python')
>>>
>>> for i,kata in enumerate(s.replace(".","").split()):
        print(i+1,kata,end='\n')

1 ini
2 adalah
3 program
4 python
```

Import and I/O

- Import mirip dengan Java, memasukkan package ke dalam program
- Mirip dengan #include pada program C
- File I/O lebih mudah dibanding Java

Contoh Membaca dari Standard Input

```
import sys
for line in sys.stdin:
    print line.lower().replace(".","").split()

#python hitung.py < text.txt
```

Dictionaries

- Dalam struktur data, dikenal dengan hash map
- Sangat efisien untuk mengelola teks
- Untuk membuat nilai default, digunakan `defaultdict`

Dictionaries : Hash Map

```
>>> d={'be':2, 'not':1, 'to':3}
>>> d
{'to': 3, 'be': 2, 'not': 1}
>>> d['to']
3
>>> d.keys()
dict_keys(['to', 'be', 'not'])
>>> d.values()
dict_values([3, 2, 1])
```

Menghitung Frekuensi Kata

1: $words = split(delimiters, S)$;
2: **for each** t in words **do**
3: $\quad lowerT = lower(t)$;
4: $\quad res[lowerT]$++;
5: **end for**
6: **return** *res*

Program : hitung2.py

```
import sys
from collections import defaultdict
daftar = defaultdict(int)
for i,line in enumerate(sys.stdin):
   for kata in line.lower().replace(".","").split():
      daftar[kata] += 1
for w in sorted(daftar, key=daftar.get, reverse=True):
