IF4041 Teknologi Basis Data
Laporan Tugas *Stream Mining* dan *Graph Mining*

Oleh:
13515032 Helena Suzane Graciella
13515091 Adrian Hartarto Pramudita
13515127 Fildah Ananda Amalia
13516132 Daniel Ryan Levyson

TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2018

Stream Mining

1. Penjelasan data

Pada percobaan mining data stream dilakukan beberapa percobaan. Percobaan yang dilakukan menggunakan data *dummy*. Percobaan pertama yaitu mengenai *history* belanja suatu barang. Data *history* belanja ini merupakan *dictionary* dengan key [‘username’, ‘product_id’, ‘product_name’, ‘purchase_at’] (contoh: {‘username’: ‘johndoe’, ‘product_id’: 1, ‘product_name’: ‘shirt’, ‘purchase_at’: 1544925480). Percobaan kedua yaitu mengenai menghitung *keyword*. Data yang dikirim merupakan *dictionary* dengan key [‘keyword’, ‘time’] (contoh: {‘keyword’: ‘Jomblo’, ‘time’: ‘2014-12-17T09:35:37 -07:00’}).

2. Source code

2.1 Percobaan 1

Data Stream pada percobaan ini, dibangkitkan secara acak oleh program dengan himpunan pilihan data sebagai berikut:

- username: {johndoe, john12, doe98, johndoe123, daniel123, ryan123, levyson123, danielryan, ryanlevyson, danielryan123, ryanlev123, ryanjohn123}
- product_id: {1, 2, 3, 4, 5, 6, 7}
- product_name: {shirt, shoes, laptop, book, carpet, mouse, keyboard}

Berikut adalah contoh stream yang dibangkitkan program:

```
{'username': 'johndoe123', 'product_id': 2, 'product_name': 'shoes', 'purchase_at': 1545137591401}
{'username': 'daniel123', 'product_id': 1, 'product_name': 'shirt', 'purchase_at': 1545137591416}
{'username': 'doe98', 'product_id': 1, 'product_name': 'shirt', 'purchase_at': 1545137591433}
{'username': 'ryanlevyson', 'product_id': 2, 'product_name': 'shoes', 'purchase_at': 1545137591445}
{'username': 'johndoe123', 'product_id': 2, 'product_name': 'shoes', 'purchase_at': 1545137591459}
{'username': 'danielryan123', 'product_id': 4, 'product_name': 'book', 'purchase_at': 1545137591471}
{'username': 'ryanlev123', 'product_id': 4, 'product_name': 'book', 'purchase_at': 1545137591482}
```

1. Sampling Data Stream

Data pada stream bisa jadi terlalu besar. Tujuan dari *sampling* adalah mengambil *sample* yang merepresentasikan keseluruhan data yang ada pada stream. Pada percobaan ini, tiap data akan dikelompokan ke dalam 10 *bucket* dengan cara melakukan hash pada *username*. *Sample* yang akan diambil adalah 2/10 dari kesuluruhan data. Berikut adalah kode yang digunakan:

```
class GeneralSampler:
    def __init__(self, threshold, bucketsize, datakeys, datafilters):
        self.threshold = threshold
        self.bucketsize = bucketsize
        self.datakeys = datakeys
        if datafilters == None:
            self.datafilters = None
        else:
            self.datafilters = [PurchaseFilter(datafilter[0], datafilter[1]) for datafilter in datafilters]

    def hash(self, data):
        s = 0
        for datakey in self.datakeys:
            if type(data[datakey]) is str:
                s += sum(ord(c) for c in data[datakey])
            else:
                s += int(data[datakey])
        return s % self.bucketsize

    def handler(self, data):
        datahash = self.hash(data)
        if datahash < self.threshold:
            if self.datafilters != None:
                for datafilter in self.datafilters:
                    if not datafilter.filter(data):
                        return
            print('received', data)

if __name__ == '__main__':
    gs = GeneralSampler(threshold=2, bucketsize=10, datakeys=['username'], datafilters=None)
    sampling_stream = Stream(throughput=100, onreceive=gs.handler)
    sampling_stream.open()
```

Hasilnya, *sample* yang didapat adalah tupel dengan username *john12* dan *daniel123*. Berikut hasil yang didapat pada stream:

```
{'username': 'john12', 'product_id': 7, 'product_name': 'keyboard', 'purchase_at': 1545139182654}
{'username': 'daniel123', 'product_id': 1, 'product_name': 'shirt', 'purchase_at': 1545139182854}
{'username': 'daniel123', 'product_id': 7, 'product_name': 'keyboard', 'purchase_at': 1545139182875}
{'username': 'john12', 'product_id': 7, 'product_name': 'keyboard', 'purchase_at': 1545139183063}
{'username': 'daniel123', 'product_id': 6, 'product_name': 'mouse', 'purchase_at': 1545139183220}
{'username': 'daniel123', 'product_id': 1, 'product_name': 'shirt', 'purchase_at': 1545139183252}
{'username': 'daniel123', 'product_id': 6, 'product_name': 'mouse', 'purchase_at': 1545139183284}
.....
```

2. Filtering Data Stream

Selain melakukan sampling, jumlah data pada stream dapat diperkecil dengan membuat filter. Tujuan dari filtering adalah untuk menyaring data stream sehingga data yang masuk ke stream hanya data dengan kondisi yang diinginkan. Pada percobaan ini akan digunakan filter untuk mengambil tupel yang memiliki nilai *danielryan* pada *username*. Berikut adalah kode yang digunakan:

```
class PurchaseFilter:
    def __init__(self, key, datafilter):
        self.bucketsize = 1000000
        hashbucket = [False for i in range(self.bucketsize)]
        for data in datafilter:
            datahash = sum(ord(c) for c in data) %  self.bucketsize
            hashbucket[datahash] = True
        self.key = key
        self.datafilter = datafilter
        self.hashbucket = hashbucket

    def filter(self, data):
        datahash = sum(ord(c) for c in data[self.key]) %  self.bucketsize
        return self.hashbucket[datahash]

if __name__ == '__main__':
    gs = GeneralSampler(threshold=10, bucketsize=10, datakeys=['username'], datafilters=[('username', ['danielryan'])])
    sampling_stream = Stream(throughput=100, onreceive=gs.handler)
    sampling_stream.open()
```

