

Tugas Mata Kuliah Teks dan Web Mining

Tugas 3

disusun untuk memenuhi tugas mata kuliah
Teks dan Web Mining

Oleh :

Mr. HADAFEE MUDO : 2008107010101



**JURUSAN INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS SYIAH KUALA
2023**

Pada Tugas 3 Text dan Web Mining meminta untuk membangun fitur dengan menggunakan halaman web dari dua kategori. Disini, kategori yang digunakan adalah kategori food dan tren dari portal berita detik. Terdapat 12000 berita yang digunakan untuk membangun fitur, dimana masing-masing kategori memiliki 6000 halaman web. Fitur tersebut dibangun dari 6 bagian halaman web, yaitu bagian judul (title) dan isi (content) dari masing-masing halaman web.

Langkah pertama yang dilakukan dalam membangun fitur adalah membaca dan membersihkan 12000 halaman web. Pada proses pembersihan, content dari masing-masing halaman web dibagi menjadi 6 bagian. Setelah semua halaman web dibersihkan dan dibagi menjadi 6 bagian, langkah selanjutnya adalah membaca folder Kamus yang berisi file .txt yang berisi hasil thesaurus dengan nilai 0.5 dan 0.45.

Setelah itu, hasil fitur yang telah dibangun akan ditampilkan dalam format ARFF.

Membagi berita menjadi 6 bagian

Proses ini dilakukan secara bersamaan dengan proses pembersihan halaman web dari tag atau gambar yang tidak diperlukan, sehingga hanya memuat berita inti saja. Pembagian dilakukan pada bagian title dan content. Bagian title dibagi menjadi 1 bagian, sedangkan bagian content dibagi menjadi 5 bagian. Proses ini bertujuan untuk memudahkan dalam membangun fitur dari masing-masing bagian halaman web yang terpisah.

```
1 from typing import Counter
2 from bs4 import BeautifulSoup as bs
3 import os
4 import sys
5 import time
6
7 start = time.time()
8 total_documents = 0
9
10
11
12 total_documents = len(os.listdir('crawl'))
13
14 for file in os.listdir('crawl'):
15     with open('crawl/' + file, "r", encoding="utf8") as data:
16
17         soup = bs(data, "html.parser")
18         data.close()
19
20         # ambil judul
21         title = soup.title
22
23         # ambil url
24         url = soup.find("link", rel="canonical").get("href")
25
26         # ambil artikel -> wabah
27         article = soup.body.find(
28             "div", class_="read_content").find_all("p")
29         article_list = []
```

```

30
31 junk_words = ["KOMPAS.com", "-", "Baca:", "Baca juga:", "Baca Juga:", "({thci.bunn)",
32 "Baca Juga:", "Baca Juga:\xa0", "Baca juga:\xa0"]
33 # hapus tag <a> dan tag <strong> yang isinya tidak diperlukan
34 for p in article:
35     strong = p.find("strong")
36     if(strong):
37         for x in junk_words:
38             if(x in strong.contents):
39                 strong.decompose()
40         article_list.append(p.text)
41
42 article_string = "".join(article_list)
43 article_string = article_string.replace("KOMPAS.com ", "")
44 article_string = article_string.replace("KOMPAS.com- ", "")
45 article_string = article_string.replace("JAKARTA,", "")
46 article_string = article_string.replace("Baca\xa0juga:", "")
47 article_string = article_string.replace("Baca" "" "juga:\xa0", "")
48 article_string = article_string.replace("Baca" "" "juga:\xa0", "")
49 article_string = article_string.replace("Baca:\xa0 ", "")
50 article_string = article_string.replace("DEPOK, ", "")
51
52 # bagi artikel
53 bagian1 = []
54 bagian2 = []
55 bagian3 = []
56 bagian4 = []
57 bagian5 = []
58 length = len(article_string)
59
60 # presentase jumlah karakter per bagian (top, mid, bottom)
61 part1 = round(20/100*length)
62 #part2 = round(33/100*length)
63
64 # cari titik
65 extra = 0
66
67 #bagian 1
68 try:
69     for x in range(part1):
70         bagian1.append(article_string[x])
71         while(bagian1 and x >= part1-1):
72             bagian1.append(article_string[x+1])
73             x += 1
74             extra += 1
75             if(bagian1[-1] == "." and not article_string[x+1].isdigit()):
76                 break
77 except:
78     print("Something is wrong")
79
80 extra2 = 0
81 #bagian 2
82 try:
83     for x in range((part1 + extra), (part1 * 2)):
84         bagian2.append(article_string[x])
85         while(bagian2 and x >= ((part1 * 2) - 1)):
86             bagian2.append(article_string[x+1])
87             x += 1
88             extra2 += 1
89             if(bagian2[-1] == "." and not article_string[x+1].isdigit()):
90                 break
91 except:
92     print("Something is wrong")
93
94 extra3 = 0
95 #bagian 3
96 try:
97     for x in range(((part1 * 2) + extra2), (part1 * 3)):
98         bagian3.append(article_string[x])
99         while(bagian3 and x >= ((part1 * 3) - 1)):
100             bagian3.append(article_string[x+1])
101             x += 1

