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

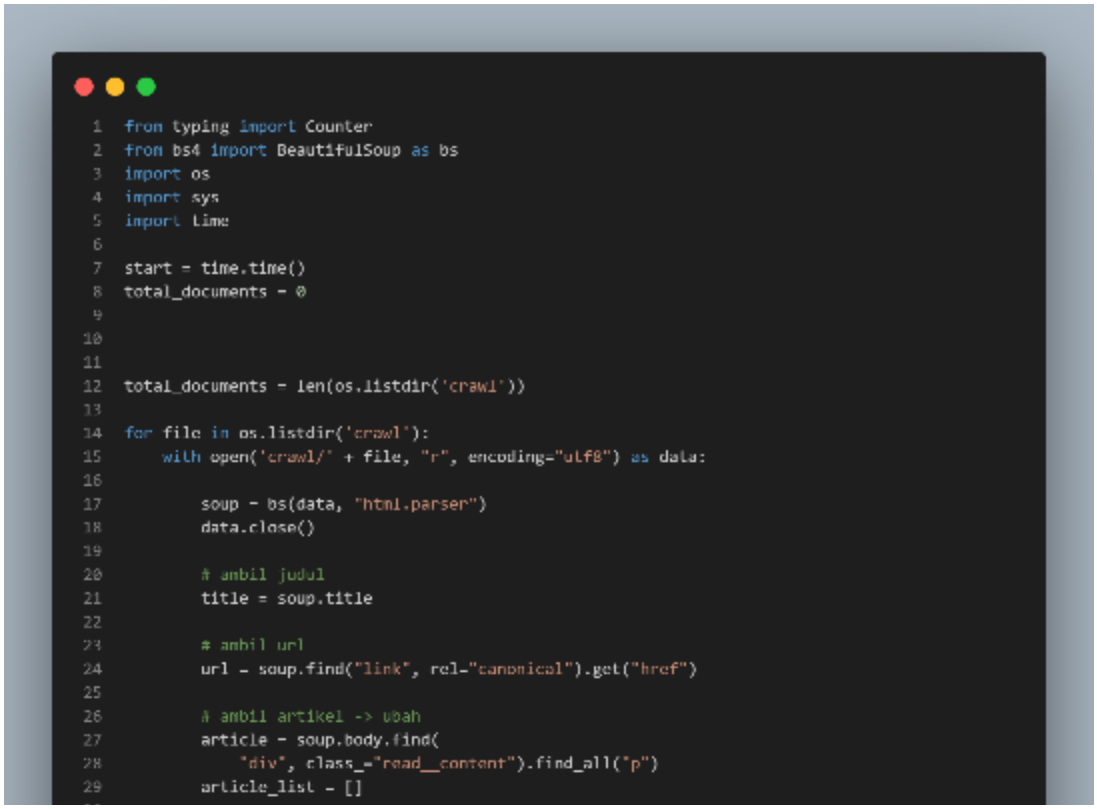
Pada Tugas 3 Text dan Web Mining meminta untuk membangun fitur dengan menggunakan halaman web dari dua kategori. Disini, kategori yang digunakan adalah kategori food dan tren dari portal berita detik. Terdapat 12000 berita yang digunakan untuk membangun fitur, dimana masing-masing kategori memiliki 6000 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.



