Analisis Kemiripan Dokumen dengan Teori Himpunan Fuzzy Set dalam Sistem Skripsi

Matthew Allen Reynaldo - 13525001
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: matthewar1101@gmail.com , 13525001@std.stei.itb.ac.id

***Abstract*—Seiring bertambahnya volume dokumen digital di perpustakaan ITB, kebutuhan akan sistem pencarian dan pendeteksi kemiripan dokumen menjadi krusial. Pendekatan dengan himpunan klasik pada umumnya terlalu kaku untuk menganalisis kemiripan, karena sistem keanggotaannya biner (hanya anggota atau bukan anggota). Makalah ini menerapkan fuzzy set sebagai solusi dari permasalahan tersebut. Fuzzy set adalah suatu bentuk himpunan yang menilai keanggotaan parsial dari rentang [0, 1]. Makalah ini membahas tentang penerapan fuzzy set untuk menentukan kemiripan dokumen dalam sistem skripsi berdasarkan frekuensi kemunculan kata. Suatu program berbasis python akan menerapkan *pre-processing text* pada dokumen, pemetaan fungsi keanggotaan fuzzy, serta melakukan perhitungan skor kemiripan menggunakan operasi irisan dan gabungan fuzzy. Berdasarkan hasil eksperimen, tahap *text mining* pada implementasi program bersifat case-sensitive sehingga menyebabkan program cenderung lebih sulit untuk mendeteksi kemiripan dokumen. Namun secara garis besar pendekatan himpunan fuzzy dapat digunakan untuk deteksi kemiripan dokumen.**

Keywords—Fuzzy Set, Fuzzy Logic, Teori Himpunan, Digital, Jaccard Similarity.

I. PENDAHULUAN

Pertumbuhan banyak mahasiswa di ITB selalu meningkat dari tahun ke tahun. Pertumbuhan banyaknya makalah yang dibuat juga mengikuti tren yang sama. Oleh karena itu diperlukannya suatu metode untuk memudahkan dosen dan mahasiswa untuk mengecek kemiripan makalah mereka.

Di era digital ini banyak makalah dibuat dalam bentuk digital, sehingga memungkinkan adanya otomatisasi dari pendeteksian kemiripan dokumen digital.

Pendekatan menggunakan himpunan klasik untuk menyimpan kata-kata pada dokumen dalam rangka mencari kemiripan dokumen tidak cocok dengan kebutuhan ini. Himpunan klasik karena keanggotaannya dinilai dengan nilai 0 atau 1 hanya dapat mendeteksi kemiripan mutlak, padahal kemiripan seringkali parsial.

Penggunaan fuzzy set menjadi solusi dari permasalahan ini. Fuzzy set adalah bentuk himpunan yang menilai keanggotaan di rentang [0, 1]. Dengan fuzzy set kemiripan dokumen dapat ditentukan dengan lebih fleksibel. Makalah ini membahas penggunaan operasi fuzzy set untuk menentukan kemiripan dokumen.

II. LANDASAN TEORI

A. Himpunan Klasik dan Himpunan Fuzzy

Himpunan klasik adalah kumpulan objek berbeda. Semua objek yang termasuk ke dalam himpunan dianggap anggota. Berdasarkan definisi dari himpunan itu semua objek dapat dikelompokkan ke dalam dua kelompok yaitu anggota dan bukan anggota, sehingga tidak ada keanggotaan parsial.

Himpunan fuzzy berbeda dengan himpunan klasik dari sistem keanggotaannya. Nilai keanggotaan dari himpunan fuzzy ditentukan oleh fungsi keanggotaan fuzzy, nilai tersebut selalu berada di rentang [0, 1]. Nilai keanggotaan fuzzy dapat menunjukkan keanggotaan parsial, tidak seperti nilai keanggotaan himpunan klasik yang menyatakan keanggotaan mutlak.