```

```

102         extra3 += 1
103         if(bagian3[-1] == "." and not article_string[x+1].isdigit()):
104             break
105     except:
106         print("Something is wrong")
107
108     extra4 = 0
109     #bagian 4
110     try:
111         for x in range(((part1 * 3) + extra3), (part1 * 4)):
112             bagian4.append(article_string[x])
113             while(bagian4 and x >= ((part1 * 4) - 1)):
114                 bagian4.append(article_string[x+1])
115                 x += 1
116                 extra4 += 1
117                 if(bagian4[-1] == "." and not article_string[x+1].isdigit()):
118                     break
119     except:
120         print("Something is wrong")
121
122     #bagian 5
123     try:
124         for x in range(((part1 * 4) + extra4), length):
125             bagian5.append(article_string[x])
126             while(bagian5 and x >= (length-1) and bagian5[-1] != '.'):
127                 bagian5.pop(-1)
128
129     except:
130         print("Something is wrong")
131
132     with open("clean/" + file[:-5] + ".clean.dat", 'w', encoding="utf8") as scrapped_html:
133
134         scrapped_html.write("<title>" + title.get_text() + "</title>\n\n")
135
136         scrapped_html.write("<bagian1>")
137         for text in bagian1:
138             scrapped_html.write(text)
139         scrapped_html.write("</bagian1>\n\n")
140
141         scrapped_html.write("<bagian2>")
142         for text in bagian2:
143             scrapped_html.write(text)
144         scrapped_html.write("</bagian2>\n\n")
145
146         scrapped_html.write("<bagian3>")
147         for text in bagian3:
148             scrapped_html.write(text)
149         scrapped_html.write("</bagian3>\n\n")
150
151         scrapped_html.write("<bagian4>")
152         for text in bagian4:
153             scrapped_html.write(text)
154         scrapped_html.write("</bagian4>\n\n")
155
156         scrapped_html.write("<bagian5>")
157         for text in bagian5:
158             scrapped_html.write(text)
159         scrapped_html.write("</bagian5>")
160
161     scrapped_html.close()
162
163     os.system('cls')
164
165
166     print("Complete.")

```

Membangun Fitur

pertama-tama dibuat array untuk masing-masing kategori, yaitu food dan tren . Setelah itu, dilakukan muat ulang file .txt dari folder Kamus dan dimasukkan ke dalam hashfood untuk kategori food dan hashtren untuk kategori tren.

Selanjutnya, dilakukan penghitungan frekuensi kemunculan kata-kata pada setiap bagian halaman web yang telah dibagi sebelumnya. Frekuensi ini dihitung dengan menggunakan hashfood dan hashtren yang telah dimuat sebelumnya.

Fitur kemudian dibangun dengan menggunakan metode bag-of-words. Metode ini merupakan teknik dasar dalam pemrosesan bahasa alami yang menggabungkan frekuensi kemunculan kata-kata pada setiap bagian halaman web menjadi satu vektor yang merepresentasikan keseluruhan halaman web.