Hasilnya, data yang masuk pada stream hanya tupel yang memiliki username *danielryan*. Berikut ini hasil yang didapat pada stream:

```
{'username': 'danielryan', 'product_id': 3, 'product_name': 'laptop', 'purchase_at': 1545140092225}
{'username': 'danielryan', 'product_id': 3, 'product_name': 'laptop', 'purchase_at': 1545140092268}
{'username': 'danielryan', 'product_id': 5, 'product_name': 'carpet', 'purchase_at': 1545140092290}
{'username': 'danielryan', 'product_id': 4, 'product_name': 'book', 'purchase_at': 1545140092783}
{'username': 'danielryan', 'product_id': 2, 'product_name': 'shoes', 'purchase_at': 1545140092804}
{'username': 'danielryan', 'product_id': 2, 'product_name': 'shoes', 'purchase_at': 1545140092890}
{'username': 'danielryan', 'product_id': 7, 'product_name': 'keyboard', 'purchase_at': 1545140093071}
.....
```

3. Counting Distinct Elements

Adakalanya, sebuah toko online ingin mengetahui berapa banyak pengguna yang berbeda yang melakukan pembelian pada toko. Untuk melakukan hal itu, kita dapat mengestimasi jumlah elemen yang berbeda pada stream berdasarkan *key* tertentu dengan menggunakan algoritma Flajolet-Martin. Untuk mengetahui jumlah pengguna berbeda yang melakukan pembelian digunakan *username* sebagai *key* untuk mengidentifikasi data. Berikut adalah kode yang digunakan:

```
class FlajoletMartin:
    def __init__(self, keys):
        self.keys = keys
        self.bitmax = 1 << 64
        self.currentmax = 0

    def hash(self, data):
        s = 0
        for key in self.keys:
            s += sum(ord(c) for c in data[key]) % self.bitmax
        return s

    def get_trailing_zeros(self, datahash):
        if datahash == 0:
            return 1 << 64
        else:
            c = 0
            while (datahash % 2) != 1:
                datahash = datahash >> 1
                c += 1
            return c

    def new(self, data):
        datahash = self.hash(data)
        lsb = self.get_trailing_zeros(datahash)
        self.currentmax = max(self.currentmax, lsb)
        print(data, self.count())

    def count(self):
        if self.currentmax == 0:
            return 0
        else:
            return 1 << self.currentmax

if __name__ == '__main__':
    fm = FlajoletMartin(['username'])
    distinct_stream = Stream(throughput=20, onreceive=fm.new)
    distinct_stream.open()
```

Hasil yang akan ditunjukan berikut ini adalah data yang masuk ke stream dan estimasi jumlah pengguna yang berbeda. Di awal stream, estimasi jumlah pengguna adalah 2. Seiring bertambahnya data pada stream, estimasi jumlah pengguna adalah 16. Jumlah pengguna yang berbeda sebenarnya adalah 12. Berikut ini hasil yang didapat pada stream:

```
{'username': 'john12', 'product_id': 2, 'product_name': 'shoes', 'purchase_at': 1545140531943} 2
{'username': 'ryanjohn123', 'product_id': 3, 'product_name': 'laptop', 'purchase_at': 1545140531994} 2
{'username': 'doe98', 'product_id': 1, 'product_name': 'shirt', 'purchase_at': 1545140532045} 2
{'username': 'ryanjohn123', 'product_id': 4, 'product_name': 'book', 'purchase_at': 1545140532096} 2
.....
{'username': 'levyson123', 'product_id': 5, 'product_name': 'carpet', 'purchase_at': 1545140532299} 16
{'username': 'doe98', 'product_id': 2, 'product_name': 'shoes', 'purchase_at': 1545140532350} 16
{'username': 'daniel123', 'product_id': 6, 'product_name': 'mouse', 'purchase_at': 1545140532400} 16
{'username': 'ryanlevyson', 'product_id': 2, 'product_name': 'shoes', 'purchase_at': 1545140532451} 16
.....
```

4. Counting Itemsets

Selanjutnya, dari data pembelian ini, ingin diketahui berapa banyak pembelian pada barang dengan *product_id* 5, yaitu *carpet*, pada 50 pembelian terakhir. Hal ini dapat dilakukan dengan menggunakan algoritma Datar-Gionis-Indynk-Motwani (DGIM). Berikut adalah kode yang digunakan:

```
class DGIM:
    def __init__(self, size, key, value):
        self.key = key
        self.value = value
        self.size = size
        self.buckets = []
        self.timestamp = 0

    def new(self, data):
        if len(self.buckets) > 0 and self.buckets[0]['timestamp'] == self.timestamp:
            del self.buckets[0]

        if data[self.key] == self.value:
            c = True
            i = len(self.buckets) - 1
            while c and i > 0:
                if self.buckets[i]['size'] == self.buckets[i-1]['size']:
                    self.buckets[i-1]['size'] += self.buckets[i]['size']
                    self.buckets[i-1]['timestamp'] = self.buckets[i]['timestamp']
                    del self.buckets[i]
                    i -= 1
                else:
                    c = False
            self.buckets.append({'size': 1, 'timestamp': self.timestamp})

        self.timestamp = (self.timestamp + 1) % self.size

        print(data, self.count())

    def count(self):
        if len(self.buckets) > 0:
            s = 0
            for bucket in self.buckets:
                s += bucket['size']
            return s - math.ceil(self.buckets[0]['size']/2)
        else:
            return 0

if __name__ == '__main__':
    sw = DGIM(size=50, 'product_id', 5)
    counting_stream = Stream(throughput=50, onreceive=sw.new)
    counting_stream.open()
```