```python
from typing import Counter
from bs4 import BeautifulSoup as bs
import os
import sys
import time

start = time.time()
total_documents = 0



total_documents = len(os.listdir('crawl'))

for file in os.listdir('crawl'):
    with open('crawl/' + file, "r", encoding="ulf8") as data:

        soup = bs(data, "html.parser")
        data.close()

        # ambil judul
        title = soup.title

        # ambil url
        url = soup.find("link", rel="canonical").get("href")

        # ambil artikel -> ubah
        article = soup.body.find(
            "div", class_="read__content").find_all("p")
        article_list = []
```

```
        junk_words = ["KOMPAS.com", "-", "Baca:", "Baca juga:", "Baca Juga:", "(@fnci.bunn)",
                      "Baca Juga:", "Baca Juga:\xa0", "Baca juga:\xa0"]
        # hapus tag <a> dan tag <strong> yang isinya tidak diperlukan
        for p in article:
            strong = p.find("strong")
            if(strong):
                for x in junk_words:
                    if(x in strong.contents):
                        strong.decompose()
            article_list.append(p.text)

        article_string = "".join(article_list)
        article_string = article_string.replace("KOMPAS.com   ", "")
        article_string = article_string.replace("KOMPAS.com- ", "")
        article_string = article_string.replace("JAKARTA,", "")
        article_string = article_string.replace("Baca\xa0juga:", "")
        article_string = article_string.replace("Baca"" ""juga:\xa0", "")
        article_string = article_string.replace("Baca"" ""juga:\xa0", "")
        article_string = article_string.replace("baca:\xa0 ", "")
        article_string = article_string.replace("DEPOK, ", "")

        # bagi artikel
        bagian1 = []
        bagian2 = []
        bagian3 = []
        bagian4 = []
        bagian5 = []
        length = len(article_string)

        # presentase jumlah karakter per bagian (top, mid, bottom)
        part1 = round(20/100*length)
        #part2 = round(43/100*length)

        # cari titik
        extra = 0

        #bagian 1
        try:
            for x in range(part1):
                bagian1.append(article_string[x])
                while(bagian1 and x >= part1-1):
                    bagian1.append(article_string[x+1])
                    x += 1
                    extra += 1
                    if(bagian1[-1] == "." and not article_string[x+1].isdigit()):
                        break
        except:
            print("Something is wrong")

        extra2 = 0
        #bagian 2
        try:
            for x in range((part1 + extra), (part1 * 2)):
                bagian2.append(article_string[x])
                while(bagian2 and x >= ((part1 *2) - 1)):
                    bagian2.append(article_string[x+1])
                    x += 1
                    extra2 += 1
                    if(bagian2[-1] == "." and not article_string[x+1].isdigit()):
                        break
        except:
            print("Something is wrong")

        extra3 = 0
        #bagian 3
        try:
            for x in range(((part1 * 2) + extra2), (part1 * 3)):
                bagian3.append(article_string[x])
                while(bagian3 and x >= ((part1 * 3) - 1)):
                    bagian3.append(article_string[x+1])
                    x += 1
```

```
                    extra3 += 1
                    if(bagian3[-1] == "." and not article_string[x+1].isdigit()):
                        break
        except:
            print("Something is wrong")

        extra4 = 0
        #bagian 4
        try:
            for x in range(((part1 * 3) + extra3), (part1 * 4)):
                bagian4.append(article_string[x])
                while(bagian4 and x >= ((part1 * 4) - 1)):
                    bagian4.append(article_string[x+1])
                    x += 1
                    extra4 += 1
                    if(bagian4[-1] == "." and not article_string[x+1].isdigit()):
                        break
        except:
            print("Something is wrong")

        #bagian 5
        try:
            for x in range(((part1 * 4) + extra4), length):
                bagian5.append(article_string[x])
                while(bagian5 and x >= (length-1) and bagian5[-1] != '.'):
                    bagian5.pop(-1)

        except:
            print("Something is wrong")

    with open("clean/" +file[:-5]+".clean.dat", 'w', encoding="utf8") as scrapped_html:

        scrapped_html.write("<title>" + title.get_text() + "</title>\n\n")

        scrapped_html.write("<bagian1>")
        for text in bagian1:
            scrapped_html.write(text)
        scrapped_html.write("</bagian1>\n\n")

        scrapped_html.write("<bagian2>")
        for text in bagian2:
            scrapped_html.write(text)
        scrapped_html.write("</bagian2>\n\n")

        scrapped_html.write("<bagian3>")
        for text in bagian3:
            scrapped_html.write(text)
        scrapped_html.write("</bagian3>\n\n")

        scrapped_html.write("<bagian4>")
        for text in bagian4:
            scrapped_html.write(text)
        scrapped_html.write("</bagian4>\n\n")

        scrapped_html.write("<bagian5>")
        for text in bagian5:
            scrapped_html.write(text)
        scrapped_html.write("</bagian5>")

    scrapped_html.close()

    os.system('cls')


print("Complete.")