Fitur yang telah dibangun kemudian ditampilkan dalam format ARFF, yaitu format file yang digunakan untuk menyimpan data yang akan digunakan sebagai input pada pemodelan machine learning. Dalam format ARFF, setiap fitur direpresentasikan sebagai atribut dengan nilai numerik yang merepresentasikan frekuensi kemunculan kata pada halaman web. Atribut ini kemudian digunakan sebagai input pada model machine learning yang akan dibangun.

```

1 use strict;
2 use warnings;
3 use lib 'Lingua-EV-Ngram-0.03/lib/';
4 use Lingua::EN::Ngram;
5 use POSIX qw(cuil);
6
7 my $ngrams = Lingua::EN::Ngram->new;
8 my $gram_store;
9 my $PATII = "./clean";
10 my $dictfile = "./titur";
11 if(! $dictfile){
12     print "Cara jalankan : $0 <directory file>\n";
13 }
14
15 my @dictionary = ("Kamus+ood-0.5.txt", "Kamus/tren-0.5.txt");
16
17 print "load & Hash Dictionary\n";
18 my (%hash+ood,%hashtren);
19
20 foreach my $dictfile (@dictionary) {
21     open my $fh, "<", $dictfile or die "Cannot open $dictfile: $!";
22     while (my $line = <$fh>) {
23         chomp($line);
24         next if ($line =~ /^#/);
25         my ($word, $score) = split /\:/, $line;
26         if ($dictfile =~ /food/ && !exists(%hash+ood{$word})) {
27             $hash+ood{$word} = 1;
28         } elsif ($dictfile =~ /tren/ && !exists(%hashtren{$word})) {
29             $hashtren{$word} = 1;
30         }
31     }
32     close $fh;
33 }

```

```

34
35 my $process=0;
36 foreach my $dir(($$PATII/food", "$PATII/tren")){
37     my $srcfile = split /\//,$dir;
38     open OUT,"> $dirfile/feature_0.5_$srcfile[2].dat" or die "Can't open file...";
39     open ARFF,"> $dirfile/feature_0.5_$srcfile[2].arff" or die "Can't open file...";
40     print ARFF "\@relation feature_0.5_$srcfile[2]\n\n";
41     foreach my $iterate (1 .. 48){
42         print ARFF "\@attribute feature$iterate numeric\n";
43     }
44     print ARFF "\@attribute class {food,tren}\n\n@DATA\n";
45
46     my @files = glob("$dir/*.clean.dat");
47
48     print $dir;
49     my $number = 1;
50
51     foreach my $file(@files){
52         $process++;
53
54         open F, $file or die "Can't open input: $!\n";
55         my $text = do { local $/; <F> };
56         close F;
57         $text =~ s/\n+//gs;
58
59         if($file =~ /food/){
60             print OUT "food ";
61         }else{
62             print OUT "tren ";
63         }
64
65         my @sections = (
66             get_text($text,"title"),
67             get_text($text,"bagian1"),
68             get_text($text,"bagian2"),
69             get_text($text,"bagian3"),
70             get_text($text,"bagian4"),
71             get_text($text,"bagian5"),
72         );
73         my @weight = (1, 0.2, 0.2, 0.2, 0.2, 0.2);
74         for (my $sec = 0 ; $sec < @sections; $sec++){
75             for (my $dict = 0 ; $dict < @dictionary ; $dict++){
76                 foreach my $n (1 .. 3){
77                     $gram_score=0;
78                     if($dict == 0){
79                         $gram_score = count_section($sections[$sec], \%hashfood, $n) * $weight[$sec];
80                     }elseif($dict == 1){
81                         $gram_score = count_section($sections[$sec], \%hashtren, $n)* $weight[$sec];
82                     }
83
84                     print ARFF sprintf("%.4f",$gram_score).",";
85                     print OUT "$number:".sprintf("%.4f",$gram_score). " ";
86                     $number++;
87                 }
88             }
89         }
90         if($file =~ /food/){
91             print ARFF "food\n";
92         }else{
93             print ARFF "tren\n";
94         }
95         print OUT "\n";
96         $number = 1;
97         if($process % 1000 == 0){
98             print "\nDone : $process\n"
99         }