Hasil yang akan ditunjukan berikut ini adalah data yang masuk ke stream dan jumlah pembelian *carpet* pada 50 pembelian terakhir. Berikut ini hasil yang didapat pada stream:

```
{'username': 'ryanlev123', 'product_id': 5, 'product_name': 'carpet', 'purchase_at': 1545141927325} 1
{'username': 'john12', 'product_id': 1, 'product_name': 'shirt', 'purchase_at': 1545141927346} 1
{'username': 'danielryan', 'product_id': 4, 'product_name': 'book', 'purchase_at': 1545141927366} 1
{'username': 'doe98', 'product_id': 6, 'product_name': 'mouse', 'purchase_at': 1545141927386} 1
{'username': 'john12', 'product_id': 5, 'product_name': 'carpet', 'purchase_at': 1545141927407} 2
{'username': 'danielryan123', 'product_id': 1, 'product_name': 'shirt', 'purchase_at': 1545141927429} 2
{'username': 'ryan123', 'product_id': 1, 'product_name': 'shirt', 'purchase_at': 1545141927450} 2
.....
{'username': 'ryanlev123', 'product_id': 5, 'product_name': 'carpet', 'purchase_at': 1545141929043} 6
{'username': 'johndoe', 'product_id': 3, 'product_name': 'laptop', 'purchase_at': 1545141929064} 6
{'username': 'johndoe', 'product_id': 5, 'product_name': 'carpet', 'purchase_at': 1545141929084} 5
{'username': 'ryan123', 'product_id': 5, 'product_name': 'carpet', 'purchase_at': 1545141929104} 6
{'username': 'doe98', 'product_id': 4, 'product_name': 'book', 'purchase_at': 1545141929125} 6
{'username': 'ryanlev123', 'product_id': 6, 'product_name': 'mouse', 'purchase_at': 1545141929146} 6
{'username': 'john12', 'product_id': 5, 'product_name': 'carpet', 'purchase_at': 1545141929167} 7
.....
```

2.2 Percobaan 2

2.2.1 Tujuan

Percobaan ini dilakukan dengan tujuan untuk mensimulasikan pencarian *trend* pada suatu data stream. Data *dummy* di-*generate* secara random lalu diolah untuk mengetahui *keyword* pencarian yang sedang *trend*. Hasil dari percobaan ini bisa diterapkan untuk kepentingan seperti pemilihan topik berita untuk diikuti agar tidak ketinggalan informasi.

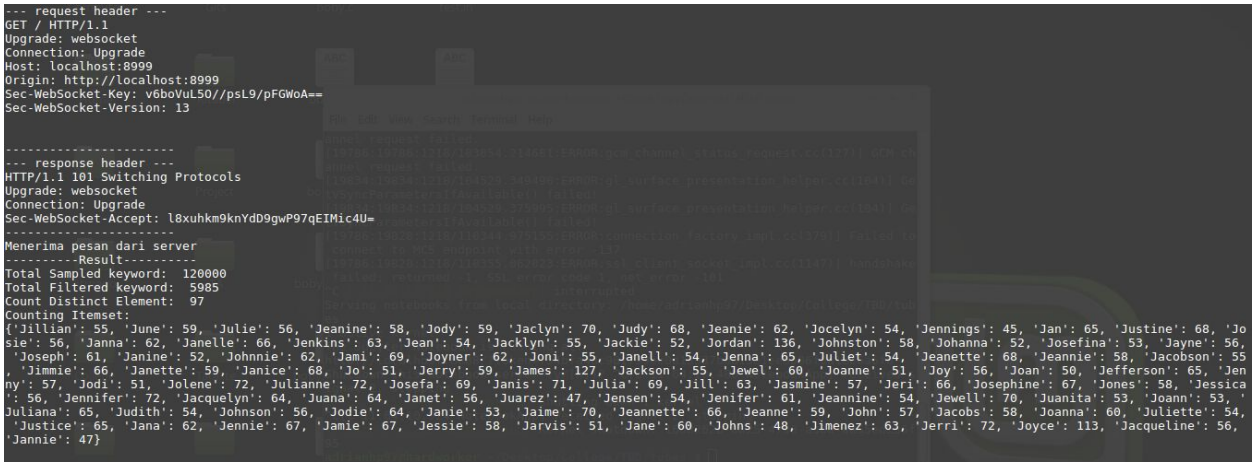
2.2.2 Algoritma

```
import websocket
import json
try:
    import thread
except ImportError:
    import _thread as thread
import time
import random

def process_data():
    def sampling_data(data):
        return random.sample(data, (int)(len(dataset)/10))

    def filter_data(data, keyword):
        return [val for val in data if keyword in val]

    def counting_distinct_element(data):
        return set(data)

    def counting_itemset(data):
        itemset = {}
        for element in data:
            if element in itemset:
                itemset[element] += 1
            else:
                itemset[element] = 1
        return itemset

    with open("keyword.txt", "r") as read_file:
        dataset = read_file.read()
    dataset = dataset.split("\n")
    sampled = sampling_data(dataset)
    print("----------Result----------")
    print("Total Sampled keyword: ", len(sampled))
    filtered = filter_data(sampled, 'J')
    print("Total Filtered keyword: ", len(filtered))
    distinct_element = counting_distinct_element(filtered)
    print("Count Distinct Element: ", len(distinct_element))
    itemset = counting_itemset(filtered)
    print("Counting Itemset: ")
    print(itemset)
    print("\n\n")

def on_message(ws, message):
    def convert_message_to_dict(message):
        return json.loads(message)

    def save_to_file(data):
        with open("keyword.txt", "r") as read_file:
            dataset = read_file.read()
```

```
        dataset = dataset.split("\n")
        with open("keyword.txt", "w+") as write_file:
            all_keyword = [item['keyword'] for item in data]
            dataset += all_keyword
            write_file.write("\n".join(dataset))

    print("Menerima pesan dari server")
    dataset = convert_message_to_dict(message)
    save_to_file(dataset)
    process_data()

def on_error(ws, error):
    print(error)

def on_close(ws):
    print("### closed ###")

def on_open(ws):

    def run(*args):
        test = 0
    thread.start_new_thread(run, ())

if __name__ == "__main__":
    websocket.enableTrace(True)
    ws = websocket.WebSocketApp("ws://localhost:8999",
                                on_message = on_message,
                                on_error = on_error,
                                on_close = on_close)
    ws.on_open = on_open
    ws.run_forever()
```

2.2.3 Hasil Eksperimen

Berikut contoh hasim eksperimen yang didapatkan



Graph Mining

1. **Dataset Yang Digunakan**

   Dataset yang digunakan oleh kelompok 13 untuk eksplorasi *graph mining* adalah *General Relativity and Quantum Cosmology collaboration network* (https://snap.stanford.edu/data/ca-GrQc.html). Terdapat 1 *file* yang digunakan, yakni CA-GrQc.txt. *File* ini berisi daftar sisi tak berarah yang direpresentasikan dengan FromNodeId \t ToNodeId. Daftar simpul didapatkan dengan menggabungkan kedua kolom menjadi 1 kolom, kemudian mengambil nilai-nilai unik dari kolom tersebut.