B. Fungsi Keanggotaan Fuzzy

Misalkan x adalah suatu kata dari dari himpunan kata pada dokumen. $\mu_A(x)$ adalah fungsi keanggotaan fuzzy untuk dokumen A dan $f_A(x)$ adalah frekuensi kemunculan kata x pada dokumen A. Fungsi keanggotaan fuzzy yang akan digunakan pada makalah ini dapat dituliskan sebagai berikut.

$$\mu_A(x) = \frac{f_A(x)}{max_{y \epsilon A} f_A(y)}$$

Fungsi keanggotaan ini mengaplikasikan normalisasi nilai frekuensi kata pada dokumen sebagai nilai keanggotaan fuzzy.

C. Operasi-operasi Pada Himpunan Fuzzy

Operasi-operasi yang pada himpunan fuzzy sama dengan operasi yang ada pada himpunan klasik. Namun cara kerja operasinya berbeda. Operasi-operasi himpunan fuzzy yang akan dipakai untuk makalah ini adalah sebagai berikut.

- Operasi Union (Gabungan)
  Untuk suatu himpunan fuzzy $\tilde{C}$ adalah union dari himpunan fuzzy $\tilde{A}$ dan $\tilde{B}$ berlaku persamaan berikut.
  $$\tilde{C} = \tilde{A} \, U \, \tilde{B}$$
  $$\mu_{\tilde{C}}(x) = max(\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x))$$
- Operasi Intersect (Irisan)

Untuk suatu himpunan fuzzy $\tilde{D}$ adalah intersection dari himpunan fuzzy $\tilde{A}$ dan $\tilde{B}$ berlaku persamaan berikut.

$$\tilde{D} = \tilde{A} \cap \tilde{B}$$
$$\mu_{\tilde{D}}(x) = min(\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x))$$

D. *Jaccard Similarity*

Jaccard Similarity adalah tolak ukur statistika yang digunakan untuk menentukan kemiripan dua himpunan. Jaccard Similarity untuk himpunan fuzzy dapat dituliskan sebagai berikut.

$$Jaccard\ Similarity = \frac{\sum \mu_{A \cap B}(x)}{\sum \mu_{A \cup B}(x)}$$

E. *Konsep Text Mining Sederhana*

Text mining adalah proses pengekstraksian informasi berguna dari dokumen. Dalam kasus ini *text mining* adalah proses pengekstraksian kata-kata dan frekuensi kata dari dokumen.

Dalam sistem komputer huruf kapital dan non-kapital dianggap berbeda. Namun karena kata-kata yang ingin dibandingkan tidak memedulikan penggunaan huruf kapital maka semua huruf pada dokumen perlu diubah menjadi huruf non-kapital.

Dalam berbagai jenis dokumen, kata-kata umum yang tidak memiliki makna penting seperti: "yang", "di", "ke", "dari", "itu", "dan", "adalah", sangat sering muncul. Karena kata-kata ini tidak seharusnya menunjukkan adanya plagiarisasi pada dokumen maka perlu adanya filter untuk kata-kata ini.