```

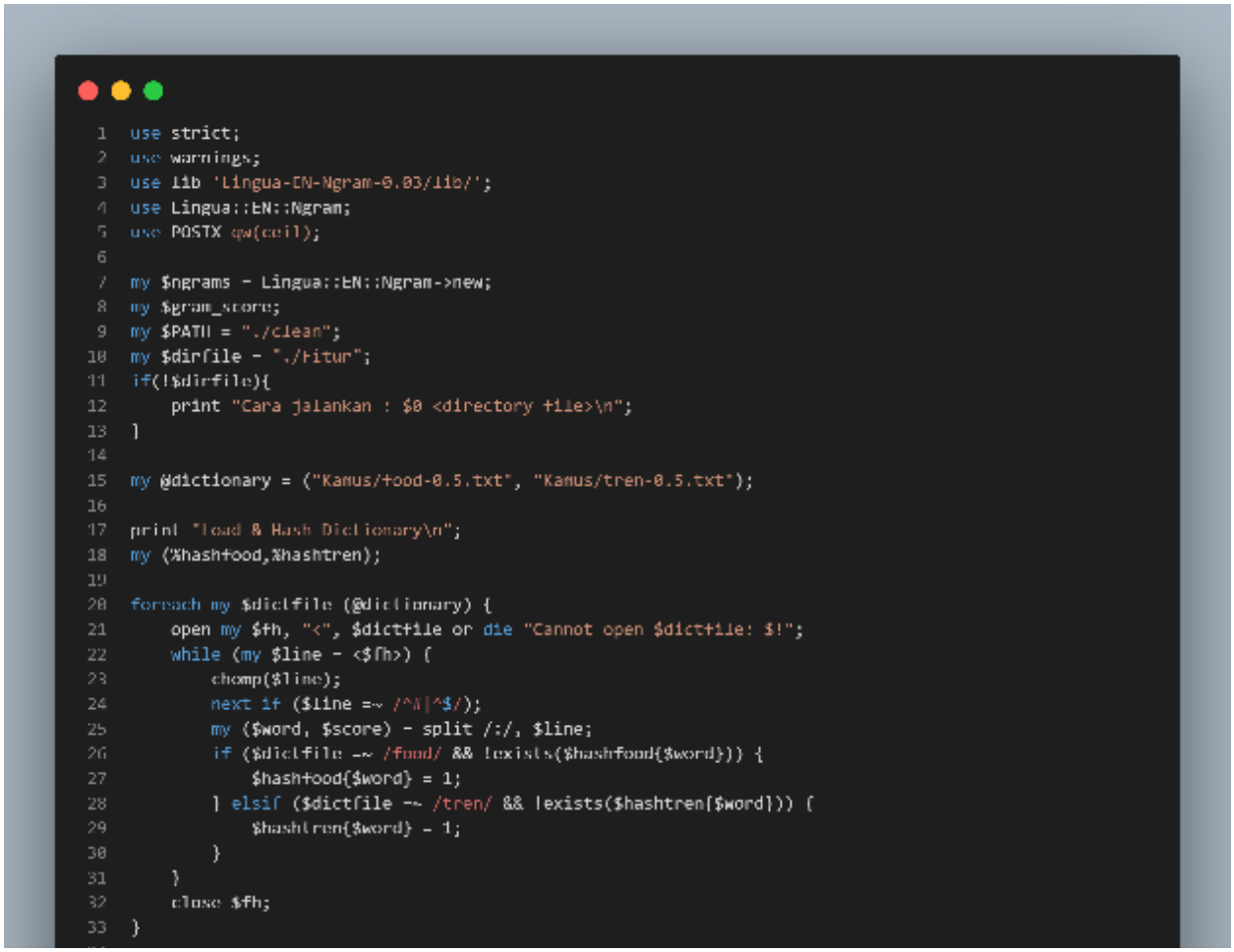
Membangun Fitur

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.