   Sebuah simpul merepresentasikan seorang peneliti, sedangkan sisi antara simpul a dan b berarti a pernah melakukan penelitian bersama b.

   Pada dataset ini, keterangan mengenai peneliti tidak tersedia.

   Berikut ini data statistik singkat dari dataset berdasarkan *web* penyedia data tersebut. Namun, *file* yang dapat diunduh pada laman tersebut memberikan daftar sisi berjumlah 28980 sisi.

Dataset statistics	
Nodes	5242
Edges	14496
Nodes in largest WCC	4158 (0.793)
Edges in largest WCC	13428 (0.926)
Nodes in largest SCC	4158 (0.793)
Edges in largest SCC	13428 (0.926)
Average clustering coefficient	0.5296
Number of triangles	48260
Fraction of closed triangles	0.3619
Diameter (longest shortest path)	17
90-percentile effective diameter	7.6

Gambar 1 Data statistik dataset

2. **Tujuan Analisis**

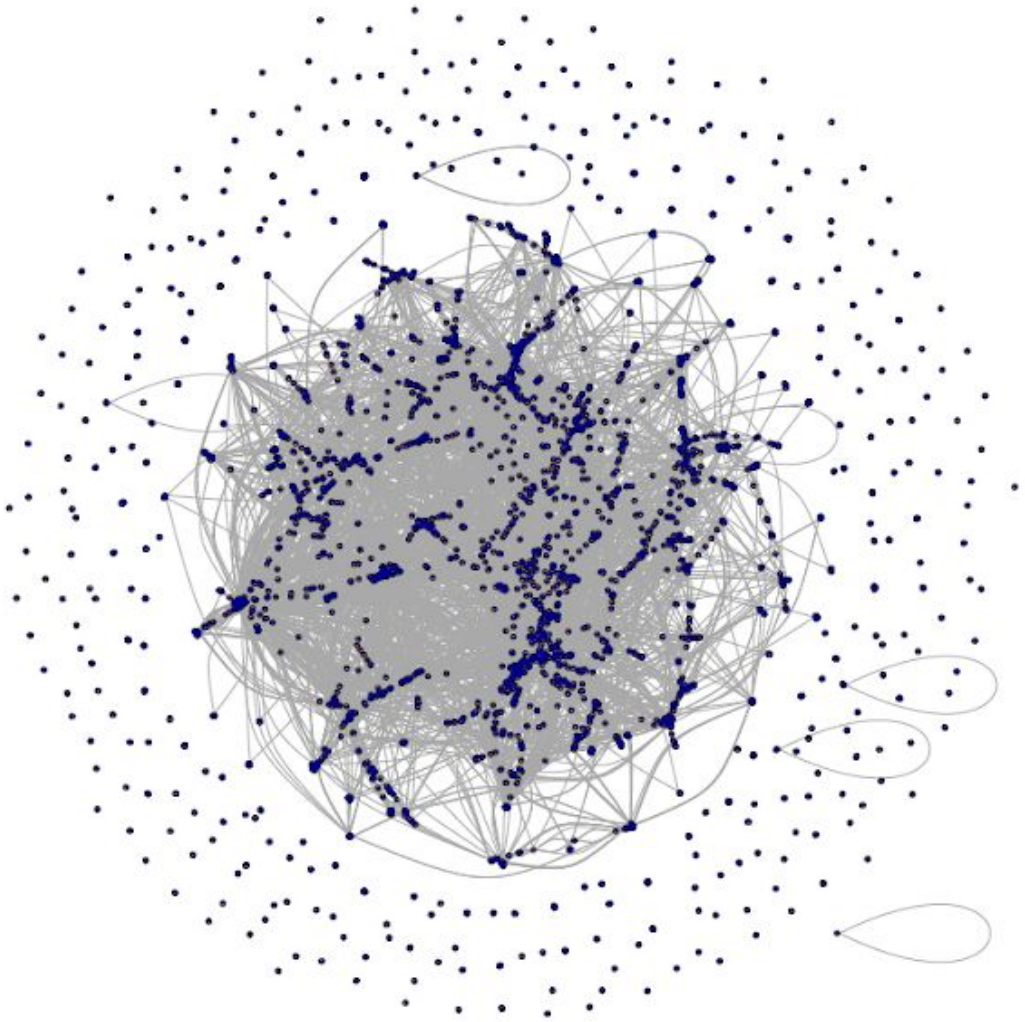
   Tujuan Analisis yang dilakukan kelompok 13 adalah untuk menemukan peneliti-peneliti yang mungkin menarik untuk seseorang jika ia sedang sering membaca sekumpulan *paper* oleh seorang peneliti, sedang sering membaca sekumpulan *paper* oleh beberapa peneliti yang saling berhubungan, atau sudah berkolaborasi dengan sejumlah peneliti. Hal ini lahir dari asumsi bahwa peneliti-peneliti yang berkolaborasi memiliki ketertarikan atau fokus pada bidang yang sama dalam penelitian.

   Ketika seseorang sering membaca *paper* hasil seorang peneliti, orang tersebut dapat dinyatakan sedang tertarik dengan bidang yang digeluti oleh peneliti tersebut. Dengan demikian, jika seorang peneliti menarik bagi seseorang, ada kemungkinan bahwa peneliti lain dengan bidang yang sama dapat menarik juga.

   Ketika seseorang pernah berkolaborasi dengan sejumlah peneliti, peneliti lain yang pernah berkolaborasi dengan peneliti-peneliti tersebut mungkin juga menjadi menarik untuk diajak berkolaborasi dengan orang tersebut.

