

**TUGAS BESAR 2**

**IF2123 ALJABAR LINEAR DAN GEOMETRI**

**SEMESTER 1 2020/2021**



**Oleh**

**Kelompok 34**

13519022	Jose Galbraith Hasintongan
13519062	Feralezer L. G. Tampubolon
13519103	Bryan Rinaldo

**PROGRAM STUDI TEKNIK INFORMATIKA**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**2020**

## DAFTAR ISI

Daftar Isi .....	
Bab1:Deskripsi Masalah.....	
Bab 2: Teori Singkat .....	
Bab 3: Implementasi program .....	
Bab 4: Eksperimen .....	
Bab 5: Kesimpulan, saran, dan refleksi.....	
Daftar pustaka .....	

# **Bab 1**

## **Deskripsi Masalah**

Membuat program mesin pencarian dengan sebuah website lokal sederhana. Spesifikasi program adalah sebagai berikut:

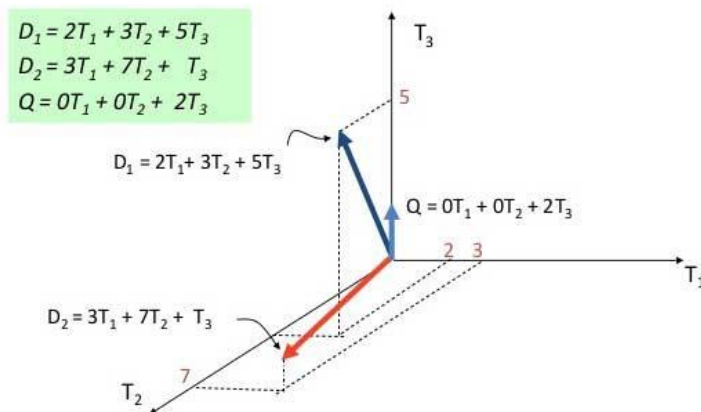
1. Program mampu menerima search query. Search query dapat berupa kata dasar maupun berimbuhan.
2. Dokumen yang akan menjadi kandidat dibebaskan formatnya dan disiapkan secara manual. Minimal terdapat 15 dokumen berbeda sebagai kandidat dokumen. Bonus: Gunakan web scraping untuk mengekstraksi dokumen dari website.
3. Hasil pencarian yang terurut berdasarkan similaritas tertinggi dari hasil teratas hingga hasil terbawah berupa judul dokumen dan kalimat pertama dari dokumen tersebut. Sertakan juga nilai similaritas tiap dokumen.
4. Program disarankan untuk melakukan pembersihan dokumen terlebih dahulu sebelum diproses dalam perhitungan cosine similarity. Pembersihan dokumen bisa meliputi hal-hal berikut ini.
  - Stemming dan Penghapusan stopwords dari isi dokumen.
  - Penghapusan karakter-karakter yang tidak perlu.
5. Program dibuat dalam sebuah website lokal sederhana. Dibebaskan untuk menggunakan framework pemrograman website apapun. Salah satu framework website yang bisa dimanfaatkan adalah Flask (Python), ReactJS, dan PHP.
6. Kalian dapat menambahkan fitur fungsional lain yang menunjang program yang anda buat (unsur kreativitas diperbolehkan/dianjurkan).
7. Program harus modular dan mengandung komentar yang jelas.
8. Dilarang menggunakan library cosine similarity yang sudah jadi.

## Bab 2

### Teori Dasar

Information retrieval atau Sistem temu balik adalah menemukan(biasanya dokumen) dari sebuah ketidakstrukturan yang alami(biasanya teks) untuk memenuhi sebuah kebutuhan informasi dari koleksi yang berukuran besar(biasanya disimpan pada komputer). Sistem information retrieval atau sistem temu balik informasi bertujuan untuk mencukupi kebutuhan informasi pengguna dengan sumber informasi yang tersedia sesuai dengan situasi. Penulis mempresentasikan ide dan pikiran mereka ke dalam sebuah dokumen, pencari dokumen mencari sebuah dokumen di dalam sekumpulan dokumen dimana pencari tersebut tidak mengetahui dengan pasti bagaimana cara menemukan dan mengenali dokumen yang tepat sesuai dengan kebutuhannya, sistem temu balik informasi mempertemukan ide yang ditulis penulis tersebut dengan kebutuhan informasi yang dibutuhkan oleh pencari dokumen tersebut yang dinyatakan kedalam pernyataan (query).

Kemiripan kalimat memainkan peran penting pada berbagai penelitian yang berhubungan dengan teks dan aplikasi. Vector Space Model (VSM) digunakan sebagai representasi dari kumpulan dataset dokumen teks. Dokumen dalam Vector Space Model (VSM) berupa matriks yang berisi bobot seluruh kata pada tiap dokumen. Bobot tersebut menyatakan kepentingan atau kontribusi kata terhadap suatu dokumen dan kumpulan dokumen. Kepentingan suatu kata dalam dokumen dapat dilihat dari frekuensi kemunculannya terhadap dokumen.



Gambar diatas menunjukkan pemodelan dokumen teks di ruang dimensi dimana (D) adalah kalimat dokumen sedangkan (T) adalah term atau kata. Untuk mendapatkan nilai jarak atau kemiripan dokumen, dapat menggunakan berbagai macam varian rumus perhitungan jarak diantaranya adalah (1) Cosine, (2) Jaccard, (3) Dice, (3) Euclidean, (4) Manhattan, (5) Minkowski, (6) Mahalanobis, (8) Weighted. Dalam tugas ini kami akan menggunakan metode Cosine.

*Cosine Similarity* dapat diimplementasikan untuk menghitung nilai kemiripan antar kalimat dan menjadi salah satu teknik untuk mengukur kemiripan teks yang populer. Contoh penggunaan *Cosine Similarity* dalam menguji kemiripan dua buah kalimat adalah sebagai berikut:

Misalkan diberikan dua buah kalimat yaitu kalimat A dan B, yaitu:

A : Julie loves me more than Linda loves me

B : Jane likes me more than Julie loves me

Uji kemiripan teks dapat dilakukan dengan rumus:

$$\frac{\sum_{n=1}^j (n_A \times n_B)}{\sqrt{\sum_{n=1}^j (n_A)^2} \times \sqrt{\sum_{n=1}^j (n_B)^2}}$$

*Rumus Cosine Similarity*

Dengan:  $j = |A \cap B|$  Kemiripan =

$n_A$  = jumlah kemunculan kata indeks ke-n dari daftar kata pada kalimat A.