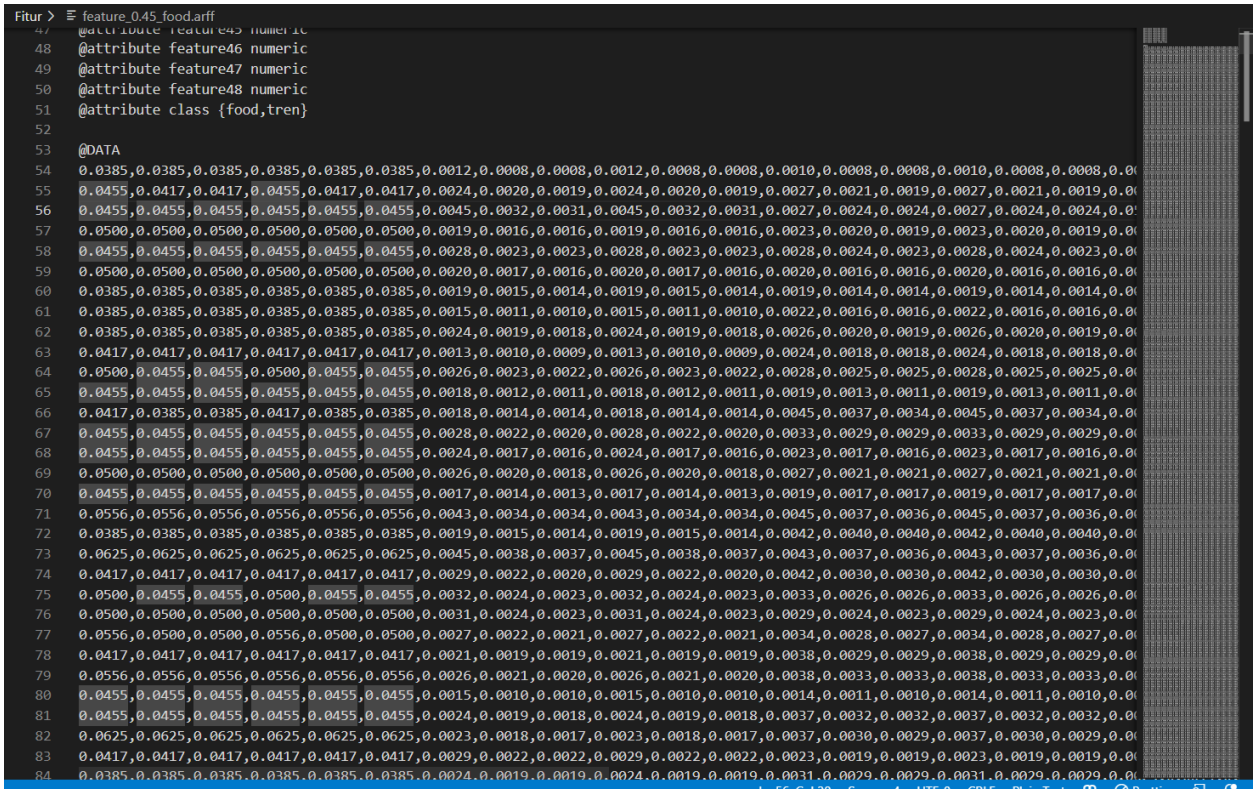
```
use strict;
use warnings;
use lib 'Lingua-EN-Ngram-0.03/lib/';
use Lingua::EN::Ngram;
use POSIX qw(ceil);

my $ngrams = Lingua::EN::Ngram->new;
my $gram_score;
my $PATH = "./clean";
my $dirfile = "./fitur";
if(!$dirfile){
    print "Cara jalankan : $0 <directory file>\n";
}

my @dictionary = ("Kamus/food-0.5.txt", "Kamus/tren-0.5.txt");

print "Load & Hash Dictionary\n";
my (%hashfood,%hashtren);

foreach my $dictfile (@dictionary) {
    open my $fh, "<", $dictfile or die "Cannot open $dictfile: $!";
    while (my $line = <$fh>) {
        chomp($line);
        next if ($line =~ /^\|^$/);
        my ($word, $score) = split /:/, $line;
        if ($dictfile =~ /food/ && !exists($hashfood{$word})) {
            $hashfood{$word} = 1;
        } elsif ($dictfile =~ /tren/ && !exists($hashtren{$word})) {
            $hashtren{$word} = 1;
        }
    }
    close $fh;
}
```

```
my $process=0;
foreach my $dir(("$PATH/food","$PATH/tren")){
    my @srcfile = split /\//,$dir;
    open OUT,"> $dirfile/feature_0.5_$srcfile[2].dat" or die "Can't open file...";