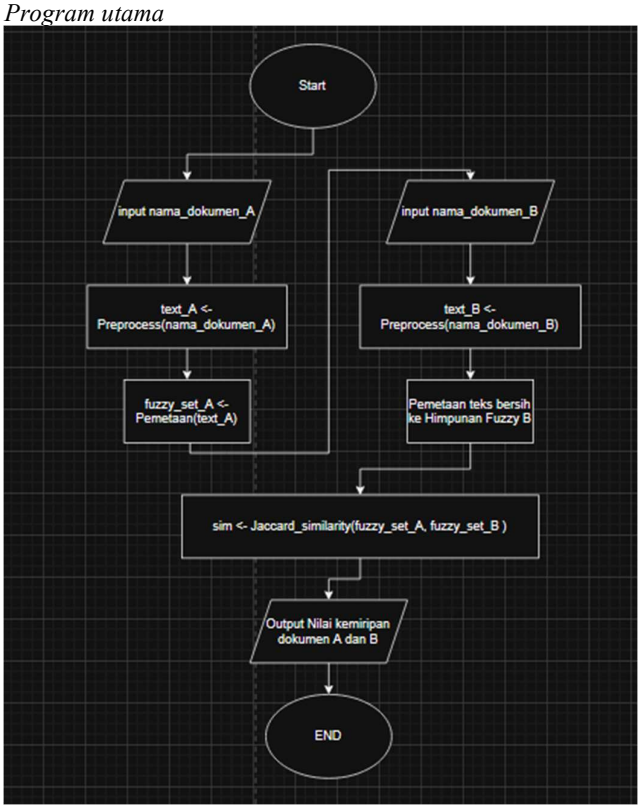
Tanda baca adalah komponen krusial untuk semua kalimat. Tanda baca dan spasi dapat digunakan sebagai penanda untuk pergantian kata yang dibaca program. Jadi untuk setiap tanda baca dan titik yang ditemui maka program dapat menyadari waktunya menyimpan kata dan mulai memindai kata baru.

Setelah ketiga bentuk *pre-processing* di atas dieksekusi, dokumen yang tadinya abstrak diubah menjadi suatu daftar kata dan frekuensi yang mudah dibaca oleh program. Dari daftar bersih ini, kata akan didefinisikan sebagai elemen x terhadap semesta U, kemudian frekuensinya akan dinormalisasi untuk membentuk nilai keanggotaan μ(x) pada himpunan fuzzy dokumen tersebut.

III. Perancangan dan Implementasi Program

A. *Flowchart*

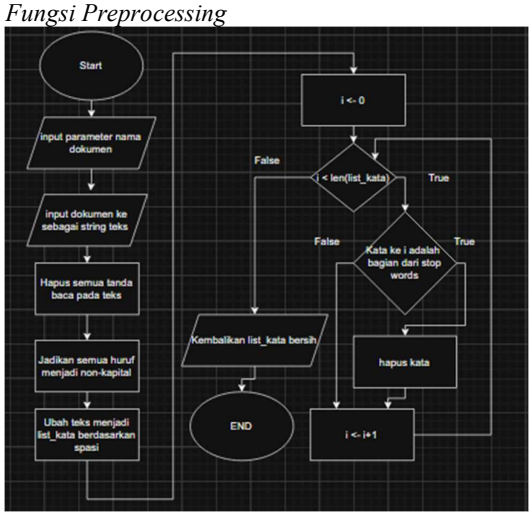
- *Program utama*



Gambar 1. Flowchart Program utama

Program menerima input nama dokumen A di terminal, melakukan preprocessing terhadap dokumen A kemudian dipetakan menjadi himpunan fuzzy. Program melakukan hal yang sama lagi untuk dokumen B. Setelah input dan pemetaan himpunan fuzzy selesai program menghitung nilai kemiripan dokumen (Jaccard Similarity) lalu mengirimkannya ke terminal.

- *Fungsi Preprocessing*



Gambar 2. Flowchart Preprocessing

Program mengambil nama dokumen dari argumen lalu meng-input-nya sebagai string teks. Program kemudian menghapus semua tanda baca pada string teks lalu menjadikan semua huruf menjadi huruf kecil. Program mengubah string teks menjadi list of string list_kata yang dipisah berdasarkan spasi. Terakhir program menghapus semua kata di list_kata yang adalah bagian dari stop_words yang sudah di-*hardcoded* sebelum mengembalikan list_kata.