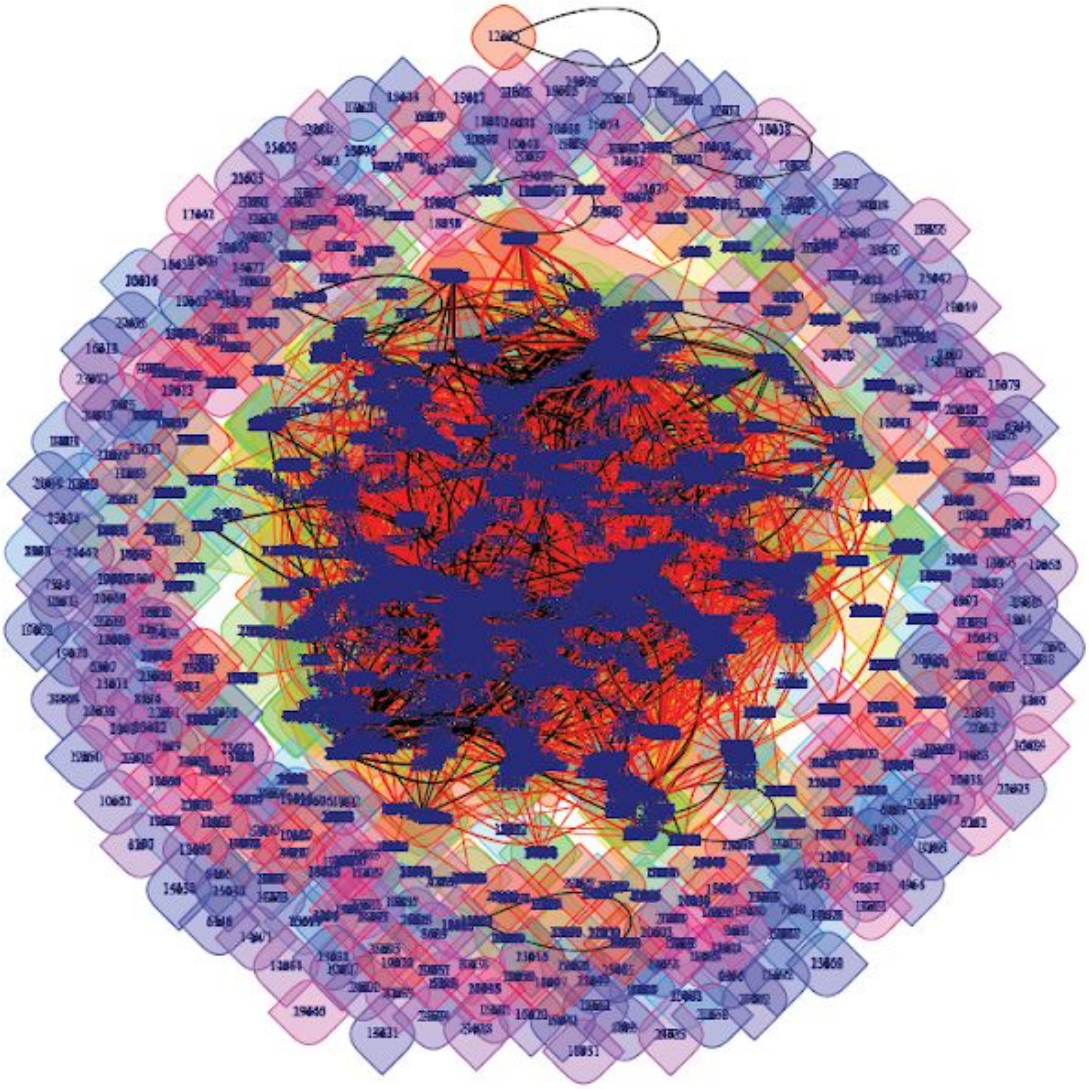
3. Hasil Eksperimen

Eksperimen dilakukan dengan metode visualisasi. Gambar di bawah adalah gambar visualisasi graf kolaborasi peneliti kategori *General Relativity and Quantum Cosmology*. Terdapat jaringan kolaborasi yang kuat (banyak keterhubungannya) antara sejumlah peneliti di bagian tengah graf. Namun demikian, terdapat juga jaringan-jaringan kolaborasi yang terisolasi.



Gambar 2 Visualisasi graf kolaborasi pada *General Relativity and Quantum Cosmology*

Dengan menggunakan iGraph, terdeteksi 703 komunitas dan 355 komponen terhubung untuk 5242 simpul. Sebuah komunitas adalah kumpulan simpul-simpul berdekatan yang saling terhubung, sementara sebuah komponen adalah kumpulan semua simpul yang terhubung baik secara langsung atau tidak. Terdapat sejumlah 4158 simpul yang saling terhubung di bagian tengah visualisasi graf dan sisanya adalah simpul-simpul yang sama sekali tidak terhubung. Gambar di atas dapat divisualisasi lebih lanjut untuk menggambarkan komu/nitas-komunitas yang terbentuk menjadi seperti gambar di bawah ini.

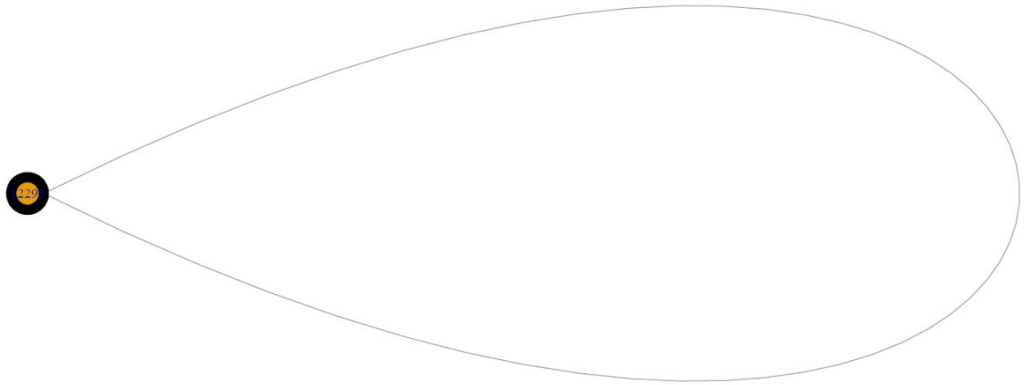


Gambar 3 Visualisasi komunitas pada graf kolaborasi pada *General Relativity and Quantum Cosmology*