```

```

100     }
101     close OUT;
102     close ARFF;
103 }
104
105 sub get_text{
106     my ($text,$regex) = @_;
107
108     if($text =~ /<$regex>(.*?)<\/$regex>/{
109         return $1;
110     }
111 }
112
113 sub count_section {
114     my($section, $hash, $n) = @_;
115     my $count=0;
116     my $sect = clean_string($section);
117
118     $ngrams->text($sect."");
119     my $lxl = $ngrams->lexl;
120     my $grams = $ngrams->ngram($n);
121     my $gramlen = keys %$grams;
122
123     foreach my $gram ( sort { $$grams{ $b } <=> $$grams{ $a } } keys %$grams ) {
124         $gramlen++;
125         chomp($gram);
126
127         next if($gram =~ /^#/ || $gram =~ /^$/);
128
129         if (defined $$hash{$gram}) {
130             $count++;
131         } else {
132             $count = 1;
133         }
134     }
135
136     if ($count == 0) {
137         return 0;
138     } else {
139         return ($count/$gramlen);
140     }
141 }
142
143 sub clean_string {
144     my $file = shift;
145     $file =~ s/<.*?>//g;
146     $file =~ s/\\n\\n-.*?>/ /g;
147     $file =~ s/>//g;
148     $file =~ s/&.*?//g;
149     $file =~ s/[\\:\/]|\[\\?\\!\\@\\#\\$\\%\\&\\'\\/\\(\\)\\;|'//g;
150     $file =~ s/-/ /g;
151     $file =~ s/\\s+/ /g;
152     $file = lc($file);
153     return $file;
154 }