   print w, daftar[w]
```

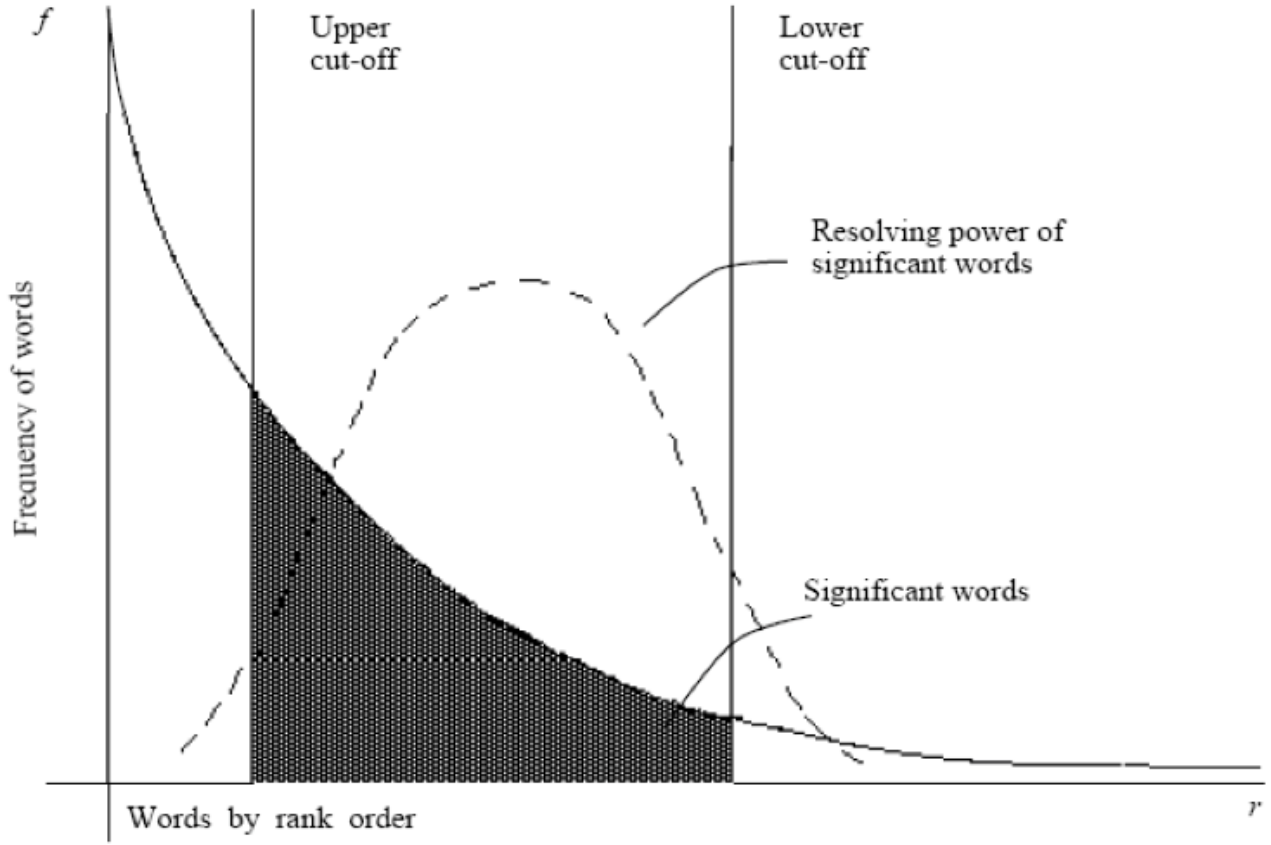
Fenomena Frekuensi Kata ... Luhn's idea

- Sejumlah kata merupakan kata yang sangat umum (frekuensi sangat besar), misalnya "the", "of", "yang", "untuk", dsb
- Kebanyakan kata sangat jarang muncul (frekuensi sangat kecil)
- Setengah dari kata-kata pada korpus hanya muncul sekali.
- Contoh:

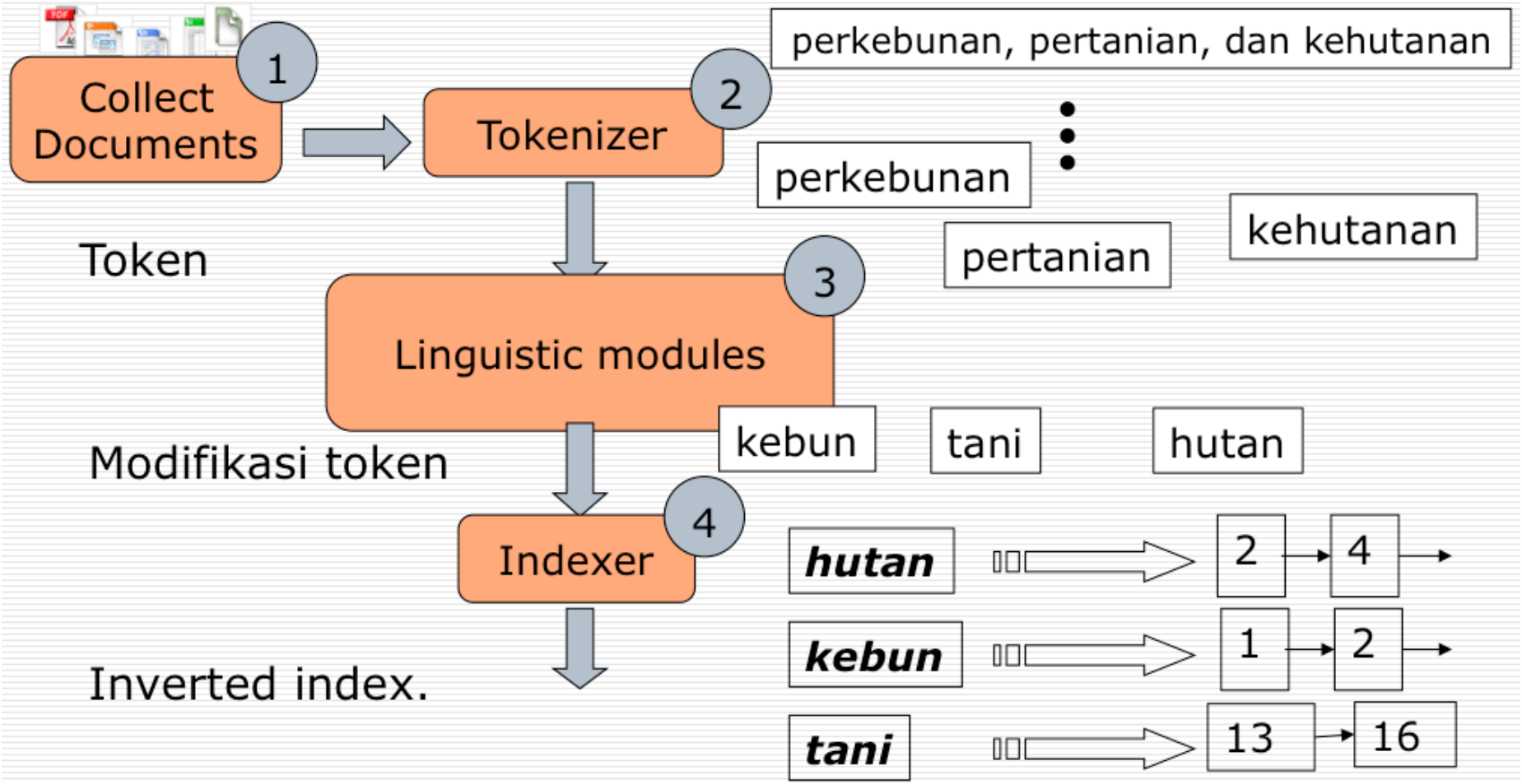
Kata	Frekuensi Kata (f)	Peringkat (r)	f * r
name	21	400	8400
comes	16	500	8000
group	13	600	7800
science	11	700	7700
family	10	800	8000
begin	9	900	8100
broke	4	2000	8000
seems	2	3000	6000
could	2	4000	8000

Hukum Zipf (George Kingsley Zipf)

- Menjelaskan adanya hubungan antara frekuensi (f) dan urutan/rank (r), yaitu f*r=constant

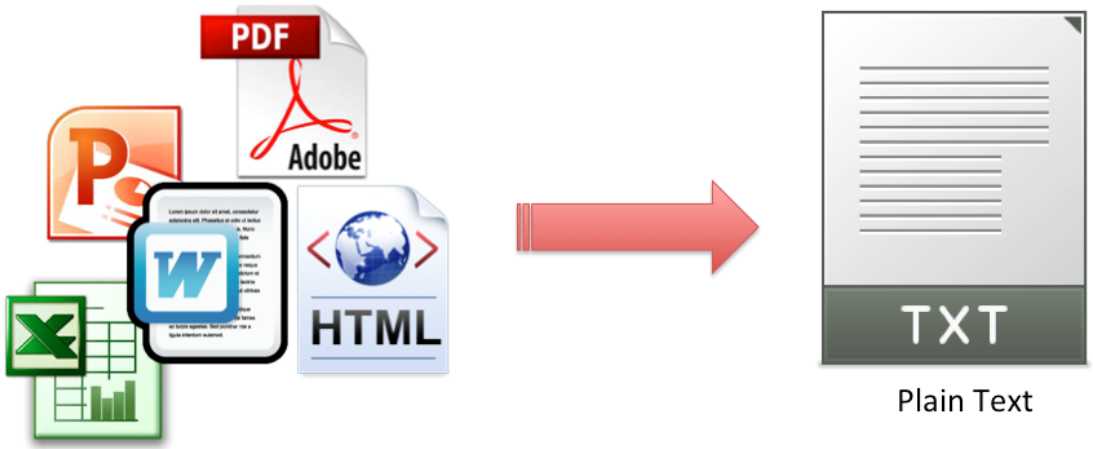


Inverted index construction (Manning et al, 2008)



[1] Collect the documents

- Mendapatkan character sequences dalam dokumen. Konversi dari file atau halaman web menjadi deretan karakter membentuk kata/kalimat. Format dokumen hasil software umumnya memiliki banyak header dan format yang khusus.
- Menentukan unit dokumen yang akan di-index



[1] Collect the documents

Banyak tersedia library untuk membaca berbagai format dokumen

Contoh Program Membaca Dokumen Web Online