- *Fungsi Pemetaan*

Gambar 3. Flowchart

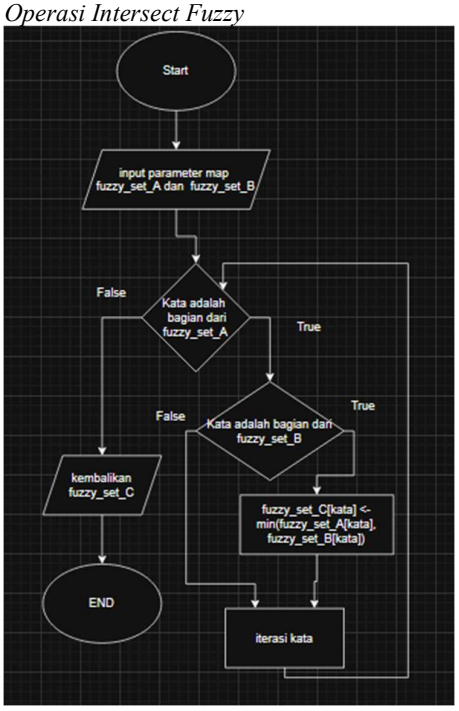
Program mengambil list_kata dari argumen fungsi kemudian menghitung frekuensi dari masing-masing kata di list_kata dan memetakannya ke dalam map frekuensi. Program kemudian membuat suatu set kata_unik untuk menyimpan semua kata unik yang diambil dari list_kata. Program mengiterasi setiap kata pada kata_unik untuk membuat pasangan kata dan nilai keanggotaan fuzzy (menggunakan fungsi Mu) pada map himpunan fuzzy.

- *Mu (Fungsi Keanggotaan Himpunan Fuzzy, μ)*

Gambar 4. Flowchart Fungsi Mu

Program mengambil argumen kata dan map frekuensi lalu mengembalikan frekuensi dari kata dibagi nilai maksimum pada map frekuensi.

- *Operasi Intersect Fuzzy*

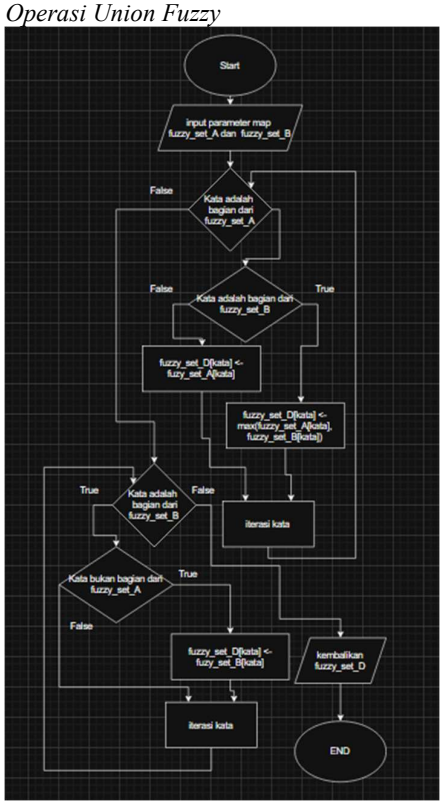


Gambar 5. Flowchart Operasi Intersect

Program mengambil argumen map dari himpunan fuzzy A dan B lalu mengiterasi kata yang ada pada kedua himpunan lalu memetakannya pada himpunan fuzzy C sebagai pasangan dari kata dan minimum dari frekuensi kata di himpunan A dan himpunan B. Dapat diasumsikan jika suatu kata x bagian dari himpunan A tapi bukan bagian dari himpunan B maka nilai keanggotaannya di himpunan

B adalah 0, berlaku juga sebaliknya. Terakhir fungsi mengembalikan himpunan fuzzy C.

- *Operasi Union Fuzzy*



Gambar 6. Flowchart operasi union

Program mengambil argumen himpunan fuzzy A dan himpunan fuzzy B dari fungsi kemudian mengiterasi kata pada himpunan A dan himpunan B, lalu memasukkan kata ke himpunan fuzzy D sebagai pasangan dari kata dan maksimum dari nilai keanggotaan kata di himpunan A dan himpunan B. Dapat diasumsikan jika suatu kata x bagian dari himpunan A tapi bukan bagian dari himpunan B maka nilai keanggotaannya di himpunan B adalah 0, berlaku juga sebaliknya. Terakhir fungsi mengembalikan himpunan fuzzy D.