$n_B$  = jumlah kemunculan kata indeks ke-n dari daftar kata pada kalimat B.

Indeks	Daftar Kata	Jumlah Kemunculan Kata	
		A	B
1	Julie	1	1
2	loves	2	1
3	me	2	2
4	more	1	1
5	than	1	1
6	Linda	1	0
7	Jane	0	1
8	likes	0	1

Berdasarkan rumus tersebut di atas dilakukan penghitungan seperti di bawah ini. Dengan Tingkat kemiripan teks =

$$= \frac{(1 \times 1) + (2 \times 1) + (2 \times 2) + (1 \times 1) + (1 \times 1) + (1 \times 0) + (0 \times 1) + (0 \times 1)}{\sqrt{1^2 + 2^2 + 2^2 + 1^2 + 1^2 + 1^2 + 0^2 + 0^2} \times \sqrt{1^2 + 1^2 + 2^2 + 1^2 + 1^2 + 0^2 + 1^2 + 1^2}}$$

$$= 0.821584$$

## Bab 3

### Implementasi Masalah

1. Backend website, import library untuk file handling, webscraping, string formatting, dan untuk membuat hash table

```
# Backend Website
from flask import Flask, render_template, request, redirect, url_for, abort, send_from_directory
from werkzeug.utils import secure_filename
# File handling
import os
import shutil
# Web Scraping
import requests
from bs4 import BeautifulSoup
# String formatting
import string
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
# Membuat hash table
import collections
from copy import deepcopy

app = Flask(__name__)
app.config["MAX_CONTENT_LENGTH"] = 5120 * 5120
app.config["UPLOAD_EXTENSIONS"] = [".txt"]
app.config["UPLOAD_PATH"] = "../test"
```

2. Melakukan format string dengan 3 langkah

```
# function FormatString(s : string) → string
# Memformat string dalam 3 langkah
def FormatString(s):
    s = StemmerFactory().create_stemmer().stem(s)
    # Double-check: Ubah string menjadi lowercase
    s = s.lower()
    # Double-check: Hilangkan punctuation
    for char in string.punctuation:
        s = s.replace(char, " ")
    # Double-check: Hilangkan duplicate whitespace, → s
    return " ".join(s.split())
```

3. Mengubah string ke dalam Counter

```
# function FrequencyCounter(s : string) → Counter
# Mengubah string ke dalam Counter, Counter adalah hash table dengan format <key: kata, value: jumlah kemunculan kata pada string>
def FrequencyCounter(s):
    # Ubah string ke dalam list kata demi kata
    s_list = s.split()
    # Ubah list menjadi Counter, → Counter
    return collections.Counter(s_list)
```

#### 4. Mengambil f dari tuple (k, v)

```
# function GetCosineSimilarity((k : keytype, v : [f : float, i : integer, s : string])) → f
# Mengambil f dari tuple (k, v)
def GetCosineSimilarity(kv):
    return kv[1][1]
```

#### 5. Mengembalikan array yang berisi daftar filename pada direktori

```
# function GetFileNames() → array of string
# Mengembalikan array yang berisi daftar filename pada direktori ../test
def GetFileNames():
    items = os.listdir(app.config["UPLOAD_PATH"])
    for item in items:
        if os.path.isdir(os.path.join(app.config["UPLOAD_PATH"], item)):
            items.remove(item)
    return items
```

#### 6. Mengupdate variabel-variabel global

```
# procedure UpdateDatabase()
# Meng-update variabel-variabel global
def UpdateDatabase():

    # Ambil daftar dokumen/berita
    FILENAMES = GetFileNames()

    # Iterasi berita satu demi satu
    for filename in FILENAMES:

        # Cek apakah file sudah pernah dimasukkan ke database
        if filename not in SEARCH_RESULTS:

            # Tambahkan berita ke dalam daftar hasil pencarian
            SEARCH_RESULTS["$s" % filename] = ["*", 0.0, 0, "*"]

            # Ubah berita ke dalam bentuk string
            with open(os.path.join(app.config["UPLOAD_PATH"], filename), "r", encoding="utf-8") as f:
                f_string = f.read()

            # Tambahkan judul berita ke dalam hasil pencarian
            f.seek(0)
            SEARCH_RESULTS["$s" % filename][0] = f.readline()
```



```

# Tambahkan kalimat pertama berita ke dalam hasil pencarian
# Ada 6 kemungkinan separator kalimat pertama: ". ", ".\n", "? ", "?\n", "! ", dan "!\n"
# Pilih yang paling pendek sebagai kalimat utama
s2 = []
s2.append(f_string.split(". ", 1)[0] + ".")
s2.append(f_string.split(".\n", 1)[0] + ".")
s2.append(f_string.split("? ", 1)[0] + "?")
s2.append(f_string.split("?\\n", 1)[0] + "?")
s2.append(f_string.split("! ", 1)[0] + "!")
s2.append(f_string.split("\\n", 1)[0] + "!")
if len(s2[0]) == len(s2[1]) == len(s2[2]) == len(s2[3]) == len(s2[4]) == len(s2[5]):
    SEARCH_RESULTS["%s" % filename][3] = s2[0][: -1]
else:
    SEARCH_RESULTS["%s" % filename][3] = min(s2, key=len)

# Format isi berita agar bisa dibuat vektornya
f_formattedstring = FormatString(f_string)

# Tambahkan berita ke dalam daftar vektor
VECTORS["%s" % filename] = FrequencyCounter(f_formattedstring)

# Tambahkan jumlah kata pada berita ke dalam hasil pencarian
SEARCH_RESULTS["%s" % filename][2] = sum(VECTORS["%s" % filename].values())