    open ARFF,"> $dirfile/feature_0.5_$srcfile[2].arff" or die "Can't open file...";
    print ARFF "\@relation feature 0.5 $srcfile[2]\n\n";
    foreach my $iterate (1 .. 48){
        print ARFF "\@attribute feature$iterate numeric\n";
    }
    print ARFF "\@attribute class {food,tren}\n\n\@DATA\n";

    my @files = glob("$dir/*.clean.dat");

    print $dir;
    my $number = 1;

    foreach my $file(@files){
        $process++;

        open F, $file or die "Can't open input: $!\n";
        my $text = do { local $/; <F> };
        close F;
        $text =~ s/\n+//gs;

        if($file =~ /food/){
            print OUT "food ";
        }else{
            print OUT "tren ";
        }

        my @sections = (
            get_text($text,"title"),
            get_text($text,"bagian1"),
            get_text($text,"bagian2"),
            get_text($text,"bagian3"),
            get_text($text,"bagian4"),
            get_text($text,"bagian5"),
            );
        my @weight = (1, 0.2, 0.2, 0.2, 0.2, 0.2);
        for (my $sec = 0 ; $sec < @sections; $sec++){
            for (my $dict = 0 ; $dict < @dictionary ; $dict++){
                foreach my $n (1 .. 3){
                    $gram_score=0;
                    if($dict == 0){
                        $gram_score = count_section($sections[$sec], \%hashfood, $n) * $weight[$sec];
                    }elsif($dict == 1){
                        $gram_score = count_section($sections[$sec], \%hashtren, $n)* $weight[$sec];
                    }

                    print ARFF sprintf("%.4f",$gram_score).",";
                    print OUT "$number:".sprintf("%.4f",$gram_score)." " ;
                    $number++;
                }
            }
        }
        if($file =~ /food/){
            print ARFF "food\n";
        }else{
            print ARFF "tren\n";
        }
        print OUT "\n";
        $number = 1;
        if($process % 1000 == 0){
            print "\nDone : $process\n"
        }
```

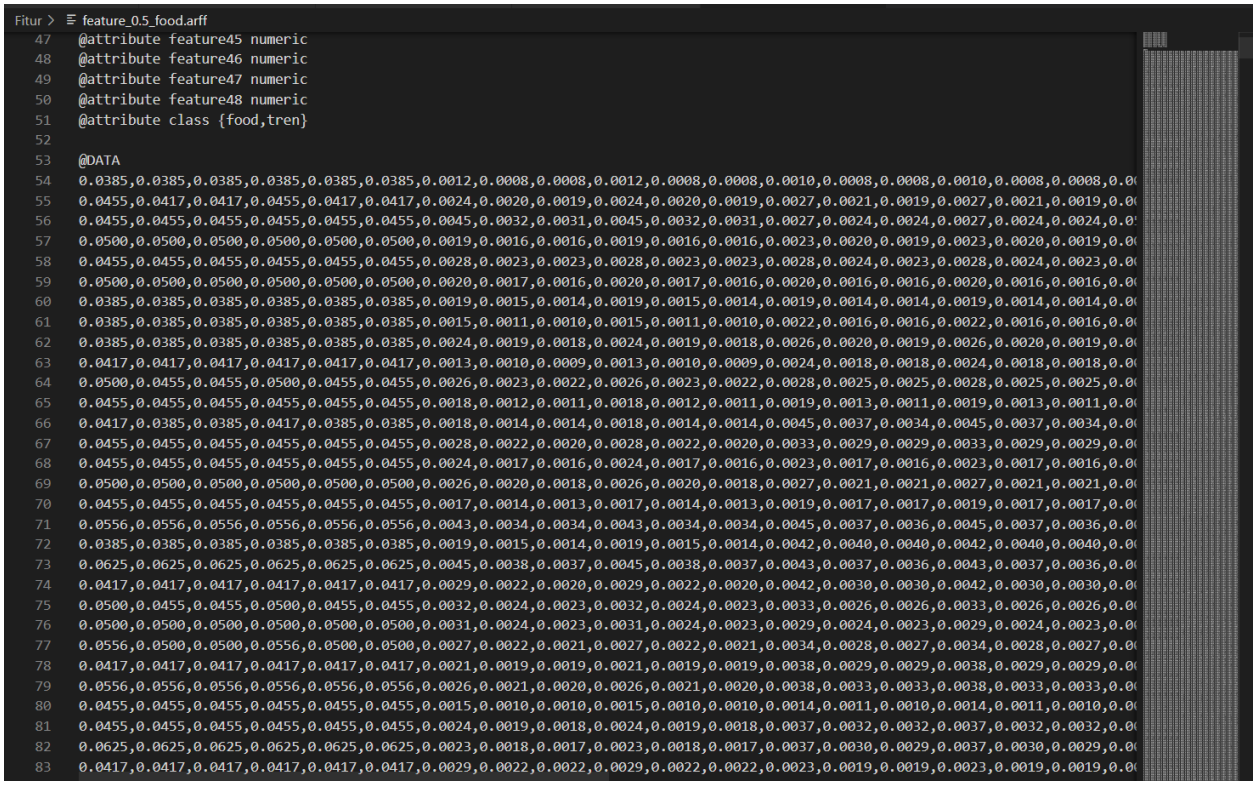
```
    }
    close OUT;
    close ARFF;
}

sub get_text{
    my ($text,$regex) = @ ;

    if($text =~ /<$regex>(.*?)<\/$regex>/){
        return $1;
    }
}

sub count_section {