- *Jaccard_similarity*

Gambar 7. Flowchart Jaccard Similarity

Program mengambil argumen himpunan fuzzy A dan B lalu membuat himpunan fuzzy baru bernama inter dan uni, dengan inter adalah irisan dari A dan B, dan uni adalah gabungan dari A dan B. Jika besar himpunan inter adalah 0 maka Jaccard similarity dari kedua himpunan pasti 0, sehingga program akan langsung mengembalikan nilai 0. Sebaliknya jika besar himpunan inter bukan 0, maka program akan menghitung jumlah dari nilai keanggotaan kata di himpunan inter, kemudian menjumlahkan juga nilai keanggotaan di himpunan uni, lalu mengembalikan hasil dari total nilai keanggotaan himpunan inter dibagi total nilai keanggotaan himpunan uni.

B. Preprocessing

```
def preprocess_text(nama_file):
    if(nama_file == ""):
        return []
    with open(nama_file, 'r',
encoding='utf-8') as file:
        teks = file.read()
    teks_bersih = re.sub(r'[^\w\s]', '',
teks.lower()) # Menghapus tanda baca &
mengubah ke huruf kecil
    kata_kata = teks_bersih.split() #
Tokenisasi
    stopwords = set(["yang", "di", "ke",
"dari", "itu", "dan", "adalah"]) # Daftar
stopwords
```

```
    for i in range(len(kata_kata)):
        if kata_kata[i] in stopwords:
            kata_kata[i] = '' # Menghapus
stopwords
    return kata_kata
```

Potongan kode pada fungsi preprocessing adalah bentuk implementasi dari text mining yang di bahas di bab 2 bagian E. Fungsi ini akan menerima nama_file sebagai argumen pertamanya. Jika fungsi menerima string kosong sebagai argumennya, maka fungsi akan mengembalikan list kosong. Sebaliknya jika string nama_file tidak kosong maka nama_file akan dibuka sebagai file dan dibaca dan dimasukkan ke string teks. teks_bersih dibuat dari teks dengan teks.lower() untuk mengubah semua huruf menjadi huruf kecil, lalu dengan library regex dengan fungsi sub semua tanda baca pada teks.lower() dihapus. Teks_bersih kemudian dipecah menggunakan fungsi split lalu dimasukkan ke list of string kata_kata. Stopwords adalah sebuah set of string yang berisi kata-kata yang ingin diflter. For loop digunakan untuk mengiterasi kata di kata_kata untuk mengosongkan string kata yang termasuk di set stopwords. Fungsi mengembalikan kata_kata.

C. Fungsi Keanggotaan

```
def mu(kata, frekuensi):
    return frekuensi[kata] /
max(frekuensi.values())
```

Didefinisikan fungsi keanggotaan sebagai mu. Fungsi ini adalah implementasi langsung dari fungsi keanggotaan seperti dibahas pada bab 2 di bagian B. Fungsi mu menerima 2 argumen yaitu sebuah string kata dan map frekuensi yang berisi string kata sebagai key dan integer frekuensi kemunculan kata sebagai value. Fungsi ini akan mengembalikan frekuensi dari kata dibagi maximum value pada map frekuensi, nilai yang dikembalikan akan berupa float.

D. Pemetaan Himpunan Fuzzy

```
def pemetaan(kata_kata):
    frekuensi = {}
    for i in range(len(kata_kata)):
        kata = kata_kata[i]
        if kata != '':  # Pastikan kata
tidak kosong
            if kata in frekuensi:
                frekuensi[kata] += 1
            else:
                frekuensi[kata] = 1
    kata_unik = set(kata_kata)  # Membuat
set kata unik
    keanggotaan_dict = {}
    for kata in kata_unik:
        if kata != '':  # Pastikan kata
tidak kosong
            keanggotaan_dict[kata] =
mu(kata, frekuensi)

    return keanggotaan_dict
```