```

## 7. Variabel-variabel global

```

# FILENAMES : array of string
# Daftar file yang ada pada database
FILENAMES = GetFileNames()

# VECTORS : {key: string, value: Counter()}
# Daftar vektor yang akan dipakai ketika perhitungan (akan dikirim ke frontend untuk membuat tabel)
VECTORS = {}
# Cara penambahan elemen: VECTORS["key"] = FrequencyCounter(s : string) di mana s adalah string yang ingin diubah ke dalam vektor frekuensi kata

# SEARCH_RESULTS : {key: string, value: [s1 : string, f : float, i : integer, s2 : string]}
# Data yang akan dikirim kembali ke frontend, berisi informasi mengenai hasil pencarian
SEARCH_RESULTS = {}
# key = nama file berita
# s1 = judul berita
# f = hasil cosine similarity berita
# i = jumlah kata pada berita
# s2 = kalimat pertama pada berita
# Cara penambahan elemen: search_results["key"] = [s1 : string, f : float, i : integer, s2 : string]

# Inisialisasi database
UpdateDatabase()

```

## 8. Render html index

```

@app.route("/")
@app.route("/index")
def index():
    FILENAMES = GetFileNames()
    return render_template("index.html", FILENAMES=FILENAMES)

```

## 9. Fungsi untuk file (.txt )

```
@app.route("/addfile", methods=["POST"])
def addfile():

    # Update daftar file
    FILENAMES = GetFileNames()

    # File yang dikirim mungkin lebih dari satu, loop berikut akan meng-iterasi file-file tersebut satu demi satu
    for txtfile in request.files.getlist("txtfile"):

        # Nama file diset supaya bisa disave di OS Linux
        filename = secure_filename(txtfile.filename)

        # Tidak akan terjadi apa-apa jika nama file tidak valid
        if filename != "":

            # Save file yang diupload dengan nama sementara
            txtfile.save(os.path.join(app.config["UPLOAD_PATH"], ".tmp/temp.txt"))

            # File akan disave
            # Mungkin file tersebut merupakan duplikat, mungkin file tersebut unik tapi namanya sudah dipakai oleh file lain pada database
            # Perlu dilakukan penanganan khusus untuk kasus-kasus ini
            i = 1
            isSaved = False
            while not isSaved:

                # Cek apakah file sudah ada di database
                if filename not in FILENAMES:

                    # Pastikan loop tidak akan diulang
                    isSaved = True

                    # Ambil extension file (.txt), dan kirimkan pesan error 400 ke pengguna jika extension file bukan .txt
                    ext = os.path.splitext(filename)[1]
                    if ext not in app.config["UPLOAD_EXTENSIONS"]:
                        abort(400)

                    # Move file ke ../test/
                    shutil.move((os.path.join(app.config["UPLOAD_PATH"], ".tmp/temp.txt")), (os.path.join(app.config["UPLOAD_PATH"], filename)))

                    # Update database
                    UpdateDatabase()

                    # Jika file bernama sama sudah ada,
                    else:

                        # Buka kedua file
                        with open(os.path.join(app.config["UPLOAD_PATH"], ".tmp/temp.txt")) as newfile:
                            newtxt = newfile.read()
                        with open(os.path.join(app.config["UPLOAD_PATH"], filename)) as existingfile:
                            existingtxt = existingfile.read()

                        # Cek apakah file yang sudah diupload merupakan file duplikat. Jika ya, file tidak akan disave
                        if newtxt == existingtxt:
                            isSaved = True
                            os.remove(os.path.join(app.config["UPLOAD_PATH"], ".tmp/temp.txt"))
                        else:
                            filename = secure_filename(os.path.splitext(txtfile.filename)[0] + "_" + str(i) + os.path.splitext(txtfile.filename)[1])
                            i += 1

            # Refresh web page
            return redirect(url_for("index"))
```

## 10. Inisialisasi untuk addurl (webscraping)

```
@app.route("/addurl", methods=["POST"])
def addurl():

    # Update daftar file
    FILENAMES = GetFileNames()

    # Ambil link yang dikirim
    link = request.form.get("urlfile")

    # Buka link
    doc = requests.get(link)

    # Ambil kode html dari link yang dikirim
    soup = BeautifulSoup(doc.text, "html.parser")
```

## 11. Webscraping pada website cnnindonesia.com

```
# Web Scraping
# Saat ini men-support cnnindonesia.com, medcom.id, kompas.com
# Untuk link cnnindonesia.com
if "cnnindonesia.com" in link:

    # Ambil elemen berita
    content_detail = soup.find("div", {"class": "content_detail"})

    # Ambil judul berita
    title = content_detail.find("h1", {"class": "title"}).get_text().strip()

    # Ambil info berita
    date = content_detail.find("div", {"class": "date"}).get_text().strip()

    # Ambil body berita
    detikdetailtext = content_detail.find(id="detikdetailtext")

    # Antarparagraf pada berita dipisahkan oleh <p></p>, buat list yang isinya adalah [paragraf 1, paragraf 2, dst.]
    list_paragraph = detikdetailtext.find_all("p")
    paragraph = []
    for i in range(len(list_paragraph)):

        # String formatting: tambahkan \n ke tiap akhir paragraf
        paragraph.append(list_paragraph[i].text.strip() + "\n")

        # String formatting: hilangkan \xa0
        paragraph[i] = paragraph[i].replace(u"\xa0", u" ")

    # Save berita ke file .txt
    with open(os.path.join(app.config["UPLOAD_PATH"], secure_filename(title)) + ".txt", "w", encoding="utf-8") as f:
        f.write(title + "\n")
        f.write(date + "\n\n")
        f.writelines(paragraph)
```