4. Analisis Hasil Eksperimen

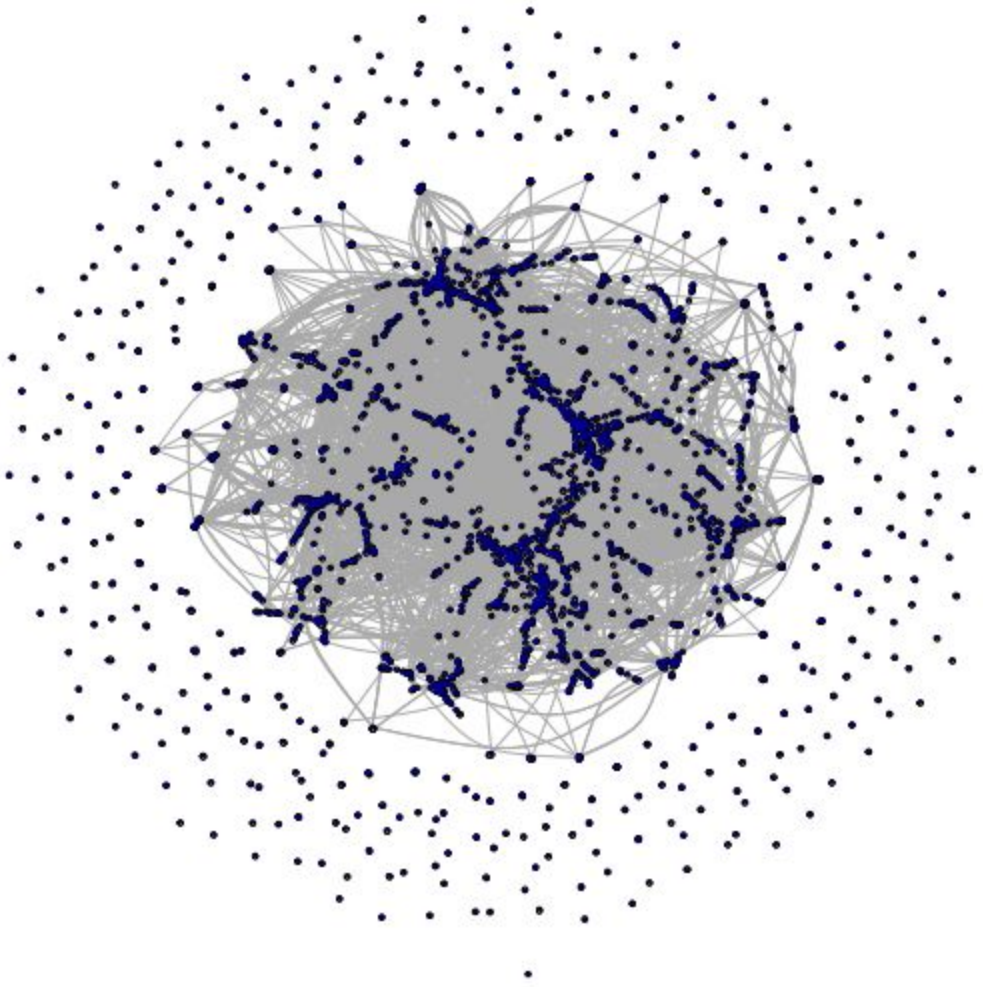
a. Akurasi Data

Ditemukan peneliti yang berkolaborasi dengan dirinya sendiri (bagian bawah kanan visualisasi graf pada bab sebelumnya), yakni peneliti dengan id 12295. Hal ini menunjukkan bahwa data belum 100% akurat, sehingga pembersihan data lebih lanjut perlu dilakukan.



Gambar 4 Peneliti yang memiliki sisi kolaborasi dengan dirinya sendiri

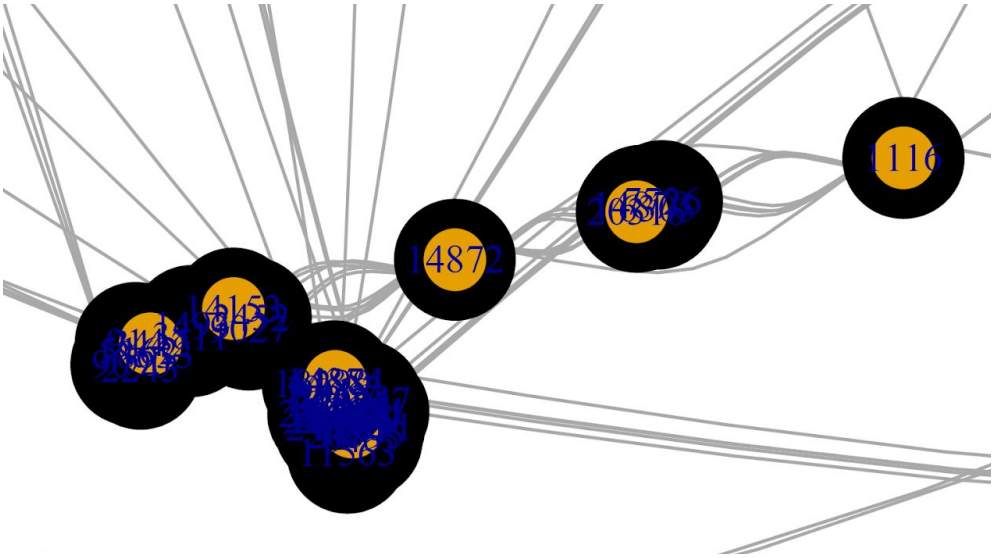
Sementara berikut ini tampilan visualisasi graf dengan menghilangkan *loop-edge* atau peneliti yang berkolaborasi dengan dirinya sendiri.