Fungsi pemetaan adalah fungsi yang dibuat sebagai intermediet untuk membentuk struktur himpunan fuzzy. Fungsi pemetaan menerima list of string kata_kata sebagai argumen. Fungsi kemudian membuat suatu map kosong bernama frekuensi, fungsi akan mengiterasi kata pada kata_kata, untuk setiap iterasi fungsi akan mengecek apakah kata sudah disimpan di map? Jika ya maka fungsi akan menambahkan frekuensinya sebanyak satu pada map. Jika sebaliknya maka suatu pasangan baru bernilai (kata, 1) akan dimasukkan ke dalam map frekuensi. Tujuan adanya pengecekan ini supaya pasangan dengan key kata tidak terus menerus ditambahkan tanpa mengubah value pada map. Kata_unik adalah himpunan kata pada kata_kata. Keanggotaan_dict adalah map kosong yang akan digunakan untuk menyimpan himpunan fuzzy. Fungsi akan mengiterasikan kata pada kata_unik untuk memasukkan pasangan (kata, mu(kata, frekuensi)) pada map keanggotaan_dict. Setelah map himpunan fuzzy terbentuk maka fungsi akan mengembalikan map keanggotaan_dict.

E. Operasi Himpunan Fuzzy

Fungsi intersect dan union adalah implementasi langsung dari operasi pada himpunan fuzzy seperti yang sudah dibahas di bab 2 bagian C.

- Operasi Intersect

```
def intersect(setA, setB):
    setD = {}
    for item in setA.keys():
        if item in setB.keys():
            setD[item] = min(setA[item],
setB[item])
    return setD
```

Fungsi intersect menerima map himpunan fuzzy setA dan setB sebagai argumennya. Fungsi kemudian membuat map kosong setD yang akan menjadi wadah himpunan fuzzy hasil intersect. Fungsi kemudian mengiterasikan item yang adalah key pada setA lalu mengecek apakah item juga merupakan bagian dari key pada setB? Jika item juga bagian dari key di setB maka tambahkan item pada map setD dengan value-nya adalah minimum dari value di setA dan setB untuk key item. Pengecekan ini hanya digunakan untuk mempercepat operasi karena jika tidak ada di kedua map maka pasti minimum value-nya 0. Fungsi mengembalikan map hasil setD.

- Operasi Union

```
def union(setA, setB):
    setC = {}
    for item in setA.keys():
        if item in setB.keys():
```

```
            setC[item] = max(setA[item],
setB[item])
        else:
            setC[item] = setA[item]
    for item in setB.keys():
        if item not in setA.keys():
            setC[item] = setB[item]
    return setC
```

Fungsi union menerima map himpunan fuzzy setA dan setB sebagai argumen. Fungsi membuat suatu map kosong setC yang akan menjadi wadah himpunan fuzzy hasil dari operasi union pada setA dan setB. Fungsi melakukan iterasi item dari setA seperti pada fungsi intersect, perbedaannya terletak pada pemasangan nilai pada setC. setC akan ditambahkan pasangan item dan nilai maksimum dari value di setA dan setB untuk key item. Jika item tidak ada di setB maka setC akan mengambil value dari setA karena value di setB sudah pasti 0 sehingga value di setA pasti lebih besar. Hal yang sama juga dilakukan untuk setB, yaitu fungsi akan mengiterasi juga item di key setB untuk menambahkan sisa item yang tidak ada di setA ke setC. Fungsi mengembalikan setC sebagai hasil.