## 12. Webscraping pada website medcom.id

```
# Untuk link medcom.id
elif "medcom.id" in link:

    # Ambil elemen berita
    article_ct = soup.find("div", {"class": "article_ct"})

    # Ambil judul berita
    title = article_ct.find("h1").get_text().strip()

    # Ambil info berita
    info_ct = article_ct.find("div", {"class": "info_ct"}).get_text().strip()

    # Ambil body berita
    articleBody = article_ct.find(attrs={"class": "text", "itemprop": "articleBody"})

    # medcom.id menaruh advertisement di tengah berita. Ambil advertisementnya
    parallax_ads = articleBody.find(attrs={"class": "parallax_ads"})

    # Hapus advertisement dari berita,
    # String formatting: hapus \xa0, hapus \r, ganti " menjadi "
    article = articleBody.get_text().replace(parallax_ads.get_text(), "").strip().replace(u"\xa0", u" ").replace("\r", "").replace("'", "\'")

    # String formatting: hapus \n berganda
    while "\n\n" in article:
        article = article.replace("\n\n", "\n")

    # Save berita ke file .txt
    with open(os.path.join(app.config["UPLOAD_PATH"], secure_filename(title)) + ".txt", "w", encoding="utf-8") as f:
        f.write(title + "\n")
        f.write(info_ct + "\n\n")
        f.write(article)
```

## 13. Webscraping pada website kompas.com

```
# Untuk link kompas.com
elif "kompas.com" in link:

    # Ambil elemen berita
    container_clearfix = soup.find("div", {"class": "container clearfix"})

    # Ambil judul berita
    read_title = container_clearfix.find("h1", {"class": "read_title"}).get_text().strip()

    # Ambil info berita
    read_time = container_clearfix.find("div", {"class": "read_time"}).get_text().strip()

    # Ambil body berita
    read_content = container_clearfix.find("div", {"class": "read_content"})

    # Antarparagraf pada berita dipisahkan oleh <p></p>, buat list yang isinya adalah [paragraf 1, paragraf 2, dst.]
    list_paragraph = read_content.find_all("p")
    paragraph = []
    for i in range(len(list_paragraph)):

        # String formatting: tambahkan \n ke tiap akhir paragraf
        paragraph.append(list_paragraph[i].text.strip() + "\n")

        # String formatting: hilangkan \xa0
        paragraph[i] = paragraph[i].replace(u"\xa0", u" ")

    # Save berita ke file .txt
    with open(os.path.join(app.config["UPLOAD_PATH"], secure_filename(read_title)) + ".txt", "w", encoding="utf-8") as f:
        f.write(read_title + "\n")
        f.write(read_time + "\n\n")
        f.writelines(paragraph)
```

14. Jika terdapat link selain cnn, medcom, atau Kompas maka akan memberikan pesan error

```
# Untuk link lainnya, kirim error 400
else:
    abort(400)

# Update database
UpdateDatabase()

# Refresh web page
return redirect(url_for("index"))
```

15. Mengambil query dan inisialisasi hash table untuk daftar vektor dan result

```
@app.route("/search", methods=["GET"])
def search():

    # Ambil query
    query = request.args.get("q")
    query_string = FormatString(query)

    # Tangani kasus khusus (query kosong)
    if not query_string:
        return render_template("index.html", FILENAMES=FILENAMES)
    else:

        # Tambahkan query ke dalam daftar vektor
        VECTORS["query"] = FrequencyCounter(query_string)
```

16. Melakukan proses searching dengan mengubah berita ke string lalu memasukkan kedalam vektor

```
# Cosine similarity
# Jika q = vektor query dan d = vektor dokumen, maka
# cosine_similarity(q, d) = (q * d) / (||q|| ||d||)

# Hitung ||q||
query_mag = 0
for term in VECTORS["query"]:
    query_mag += (VECTORS["query"][term] ** 2)
query_mag **= 0.5

# Hitung ||d|| dan q * d
for filename in SEARCH_RESULTS:

    # ||d||
    file_mag = 0
    for term in VECTORS[filename]:
        file_mag += (VECTORS[filename][term] ** 2)
    file_mag **= 0.5

    # q * d
    cross = 0
    for term in VECTORS["query"]:
        if term in VECTORS[filename]:
            cross += (VECTORS["query"][term] * VECTORS[filename][term])

    # Hitung cosine similarity (dalam %), tambahkan ke dalam hasil pencarian
    SEARCH_RESULTS[filename][1] = (cross / (query_mag * file_mag)) * 100
```

## 17. Mengubah cosine similarity ke list lalu membuat urutan term untuk tabel

```
# Ubah cosine similarity ke dalam bentuk list dengan [(k1, [s11, f1, i1, s21]), (k2, [s12, f2, i2, s22]), dst.], f1 ≥ f2 ≥ f3 dst.
results = list(SEARCH_RESULTS.items())
results.sort(reverse=True, key=GetCosineSimilarity)

# Untuk pembuatan tabel, perlu dibuat suatu urutan term
order = VECTORS["query"].most_common()

return render_template("search.html", query=query, results=results, VECTORS=VECTORS, order=order)
```

## 18. Render html about dan display result

```
@app.route("/about")
def about():
    return render_template("about.html")

@app.route("/uploads/<path:filename>")
def display_result(filename):
    return send_from_directory(app.config["UPLOAD_PATH"], filename)
```

## 19. Base.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
    integrity="sha384-TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2" crossorigin="anonymous">
    {% block head %}{% endblock %}
  </head>

  <body>
    {% block body %}{% endblock %}
    <footer>
      <hr>
      <p class="text-center">&copy; 2020 - <a href="/about">About Us</a></p>
    </footer>
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
    integrity="sha384-Dfxdz2htPH01s5Ss5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
    crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"
    integrity="sha384-9/reFTGAW83EW2RDu2S0VKA1Zap3H66lZ81PoYlFhbGU+6BZp6G7niu735Sk7lN"
    crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/js/bootstrap.min.js"
    integrity="sha384-w1Q4orYjBQndcko6MimVbzY0tgp4pWB41Z7lr30WKz0vr/aWKhXdBNmNb5D92v7s"
    crossorigin="anonymous"></script>
  </body>
</html>
```

## 20. header.html

```
{% extends 'base.html' %}

{% block head %}
    {% block header_head %}{% endblock %}
{% endblock %}

{% block body %}
    <header>
        <div class="container-fluid pt-3 pl-3">
            <div class="row">
                <div class="col-1">
                    <a href="/"></a>
                </div>
                <div class="col-11">
                    <form class="form-inline" method="GET" action="/search">
                        <div class="form-group mx-3">
                            <input class="form-control" type="text" name="q" autofocus value="{{ query }}">
                        </div>
                        <button class="btn btn-primary btn-success" type="submit">Search</button>
                    </form>
                </div>
            </div>
            {% block header_header %}{% endblock %}
        </div>
        <hr>
    </header>
    {% block header_body %}{% endblock %}
{% endblock %}
```

## 21. index.html