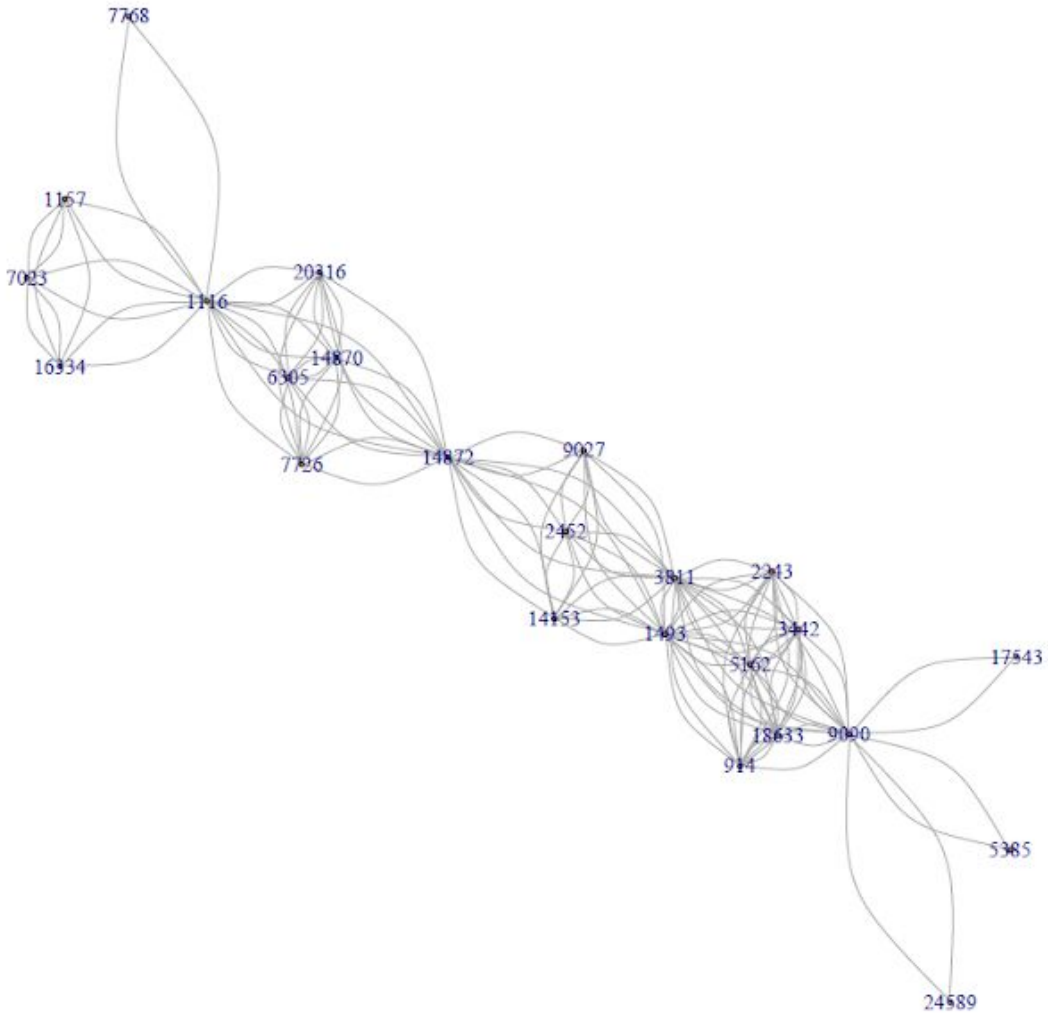
Gambar 5 Visualisasi graf kolaborasi tanpa *loop-edge*

b. Keterhubungan Peneliti

Contoh pemanfaatan graf kolaborasi peneliti adalah sebagai berikut. Jika ada seseorang yang sedang membaca banyak penelitian dari 14872 dan peneliti 1116, dilakukan pemrosesan lebih lanjut untuk melihat hubungan antara kedua peneliti ini.



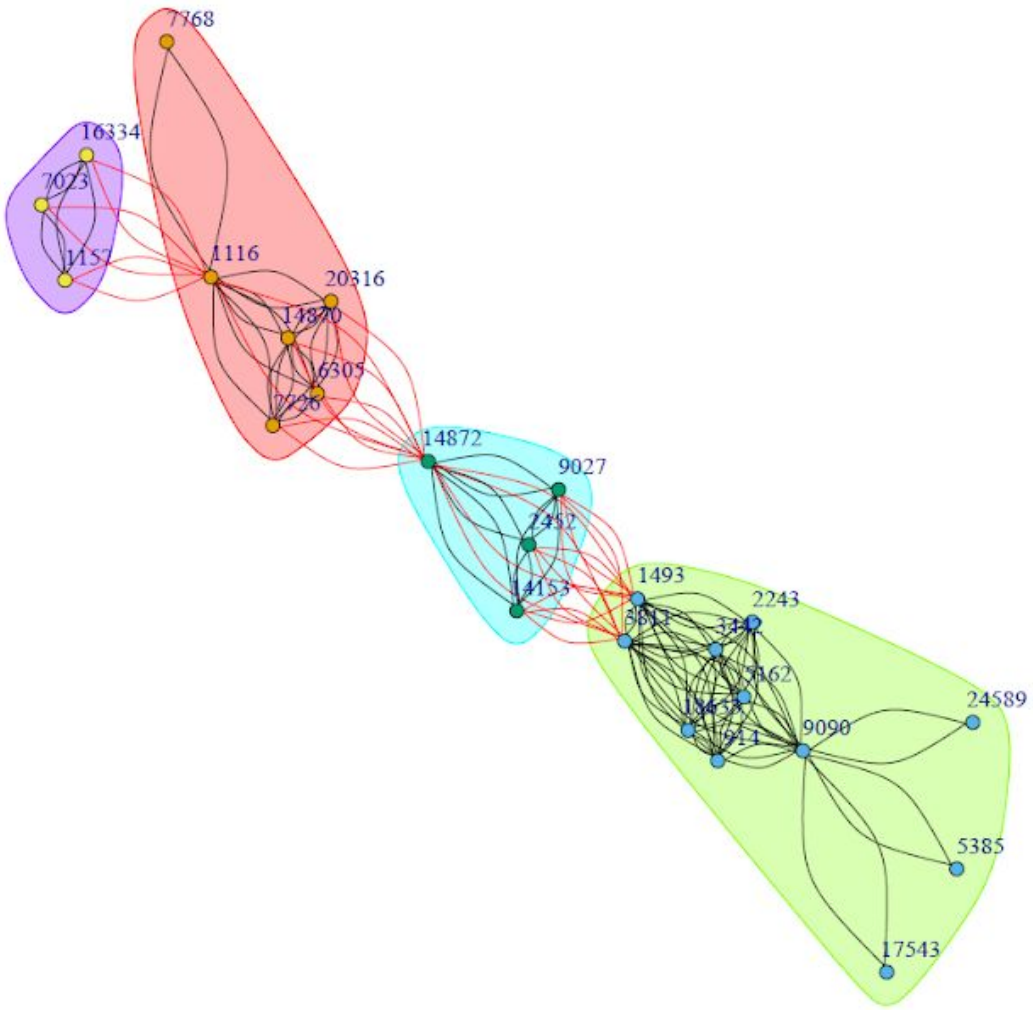
Gambar 6 Peneliti 14872 dan peneliti 1116