F. Jaccard Similarity

```
def Jaccard_similarity(setA, setB):
    setD = intersect(setA, setB)
    setC = union(setA, setB)
    if len(setC) == 0:
        return 0.0
    sum_D = 0.0;
    sum_C = 0.0;
    for item in setD.keys():
        sum_D += setD[item]
    for item in setC.keys():
        sum_C += setC[item]
    return sum_D / sum_C
```

Fungsi Jaccard_similarity adalah implementasi langsung dari jaccard similarity seperti yang sudah dibahas di bab 2 bagian D. Fungsi ini menerima map himpunan fuzzy setA dan setB sebagai argumen. Fungsi membuat map setD dan setC yang secara berturut-turut adalah intersect dan union dari setA dan setB. Fungsi mengecek dahulu besar setD. Jika besar setD bernilai 0 maka fungsi dapat mengabaikan langkah-langkah berikutnya dan langsung mengembalikan nilai 0. Jika sebaliknya maka fungsi akan lanjut untuk mendeklarasikan variabel float sum_D dan sum_C yang berturut-turut adalah total nilai keanggotaan pada setD dan setC. Fungsi kemudian mengiterasikan item di setD dan menjumlahkan semua value nya pada sum_D. Hal yang sama dilakukan untuk setC. Fungsi mengembalikan nilai sum_D dibagi sum_C.

G. Fungsi Utama Program

```
def main():
    nama_file = input("Masukkan nama file A
teks (misal: dokumen.txt): ")
    kata_kata = preprocess_text(nama_file)
    setA = pemetaan(kata_kata)
    nama_file = input("Masukkan nama file B
teks (misal: dokumen2.txt): ")
    kata_kata = preprocess_text(nama_file)
    setB = pemetaan(kata_kata)

    sim = Jaccard_similarity(setA, setB)
    print(f"Similarity setA dan setB:
{sim*100:.2f}%")
    if sim > 0.8:
        print("Teks memiliki kemiripan yang
tinggi.")
    elif sim > 0.5:
        print("Teks memiliki kemiripan yang
cukup.")
    else:
        print("Teks memiliki kemiripan yang
rendah.")
```

Fungsi main adalah fungsi utama untuk program berjalan sekaligus bagian yang mengatur CLI bagi pengguna. Di fungsi main pengguna akan diminta untuk menempatkan dokumen di folder yang sama dengan program dan memasukkan nama dokumen itu ke terminal, bagian ini dilakukan 2 kali yaitu untuk dokumen A dan dokumen B. Fungsi main akan melakukan preprocessing nama_dokumen sebelum membuat setA dari dokumen A dan dilanjut dengan preprocessing lagi dan pembuatan setB dari dokumen B. Fungsi kemudian menghitung nilai Jaccard_similarity pada kedua himpunan fuzzy lalu menampilkan hasilnya kepada pengguna sekaligus memberi komentar tentang kemiripan dokumen berdasarkan nilai jaccard similarity.

IV. Analisis dan Hasil Eksperimen

A. Skenario dan Data Pengujian

Program yang sudah dibuat ini rencananya akan diuji dengan tiga skenario berbeda. Untuk mempermudah proses pengujian dokumen yang digunakan hanya berbentuk ".txt" *file* dan berisi beberapa potong kalimat. Skenario beserta data pengujian dapat diakses melalui hyperlink pada tabel berikut.

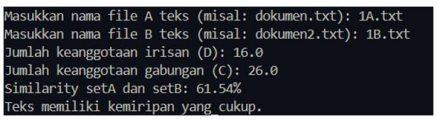
TABLE I. Tabel Skenario dan Ekspektasi

Skenario	Dokumen A	Dokumen B	Ekspektasi Nilai Kemiripan
1	1A	1B	>75%
2	2A	2B	30-50%
3	3A	3B	≥0%

Skenario 1 menggunakan 2 dokumen yang sangat mirip mengenai mata kuliah mat diskrit di Teknik Informatika ITB.
Skenario 2 menggunakan 2 dokumen yang memiliki topik yang mirip tapi isinya berbeda.
Skenario 3 menggunakan 2 dokumen yang sama sekali tidak berhubungan.

B. Proses Eksperimen dan Perhitungan Sistem

Supaya perhitungan dapat terlihat lebih jelas, program akan dimodifikasi untuk menampilkan frekuensi dan himpunan dari dokumen A dan B. Untuk memperlihatkan bagaimana proses pengujian dilaksanakan maka diambilah contoh menggunakan data pengujian skenario 1.