```
{% extends 'base.html' %}

{% block head %}
    <title>Entah Search Engine</title>
{% endblock %}

{% block body %}
    <div class="container pt-5">
        <div class="row text-center">
            <div class="col">
                <a href=""></a>
            </div>
        </div>
        <div class="row">
            <div class="col">
                </div>
        </div>
        <div class="row">
            <div class="col">
                <ul class="nav nav-tabs">
                    <li class="nav-item">
                        <a href="#search_tab" class="nav-link active" data-toggle="tab">Search</a>
                    </li>
                    <li class="nav-item">
                        <a href="#add_tab" class="nav-link" data-toggle="tab">Upload</a>
                    </li>
                </ul>
                <br>
            </div>
        </div>
    </div>
```



```

<div class="tab-content">
  <div class="tab-pane fade show active" id="search_tab">
    <form method="GET" action="/search">
      <div class="form-group">
        <input class="form-control form-control-lg" type="text" name="q" autofocus>
      </div>
      <button class="btn btn-primary btn-lg btn-block btn-success" type="submit">Search</button>
    </form>
  </div>
  <div class="tab-pane fade" id="add_tab">
    <p>
      Ada {{ FILENAMES|length }} file pada database:<br>
      <small>{{ " ", ".join(FILENAMES) }}</small>
    </p>
    <hr>
    <form method="POST" enctype="multipart/form-data" action="/addfile">
      <div class="form-group">
        <label for="addByFile">Upload file .txt</label>
        <input type="file" class="form-control-file" id="addByFile" name="txtfile" accept=".txt" multiple>
      </div>
      <button class="btn btn-primary" type="submit">Upload</button>
    </form>
  </div>
</div>

```

```

    <form method="POST" action="/addurl">
      <div class="form-group">
        <label for="addByURL">
          <small class="text-danger">BETA!</small>&nbsp;Web Scrapping&nbsp;
          <small>(saat ini hanya bisa untuk kompas.com, medcom.id, dan cnnindonesia.com)</small>
        </label>
        <input class="form-control" id="addByURL" type="url" name="urlfile"
          placeholder="https://example.com atau http://example.com" pattern="https://.*+http://.*">
      </div>
      <button class="btn btn-primary" type="submit">Kirim</button>
    </form>
  </div>
</div>
</div>
</div>
</div>
{% block body_search %}{% endblock %}
{% endblock %}

```

## 22. search.html

```

{% extends 'header.html' %}

{% block header_head %}
  <title>{{ query }} - Entah</title>
{% endblock %}

{% block header_header %}
  <div class="row mt-3">
    <div class="col-1"></div>
    <div class="col-11">
      <ul class="nav nav-pills">
        <li class="nav-item">
          <a href="#search_results" class="nav-link active" data-toggle="tab">Result</a>
        </li>
        <li class="nav-item">
          <a href="#search_table" class="nav-link" data-toggle="tab">Table</a>
        </li>
      </ul>
    </div>
  </div>
{% endblock %}

```



```
{% block header_body %}
<div class="container-fluid">
  <div class="tab-content">
    <div class="tab-pane fade show active" id="search_results">
      <div class="row">
        <div class="col-1"></div>
        <div class="col-5">
          <p class="lead">Menampilkan hasil untuk <a href="/search?q={{ query }}"
            style="color: blue !important"><em><u>{{ query }}</u></em></a></p>
          <div id="results">
            {% for i in range(results|length) %}
              <p>
                <div class="result">
                  <div class="font-weight-bold">
                    <a href="/uploads/{{ results[i][0] }}"
                      style="color: blue !important"><u>{{ results[i][1][0] }}</u></a>&nbsp;<small
                        class="text-danger">(D{{ i + 1 }})</small>
                  </div>
                  <div class="text-success">
                    Jumlah kata: {{ results[i][1][2] }}
                  </div>
                  <div class="text-success">
                    Tingkat Kemiripan: {{ results[i][1][1] }}%
                  </div>
                  <div class="result_firstsentence text-muted">
                    <small>{{ results[i][1][3] }}</small>
                  </div>
                </div>
              </p>
            {% endfor %}
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
{% endblock %}
```

```
<div class="tab-pane fade" id="search_table">
  <h3>Tabel kata dan kemunculan:</h3>
  <div class="table-responsive">
    <table class="table table-bordered table-hover">
      <thead class="bg-info">
        <tr>
          <th scope="col">Term</th>
          <th scope="col">Query</th>
          {% for i in range(results|length) %}
            <th scope="col">(D{{ i + 1 }})</th>
          {% endfor %}
        </tr>
      </thead>
      <tbody>
        {% for i in range(order|length) %}
          <tr>
            <th scope="row">{{ order[i][0] }}</th>
            <td>{{ VECTORS["query"][order[i][0]] }}</td>
            {% for j in range(results|length) %}
              <td>{{ VECTORS[results[j][0]][order[i][0]] }}</td>
            {% endfor %}
          </tr>
        {% endfor %}
      </tbody>
    </table>
  </div>
</div>
</div>
{% endblock %}
```

## 23. about.html

```
{% extends 'header.html' %}

{% block header_head %}
    <title>Perihal - Entah</title>
{% endblock %}

{% block header_body %}
    <div class="container">
        <div class="row">
            <div class="col">
                <h1>About Us</h1>
                <p>
                    Kami merupakan kelompok yang beranggotakan tiga orang yaitu Jose, Feral, dan Bryan.
                    Kelompok kami bernama kelompok Entah.
                    Penamaan "Entah" sebagai nama dari kelompok kami tidak lain dan tidak bukan karena ketidakmampuan kami
                    dalam menciptakan nama kelompok.
                    Kelompok ini dibentuk untuk memenuhi tugas besar Aljabar Geometri IF 2123.
                </p>
            </div>
        </div>
    </div>