    my($section, $hash, $n) = @_;
    my $count=0;
    my $sect = clean_string($section);

    $ngrams->text($sect.".");
    my $lxl = $ngrams >lexl;
    my $grams = $ngrams->ngram($n);
    my $gramlen = keys %$grams;

    foreach my $gram ( sort { $$grams{ $b } <=> $$grams{ $a } } keys %$grams ) {
        $gramlen++;
        chomp($gram);

        next if($gram =~ /^#/ || $gram =~ /^$/);

        if (defined $$hash{$gram}) {
            $count++;
        } else {
            $count = 1;
        }
    }

    if ($count == 0) {
        return 0;
    } else {
        return ($count/$gramlen);
    }
}

sub clean_string {
    my $file = shift;
    $file =~ s/<.*?>//g;
    $file =~ s/\s\w+=.*?>/ /g;
    $file =~ s/>//g;
    $file =~ s/&.*?;//g;
    $file =~ s/[\:\]\|\[\?\!\@\#\$\%\*\&\,\/\\\(\)\;]+//g;
    $file =~ s/-/ /g;
    $file =~ s/\s+/ /g;
    $file = lc($file);
    return $file;
}
```

Hasil Fitur Yang Dibangun

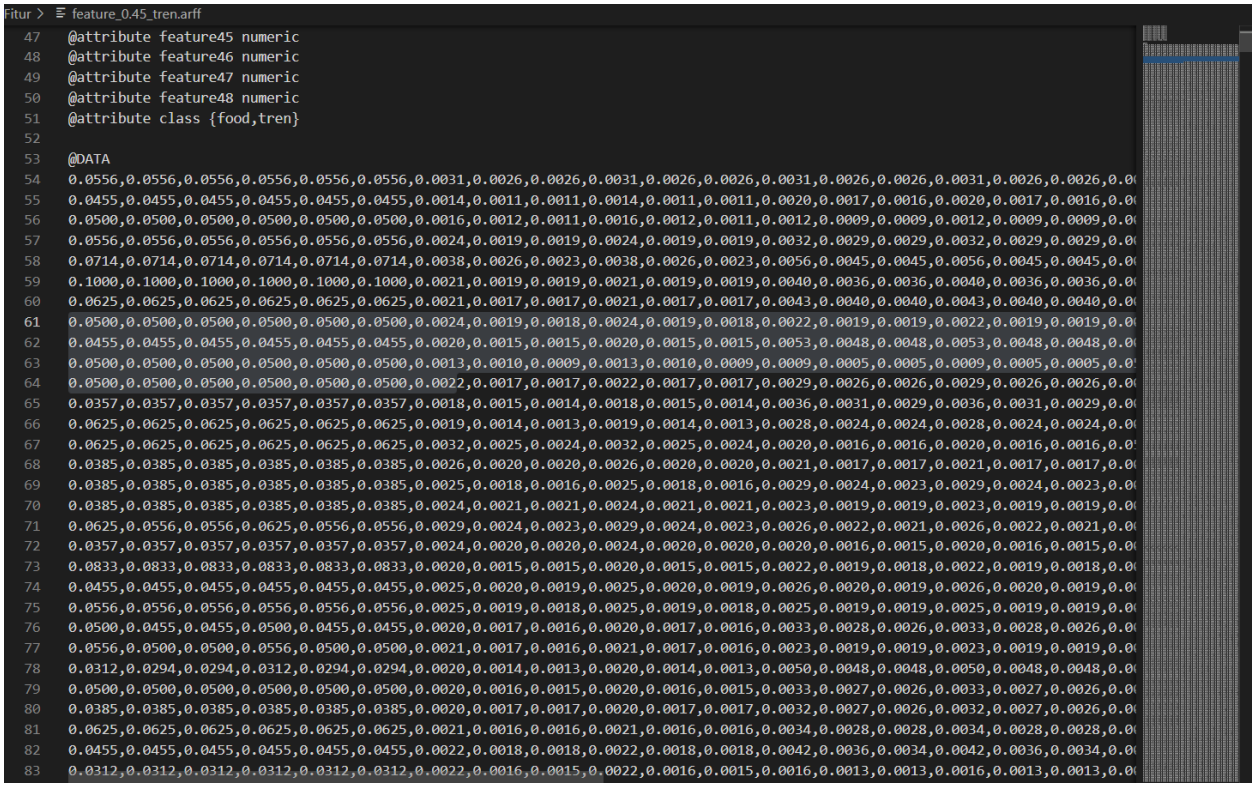
- Kategori Food - Threshold 0.45



- Kategori Food - Threshold 0.5



- Kategori Tren - Threshold 0.45



- Kategori Tren - Threshold 0.5