Gambar 8. Contoh keluaran program dengan skenario 1

C. Hasil Pengujian Keseluruhan

Berikut adalah hasil pengujian berdasarkan data percobaan di Tabel I.

TABLE II. TABEL PERCOBAAN

Skenario	Nilai Irisan	Nilai Gabungan	Similarity (%)	Status Kemiripan
1	**16.0**	**26.0**	**61.54**	**Cukup Mirip**
2	**4.0**	**27.0**	**14.81**	**Tidak Mirip**
3	**0.0**	**17.0**	**0**	**Tidak Mirip**

D. Evaluasi Algoritma

Berdasarkan hasil percobaaan di Tabel II jika dibandingkan dengan prediksi di Tabel I, terlihat bahwa hasil program tidak mengikuti prediksi pada skenario 1 dan 2. Kejomplangan nilai irisan dan gabungan adalah alasan utama hasil rumus Jaccard Similarity lebih rendah dari prediksi.

Setelah analisis lebih lanjut, disadari bahwa bagian *pre-processing (text mining)* pada program menganggap kata-kata yang mirip artinya sebagai kata berbeda sama sekali. Sebagai contoh kata "mempelajari" dan "pelajari" memiliki arti serupa tapi dianggap berbeda dengan program. Karena ini jika program ditambahkan fungsi *stemming* pada bagian *pre-processing* program dapat menangkap kata-kata yang bermakna mirip dengan lebih akurat.

V. KESIMPULAN

Himpunan fuzzy adalah salah satu dari berbagai struktur data yang dapat digunakan dengan mudah untuk mencari hubungan antar data. Efektivitas himpunan fuzzy terbukti dapat mengatasi keterbatasan himpunan klasik dalam analisis tekstual melalui pembobotannya yang berbasis frekuensi kata dan tidak biner. Operasi logika fuzzy masih terbukti sukses untuk menentukan nilai kemiripan yang terukur dengan rumus jaccard similarity. Karakteristik rumus jaccard yang sensitif terhadap kemunculan variasi morfologi bahasa dan kemunculan kata unik yang sepihak membuat nilai kemiripan cenderung rendah. Oleh karena itu diperlukan kalibrasi ulang batasan nilai kemiripan untuk menentukan status kemiripan.

VI. LAMPIRAN

Source code program: https://github.com/spazmash/Sistem-dasar-pendeteksi-kemiripan-dokumen

VII. UCAPAN TERIMA KASIH

Terima kasih kepada Tuhan yang Maha Esa karena berkat rahmat dan kasih karunia-Nya penulis dapat menyelesaikan makalah ini tepat waktu. Penulis juga ingin mengucapkan terima kasih kepada keluarga penulis yang selalu mendukung penulis hingga penulis dapat menyelesaikan makalah ini. Terima kasih juga untuk Bapak Rinaldi Munir selaku Dosen mata kuliah IF1230 untuk kelas K-01 karena telah memberi pengetahuan yang mendasari bahan pembuatan makalah ini. Penulis berharap bahwa kelak makalah ini bisa menjadi referensi bagi pelajar yang ingin mendalami ilmunya di bidang ini.

REFERENCES

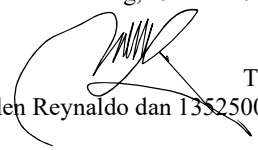
[1] GeeksforGeeks. 2022. *Fuzzy Logic | Set 2 (Classical and Fuzzy Sets).* https://www.geeksforgeeks.org/aptitude/fuzzy-logic-set-2-classical-fuzzy-sets/ . Diakses pada 18 Juni 2026.

[2] GeeksforGeeks. 2026. *Fuzzy Logic | Introduction.* https://www.geeksforgeeks.org/artificial-intelligence/fuzzy-logic-introduction/ . Diakses pada 18 Juni 2026.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Juni 2026



Ttd

Matthew Allen Reynaldo dan 13525001