    <hr>
    <div class="row">
        <div class="col">
            <h1>How To Use</h1>
            <p>
                1. Pengguna bisa memilih untuk <i>upload</i> dokumen txt atau via url yg nanti akan dikonversikan ke dokumen txt.
            </p>
            <p>
                2. Pengguna bisa memasukkan query untuk mencari <i>term</i>-nya.
            </p>
            <p>
                3. Setelah <i>term</i>-nya dicari, akan ditampilkan hasilnya.
                Hasilnya merupakan <i>link</i> yang akan men-<i>direct</i> ke dokumen-dokumen yg telah diupload.
                Dokumen diurutkan dari yang paling sesuai dengan term pencarian.
                Akan muncul juga tabel yang menunjukkan jumlah kemunculan <i>term-term</i> yang dicari pada dokumen-dokumen tersebut.
            </p>
        </div>
    </div>
    <hr>

    <div class="row">
        <div class="col">
            <h1>Konsep Search Engine</h1>
            <p>
                Information retrieval atau Sistem temu balik adalah menemukan (biasanya dokumen) dari sebuah
                ketidakstrukturan yang alami (biasanya teks) untuk memenuhi sebuah kebutuhan informasi dari
                koleksi yang berukuran besar (biasanya disimpan pada komputer).
                Sistem information retrieval atau sistem temu balik informasi bertujuan untuk
                mencukupi kebutuhan informasi pengguna dengan sumber informasi yang tersedia sesuai dengan situasi.
                Penulis mempresentasikan ide dan pikiran mereka ke dalam sebuah dokumen, pencari dokumen mencari
                sebuah dokumen di dalam sekumpulan dokumen dimana pencari tersebut tidak mengetahui dengan pasti
                bagaimana cara menemukan dan mengenali dokumen yang tepat sesuai dengan kebutuhannya,
                sistem temu balik informasi mempertemukan ide yang ditulis penulis tersebut dengan kebutuhan informasi
                yang dibutuhkan oleh pencari dokumen tersebut yang dinyatakan kedalam pernyataan (query).
            </p>
            <p>
                Kemiripan kalimat memainkan peran penting pada berbagai penelitian yang berhubungan dengan teks dan aplikasi.
                Vector Space Model (VSM) digunakan sebagai representasi dari kumpulan dataset dokumen teks.
                Dokumen dalam Vector Space Model (VSM) berupa matriks yang berisi bobot seluruh kata pada tiap dokumen.
                Bobot tersebut menyatakan kepentingan atau kontribusi kata terhadap suatu dokumen dan kumpulan dokumen.
                Kepentingan suatu kata dalam dokumen dapat dilihat dari frekuensi kemunculannya terhadap dokumen.
            </p>
            
            <p>
                Gambar diatas menunjukkan pemodelan dokumen teks di ruang dimensi dimana (D) adalah kalimat dokumen sedangkan (T)
                adalah term atau kata. Untuk mendapatkan nilai jarak atau kemiripan dokumen, dapat menggunakan berbagai macam
                varian rumus perhitungan jarak diantaranya adalah (1) Cosine, (2) Jaccard, (3) Dice, (3) Euclidean, (4) Manhattan,
                (5) Minkowski, (6) Mahalanobis, (8) Weighted.
                Dalam tugas ini kami akan menggunakan metode Cosine.
            </p>
        </div>
    </div>
```

```

<p>
    Cosine Similarity dapat diimplementasikan untuk menghitung nilai kemiripan antar kalimat dan menjadi salah satu
    teknik untuk mengukur kemiripan teks yang populer.
    Contoh penggunaan Cosine Similarity dalam menguji kemiripan dua buah kalimat adalah sebagai berikut:
</p>
<p>
    Misalkan diberikan dua buah kalimat yaitu kalimat A dan B, yaitu:
</p>
<p>
    A : Julie loves me more than Linda loves me
</p>
<p>
    B : Jane likes me more than Julie loves me
</p>
<p>
    Uji kemiripan teks dapat dilakukan dengan rumus:
</p>

<p>
    Dengan:  $j = |A \cap B|$     Kemiripan =
</p>
<p>
    nA = jumlah kemunculan kata indeks ke-n dari daftar kata pada kalimat A.
</p>
<p>
    nB = jumlah kemunculan kata indeks ke-n dari daftar kata pada kalimat B.
</p>

<p>
    Berdasarkan rumus tersebut di atas dilakukan penghitungan seperti di bawah ini. Dengan tingkat kemiripan teks =
</p>