```


Hasil Fitur Yang Dibangun

- Kategori Food - Threshold 0.45

- Kategori Food - Threshold 0.45

```

47 # feature.045.food.tren
48 @attribute feature46 numeric
49 @attribute feature47 numeric
50 @attribute feature48 numeric
51 @attribute class {food,tren}
52
53 @DATA
54 0.0385,0.0385,0.0385,0.0385,0.0385,0.0385,0.0012,0.0008,0.0008,0.0012,0.0008,0.0008,0.0010,0.0008,0.0008,0.0010,0.0008,0.0008,0.0
55 0.0455,0.0417,0.0417,0.0455,0.0417,0.0417,0.0024,0.0020,0.0020,0.0019,0.0024,0.0020,0.0019,0.0027,0.0021,0.0019,0.0027,0.0021,0.0019,0.0
56 0.0455,0.0455,0.0455,0.0455,0.0455,0.0455,0.0045,0.0032,0.0031,0.0045,0.0032,0.0031,0.0027,0.0024,0.0024,0.0027,0.0024,0.0024,0.0
57 0.0500,0.0500,0.0500,0.0500,0.0500,0.0500,0.0019,0.0016,0.0016,0.0019,0.0016,0.0016,0.0023,0.0020,0.0019,0.0023,0.0020,0.0019,0.0
58 0.0455,0.0455,0.0455,0.0455,0.0455,0.0455,0.0028,0.0023,0.0023,0.0028,0.0023,0.0023,0.0028,0.0024,0.0023,0.0028,0.0024,0.0023,0.0
59 0.0500,0.0500,0.0500,0.0500,0.0500,0.0500,0.0020,0.0017,0.0016,0.0020,0.0017,0.0016,0.0020,0.0016,0.0016,0.0020,0.0016,0.0016,0.0
60 0.0385,0.0385,0.0385,0.0385,0.0385,0.0385,0.0019,0.0015,0.0014,0.0019,0.0015,0.0014,0.0019,0.0014,0.0014,0.0019,0.0014,0.0014,0.0
61 0.0385,0.0385,0.0385,0.0385,0.0385,0.0385,0.0015,0.0011,0.0010,0.0015,0.0011,0.0010,0.0022,0.0016,0.0016,0.0022,0.0016,0.0016,0.0
62 0.0385,0.0385,0.0385,0.0385,0.0385,0.0385,0.0024,0.0019,0.0018,0.0024,0.0019,0.0018,0.0026,0.0020,0.0019,0.0026,0.0020,0.0019,0.0
63 0.0417,0.0417,0.0417,0.0417,0.0417,0.0417,0.0013,0.0010,0.0009,0.0013,0.0010,0.0009,0.0024,0.0018,0.0018,0.0024,0.0018,0.0018,0.0
64 0.0500,0.0455,0.0455,0.0500,0.0455,0.0455,0.0026,0.0023,0.0022,0.0026,0.0023,0.0022,0.0028,0.0025,0.0025,0.0028,0.0025,0.0025,0.0
65 0.0455,0.0455,0.0455,0.0455,0.0455,0.0455,0.0018,0.0012,0.0011,0.0018,0.0012,0.0011,0.0019,0.0013,0.0011,0.0019,0.0013,0.0011,0.0
66 0.0417,0.0385,0.0385,0.0417,0.0385,0.0385,0.0018,0.0014,0.0014,0.0018,0.0014,0.0014,0.0045,0.0037,0.0034,0.0045,0.0037,0.0034,0.0
67 0.0455,0.0455,0.0455,0.0455,0.0455,0.0455,0.0028,0.0022,0.0020,0.0028,0.0022,0.0020,0.0033,0.0029,0.0029,0.0033,0.0029,0.0029,0.0
68 0.0455,0.0455,0.0455,0.0455,0.0455,0.0455,0.0024,0.0017,0.0016,0.0024,0.0017,0.0016,0.0023,0.0017,0.0016,0.0023,0.0017,0.0016,0.0
69 0.0500,0.0500,0.0500,0.0500,0.0500,0.0500,0.0026,0.0020,0.0018,0.0026,0.0020,0.0018,0.0027,0.0021,0.0021,0.0027,0.0021,0.0021,0.0
70 0.0455,0.0455,0.0455,0.0455,0.0455,0.0455,0.0017,0.0014,0.0013,0.0017,0.0014,0.0013,0.0019,0.0017,0.0017,0.0019,0.0017,0.0017,0.0
71 0.0556,0.0556,0.0556,0.0556,0.0556,0.0556,0.0043,0.0034,0.0034,0.0043,0.0034,0.0034,0.0045,0.0037,0.0036,0.0045,0.0037,0.0036,0.0
72 0.0385,0.0385,0.0385,0.0385,0.0385,0.0385,0.0019,0.0015,0.0014,0.0019,0.0015,0.0014,0.0042,0.0040,0.0040,0.0042,0.0040,0.0040,0.0
73 0.0625,0.0625,0.0625,0.0625,0.0625,0.0625,0.0045,0.0038,0.0037,0.0045,0.0038,0.0037,0.0043,0.0037,0.0036,0.0043,0.0037,0.0036,0.0
74 0.0417,0.0417,0.0417,0.0417,0.0417,0.0417,0.0029,0.0022,0.0020,0.0029,0.0022,0.0020,0.0042,0.0030,0.0030,0.0042,0.0030,0.0030,0.0
75 0.0500,0.0455,0.0455,0.0500,0.0455,0.0455,0.0032,0.0024,0.0023,0.0032,0.0024,0.0023,0.0033,0.0026,0.0026,0.0033,0.0026,0.0026,0.0
76 0.0500,0.0500,0.0500,0.0500,0.0500,0.0500,0.0031,0.0024,0.0023,0.0031,0.0024,0.0023,0.0029,0.0024,0.0023,0.0029,0.0024,0.0023,0.0
77 0.0556,0.0500,0.0500,0.0556,0.0500,0.0500,0.0027,0.0022,0.0021,0.0027,0.0022,0.0021,0.0034,0.0028,0.0027,0.0034,0.0028,0.0027,0.0
78 0.0417,0.0417,0.0417,0.0417,0.0417,0.0417,0.0021,0.0019,0.0019,0.0021,0.0019,0.0019,0.0038,0.0029,0.0029,0.0038,0.0029,0.0029,0.0
79 0.0556,0.0556,0.0556,0.0556,0.0556,0.0556,0.0026,0.0021,0.0020,0.0026,0.0021,0.0020,0.0038,0.0033,0.0033,0.0038,0.0033,0.0033,0.0
80 0.0455,0.0455,0.0455,0.0455,0.0455,0.0455,0.0015,0.0010,0.0010,0.0015,0.0010,0.0010,0.0014,0.0011,0.0010,0.0014,0.0011,0.0010,0.0
81 0.0455,0.0455,0.0455,0.0455,0.0455,0.0455,0.0024,0.0019,0.0018,0.0024,0.0019,0.0018,0.0037,0.0032,0.0032,0.0037,0.0032,0.0032,0.0
82 0.0625,0.0625,0.0625,0.0625,0.0625,0.0625,0.0023,0.0018,0.0017,0.0023,0.0018,0.0017,0.0037,0.0030,0.0029,0.0037,0.0030,0.0029,0.0
83 0.0417,0.0417,0.0417,0.0417,0.0417,0.0417,0.0029,0.0022,0.0022,0.0029,0.0022,0.0022,0.0023,0.0019,0.0019,0.0023,0.0019,0.0019,0.0
84 0.0385,0.0385,0.0385,0.0385,0.0385,
```

- Kategori Food - Threshold 0.5

[illegible]

- Kategori Tren - Threshold 0.45

[illegible]

- Kategori Tren - Threshold 0.5

[illegible]