```
import os
import os.path
import requests
import re
from collections import defaultdict

daftar = defaultdict(int)
rawhtml=requests.get("http://www.ipb.ac.id")
for i,line in enumerate(rawhtml):
   for kata in line.lower().replace(".","").split():
      kata=re.sub('<[A-Za-z\/][^>]*>', '', kata)
      daftar[kata] += 1
for w in sorted(daftar, key=daftar.get, reverse=True):
   print w, daftar[w]
```

[2] Tokenization ... Tokenisasi

- Proses memotong dokumen menjadi bagian-bagian yang lebih kecil, disebut *token*.
- Token dapat berupa: (1) paragraf, (2) kalimat, (3) frasa, (4) kata tunggal (sederhana), (5) konsep
- Teknik yang digunakan: segmentasi, memilah

[3] Linguistic modules

- Melakukan proses normalisasi terhadap token:
  - membuang tanda baca, membuang imbuhan (stemming), dsb
  - case-folding
  - membuang stopwords (daftar kata yang tidak perlu, misalnya dan, kepada, dari, dsb)
  - menambahkan sinonim, antonim, dsb
- Istilah *term* sering digunakan bagi token kata/frasa yang sudah mengalami proses normalisasi
- Daftar dari *term* biasa disebut sebagai *dictionary*

[4] Membentuk Inverted Index

Membuat daftar posting (*posting list*) dari setiap term, yaitu kemunculan suatu term ada di unit dokumen mana saja. Contoh:

```
pertanian, 993427:
(1, 6: (7, 18, 33, 72, 86, 231);
 4, 5: (8, 16, 190, 429, 433);
 7, 3: (13, 23, 191); ...)
```

Artinya, term ”pertanian” ada sebanyak 993427 di seluruh dokumen dalam korpus, muncul 6 kali di dokumen 1 pada posisi ke-7, 18, 33, 72, 86, dan 231. Dan seterusnya.

Bagaimana struktur datanya?

Pelajari contoh program invertedindex.py !

Term Weighting

- Perlunya suatu term diberi bobot:
  Makin sering suatu term muncul pada suatu dokumen, maka diduga semakin penting term itu untuk dokumen tersebut.
- Menggunakan pendekatan statistik
- Beberapa ukuran bobot:
  - Boolean
  - TF (Term Frequency)
  - TF.IDF (TF-Inverse Document Frequency)
  - BM25 (dibahas pada pertemuan lainnya)

Term Frequency

- Frekuensi kemunculan suatu term t pada dokumen $d \rightarrow tf_{td}$
- Paling sederhana
- Contoh (perhatikan term `dari`, `dengan`, `yang`):

term	d1	d2	d3	d4	d5
dari	20	100	10	22	10
database	15	0	0	0	12
dengan	12	40	12	14	24
informasi	8	30	0	0	18
komputer	10	35	0	0	0
struktur	0	24	6	10	10
yang	35	120	15	32	20

IDF

- Mana yang lebih informatif bagi suatu term untuk mencirikan dokumen?
  - Seberapa jarang suatu term muncul di seluruh dokumen?
  - Seberapa sering suatu term muncul di seluruh dokumen?
- Banyaknya dokumen dimana suatu term t muncul (disebut Document Frequency) adalah DF_t, menunjukkan ukuran ketidakpentingan suatu term dalam korpus dokumen.
- Ukuran kepentingan suatu term t dalam korpus dokumen adalah: $IDF_t = \log(\frac{N}{DF_t})$. Berarti $IDF_i < IDF_j$ menunjukkan bahwa term j lebih penting dibanding term i.
- Akan *misleading* kalau ada term yang hanya muncul di sangat sedikit dokumen akibat salah ketik, istilah khusus, dsb

TF.IDF

- Bagaimanapun juga, peran TF dalam bobot suatu term tetap penting
- *IDF* akan menimbulkan *misleading* kalau ada term yang hanya muncul di sangat sedikit dokumen akibat salah ketik, istilah khusus, dsb
- Maka pembobotan TF.IDF term t pada dokumen d adalah:

$$TF.IDF_{t,d} = TF_{t,d} \times \log(\frac{N}{DF_t})$$

Variasi IDF

- Sub-linear Scaling:

$$IDF_{t,d} = \begin{cases} 1 + \log(TF_{t,d}) & \texttt{untuk } TF_{t,d} > 0 \\ 0 & \texttt{selainnya} \end{cases}$$

- Okapi Weighting:

$$IDF_{t,d} = \max\left\{0, \log\left(\frac{N - DF_{t,d} + 1/2}{DF_{t,d} + 1/2}\right)\right\}$$

SMART Notation for Term Weighting

Term frequency		Document frequency		Normalization	
n (natural)	$\mathrm{tf}_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1+\log(\mathrm{tf}_{t,d})$	t (idf)	$\log \frac{N}{\mathrm{df}_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2+w_2^2+\ldots+w_M^2}}$
a (augmented)	$0.5+\frac{0.5\times\mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$	p (prob idf)	$\max\{0,\log\frac{N-\mathrm{df}_t}{\mathrm{df}_t}\}$	u (pivoted unique)	$1/u$ (Section 6.4.4)
b (boolean)	$\begin{cases}1 & \text{if } \mathrm{tf}_{t,d}>0\\ 0 & \text{otherwise}\end{cases}$			b (byte size)	$1/CharLength^{\alpha}$, $\alpha<1$
L (log ave)	$\frac{1+\log(\mathrm{tf}_{t,d})}{1+\log(\mathrm{ave}_{t\in d}(\mathrm{tf}_{t,d}))}$				

- Sebagai contoh, pembobotan *TF.IDF* dinotasikan sebagai pembobotan *ntn*
- *CharLength* adalah banyaknya karakter dalam dokumen