<p>
    = 0.821584
</p>
</div>
</div>
</div>
{% endblock %}

```

## Bab 4

### Eksperimen

#### 1. Search query

The screenshot shows the Entah search engine interface. The search bar contains the word "ekonomi" and the "Search" button is highlighted. Below the search bar, there are tabs for "Result" and "Table". The search results are displayed in a list format, showing the top three results for the query "ekonomi".

Menampilkan hasil untuk [ekonomi](#)

**Jokowi Ungkapkan 3 Poin Peningkatan Teknologi Digital ASEAN.** (D1)  
Jumlah kata: 434  
Tingkat Kemiripan: 17.348855898381338%

Jokowi Ungkapkan 3 Poin Peningkatan Teknologi Digital ASEAN Marcheilla Ariesta • 13 November 2020 22:57 Jakarta: Presiden Republik Indonesia menjadi salah satu pembicara kunci dalam Pertemuan ASEAN Business and Investment Summit 2020 (ABIS 2020) bertema "Digital ASEAN: Sustainable and Inclusive" yang dilaksanakan di Hanoi, Vietnam, Jumat, 13 November 2020.

**Hotel dan Restoran di Kota Bogor Terima Hibah Pariwisata.** (D2)  
Jumlah kata: 331  
Tingkat Kemiripan: 11.571137082807434%

Hotel dan Restoran di Kota Bogor Terima Hibah Pariwisata Rizky Dewantara • 16 November 2020 08:18 Bogor: Tim Pelaksana Pemulihan Ekonomi Akibat Pandemi Covid-19 Kota Bogor, Jawa Barat, mematangkan petunjuk teknis (juknis) pelaksanaan bantuan hibah pariwisata 2020 dari Kementerian Pariwisata dan Ekonomi Kreatif.

**Aturan Turunan UU Cipta Kerja Ditargetkan Selesai Pekan Ini.** (D3)  
Jumlah kata: 345

**Tabel kata dan kemunculan:**

Term	Query	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
ekonomi	1	7	4	3	1	1	0	0	0	0	0	0	0	0	0	0

© 2020 - [About Us](#)

## 2. Upload file (.txt)

The screenshot shows a web browser window with the URL `127.0.0.1:5000/index`. The page has a header with the text "by Jose Feral Bryan". Below the header, there are two tabs: "Search" and "Upload". The "Upload" tab is active. Under the "Upload" tab, there is a section titled "Ada 16 file pada database:" followed by a list of 16 file names. Below this list, there is a section titled "Upload file .txt" with a "Choose Files" button and a text input field containing "test\_upload.txt". Below the text input field, there is a blue "Upload" button. At the bottom of the page, there is a section titled "BETA! Web Scraping" with a text input field containing "https://example.com atau http://example.com".

by Jose Feral Bryan

Search Upload

Ada 16 file pada database:

Adu\_Gahar\_Ponsel\_Gaming\_Rp2\_Jutaan.txt, Apa Itu Omnibus Law Cipta Kerja Isi dan Dampaknya bagi Buruh.txt, Aturan Turunan UU Cipta Kerja Ditargetkan Selesai Pekan Ini.txt, Bandara Ethiopia Jadi Sasaran Serangan Roket.txt, Bintang Mahaputera Hakim MK Kekhawatiran akan Independensi dalam Pengujian UU Kontroversial.txt, BLT Subsidi Gaji untuk 271 Juta Pekerja Cair Hari Ini.txt, Bocoran Samsung Galaxy S21 hingga Fold Tahun Depan.txt, Efek Dipukul Tyson Mata Berair Hidung Berdarah.txt, Harga PS5 Diprediksi Turun Enam Bulan Lagi.txt, Hotel dan Restoran di Kota Bogor Terima Hibah Pariwisata.txt, Jokowi Ucapkan Selamat ke Jacinda Ardern dan Dorong Penguatan Kemitraan ASEAN-Selandia Baru.txt, Jokowi Ungkapkan 3 Poin Peningkatan Teknologi Digital ASEAN.txt, Pertalite Seharga Premium Rp6.450 di Jakarta Berlaku 2 Bulan.txt, Sarah Jessica Parker Jual Rumah Townhouse Rp224 Miliar.txt, UPDATE Bertambah 4.106 Kasus Covid-19 Indonesia Mencapai 467.113.txt, Update Corona Dunia 16 November 547 Juta Orang Terinfeksi AS Catatkan 11 Juta Kasus.txt

Upload file .txt

Choose Files test\_upload.txt

Upload

BETA! Web Scraping (saat ini hanya bisa untuk kompas.com, medcom.id, dan cnnindonesia.com)

https://example.com atau http://example.com

The screenshot shows the same web browser window as the previous one, but the file list under "Ada 17 file pada database:" now includes "test\_upload.txt". The "Choose Files" button is disabled and shows "No file chosen". The "Upload" button is still present. The "BETA! Web Scraping" section is also visible at the bottom.

by Jose Feral Bryan

Search Upload

Ada 17 file pada database:

Adu\_Gahar\_Ponsel\_Gaming\_Rp2\_Jutaan.txt, Apa Itu Omnibus Law Cipta Kerja Isi dan Dampaknya bagi Buruh.txt, Aturan Turunan UU Cipta Kerja Ditargetkan Selesai Pekan Ini.txt, Bandara Ethiopia Jadi Sasaran Serangan Roket.txt, Bintang Mahaputera Hakim MK Kekhawatiran akan Independensi dalam Pengujian UU Kontroversial.txt, BLT Subsidi Gaji untuk 271 Juta Pekerja Cair Hari Ini.txt, Bocoran Samsung Galaxy S21 hingga Fold Tahun Depan.txt, Efek Dipukul Tyson Mata Berair Hidung Berdarah.txt, Harga PS5 Diprediksi Turun Enam Bulan Lagi.txt, Hotel dan Restoran di Kota Bogor Terima Hibah Pariwisata.txt, Jokowi Ucapkan Selamat ke Jacinda Ardern dan Dorong Penguatan Kemitraan ASEAN-Selandia Baru.txt, Jokowi Ungkapkan 3 Poin Peningkatan Teknologi Digital ASEAN.txt, Pertalite Seharga Premium Rp6.450 di Jakarta Berlaku 2 Bulan.txt, Sarah Jessica Parker Jual Rumah Townhouse Rp224 Miliar.txt, test\_upload.txt, UPDATE Bertambah 4.106 Kasus Covid-19 Indonesia Mencapai 467.113.txt, Update Corona Dunia 16 November 547 Juta Orang Terinfeksi AS Catatkan 11 Juta Kasus.txt

Upload file .txt

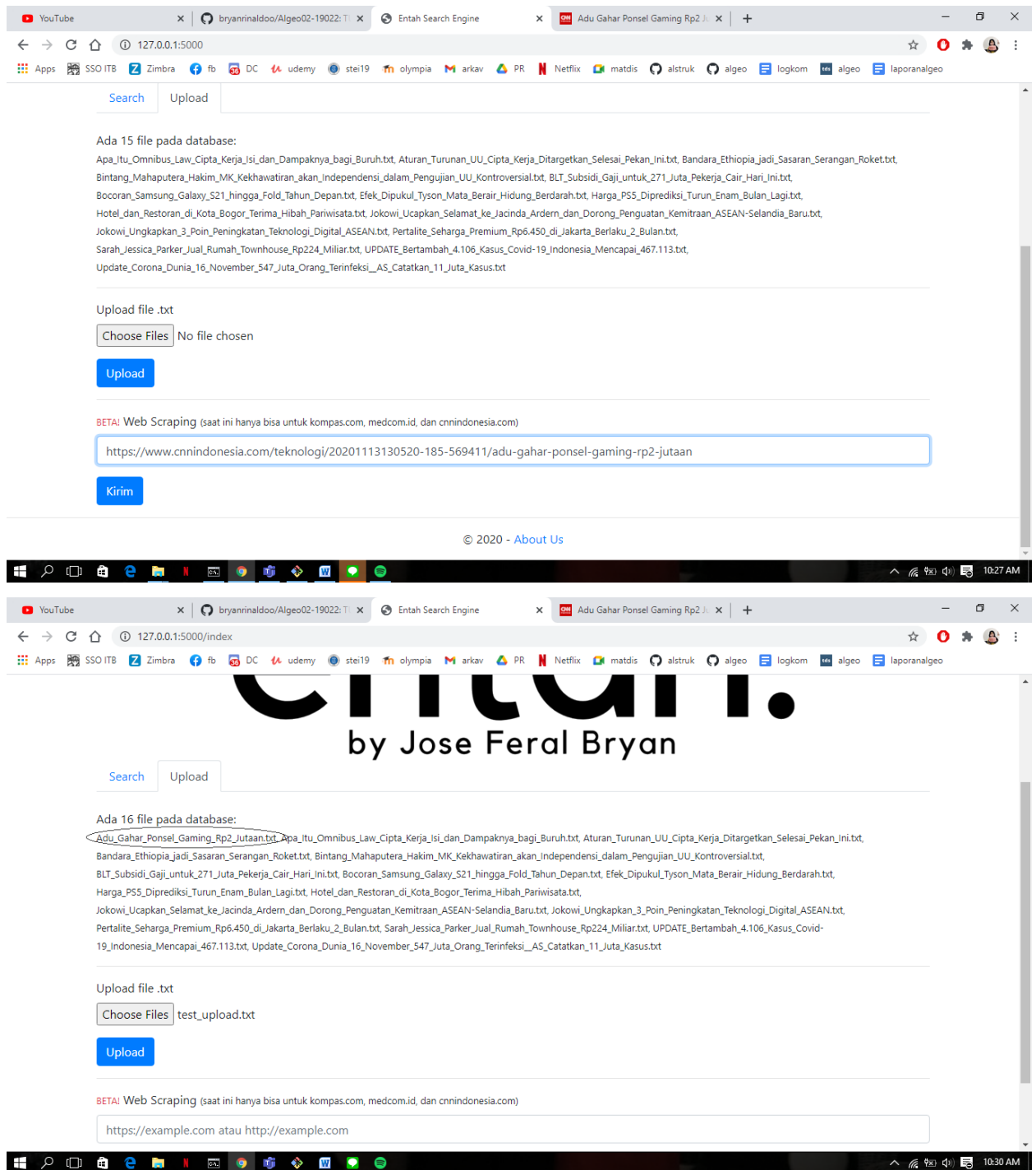
Choose Files No file chosen

Upload

BETA! Web Scraping (saat ini hanya bisa untuk kompas.com, medcom.id, dan cnnindonesia.com)

https://example.com atau http://example.com

### 3. Webscraping



The image displays two screenshots of a web browser window showing a web scraping tool interface. The browser has multiple tabs open, including YouTube, bryannaldao/Algeo02-19022:T, Entah Search Engine, and Adu Gahar Ponsel Gaming Rp2. The address bar shows the URL 127.0.0.1:5000.

The top screenshot shows the tool's main interface. It features a search bar and an upload button. Below the search bar, there is a list of files found in the database, including "Adu\_Gahar\_Ponsel\_Gaming\_Rp2\_Jutaan.txt". The interface also includes an "Upload file .txt" section with a "Choose Files" button and an "Upload" button. At the bottom, there is a "BETA! Web Scraping" section with a text input field containing the URL "https://www.cnnindonesia.com/teknologi/20201113130520-185-569411/adu-gahar-ponsel-gaming-rp2-jutaan" and a "Kirim" button.

The bottom screenshot shows the tool's output. It displays the scraped data from the URL, including the title "Adu Gahar Ponsel Gaming Rp2 Jutaan" and a list of files found in the database. The interface also includes an "Upload file .txt" section with a "Choose Files" button and an "Upload" button. At the bottom, there is a "BETA! Web Scraping" section with a text input field containing the URL "https://example.com atau http://example.com" and a "Kirim" button.

## **Bab 5**

### **Kesimpulan, saran, dan refleksi**

Vektor merupakan materi yang biasanya muncul pada bidang matematika ataupun fisika. Akan tetapi, dari tugas besar ini kita dapat melihat bahwa vektor juga dapat digunakan untuk memodelkan sebuah sistem pencarian. Dengan menggunakan vektor pada sistem pencarian, proses pencarian dapat lebih mudah. Hal ini membantu kita untuk menemukan dokumen yang relevan sesuai dengan yang pengguna cari. Setelah mempelajari dari berbagai macam sumber, kami berhasil untuk membuat sistem pencarian dengan rumus cosine similarity menggunakan python (flask).

Untuk tugas besar yang diberikan kali ini sudah sangat relevan dengan dunia pekerjaan dibandingkan dengan tugas besar 1 yang lalu. Mungkin untuk tugas-tugas besar selanjutnya bisa diberikan yang lebih relevan dan lebih menarik lagi, agar para mahasiswa lebih semangat dalam membuat tugas besar.

Melalui tugas besar ini kami dapat belajar banyak hal, terutama tentang flask, html, dan webscraping. Dari tugas besar ini kami juga belajar untuk saling berkomunikasi satu sama lain dan juga belajar bagaimana cara untuk bekerja sama dalam sebuah tim. Kami dituntut untuk membagi tugas dan saling membantu satu sama lain. Ketika ada yang tidak bisa, yang sudah bisa langsung membantu mengerjakan.



### **Daftar Pustaka**

<http://eprints.umm.ac.id/37604/3/jiptummpp-gdl-fariskadwi-47791-3-bab2.pdf>

<https://informatikalogi.com/vector-space-model-pengukuran-jarak/>

<https://theonotesblog.wordpress.com/2017/05/03/cosine-similarity-indonesia/>