Gambar 7 Peneliti-peneliti yang terhubung dengan peneliti 14872 dan peneliti 1116

Gambar di atas merupakan hasil visualisasi dari peneliti-peneliti yang memiliki hubungan dengan peneliti 14872 dan peneliti 1116. Hubungan yang dimaksud adalah peneliti-peneliti yang berasal dari komunitas yang sama (dari analisis komunitas pada seluruh graf yang sudah dibahas pada bab Hasil Eksperimen) beserta peneliti-peneliti yang berhubungan dengan peneliti-peneliti tersebut. Dari pemrosesan lebih lanjut, didapatkan bahwa ternyata 14872 dan 1116 tidak pernah melakukan penelitian bersama, namun mereka pernah melakukan penelitian dengan orang yang sama, yakni peneliti 7726, peneliti 20316, peneliti 6305, dan peneliti 14870. Dengan adanya analisis ini, orang yang sedang sering membaca penelitian dari peneliti 14872 dan peneliti 1116 dapat disarankan untuk membaca penelitian-penelitian dari peneliti 7726, 20316, 6305, dan 14870. Rekomendasi lebih lanjut akan dibahas pada bagian selanjutnya.

c. Keterhubungan Bidang

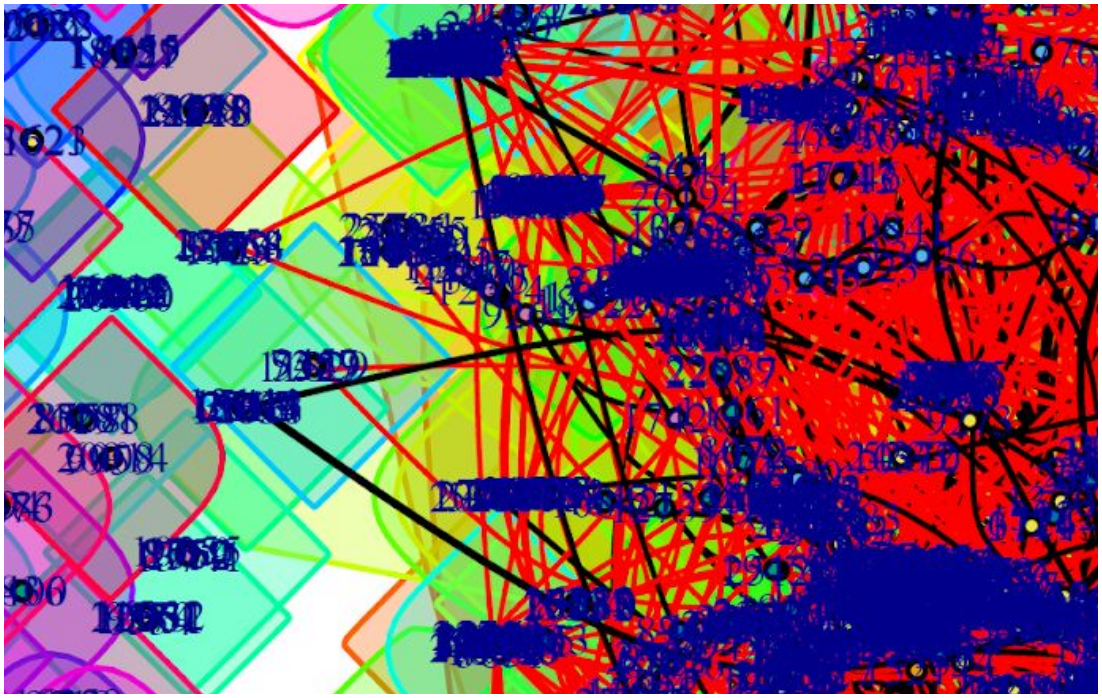


Gambar 8 Tinjauan spesifik komunitas jaringan peneliti yang terhubung dengan peneliti 1116 dan 14782

Berdasarkan identifikasi komunitas pada graf keseluruhan, peneliti 1116 dan 14872 berada pada komunitas yang sama dengan peneliti 6305, 7726, 14870, 20316, 914, 5162, 3811, 9090, 18633, 2243, 1493, 2452, 9027, 14153, 1157, 16334, dan 3442. Tetapi saat dilakukan analisis yang lebih spesifik, dapat dilihat bahwa terdapat pula hubungan-hubungan yang lebih erat di dalamnya. Dengan demikian, dapat ditentukan peneliti yang lebih dahulu direkomendasikan pada seseorang yang sedang sering mempelajari penelitian atau berkolaborasi dengan peneliti 1116 dan 14872. Urutan rekomendasi peneliti adalah sebagai berikut:

1. peneliti-peneliti yang memiliki hubungan langsung dengan 1116 *dan* 14872
2. peneliti-peneliti yang tidak memiliki hubungan langsung dengan 1116 maupun 14872, namun terdapat komunitas yang sama dengan salah satunya pada tinjauan spesifik

3. peneliti-peneliti yang terdapat pada komunitas yang sama dengan salah satunya pada tinjauan keseluruhan
4. peneliti-peneliti lain.



Gambar 9 Hubungan antarkomunitas

Pada gambar 9, dapat dilihat bahwa komunitas satu dengan yang lainnya dapat memiliki keterhubungan. Seperti dikatakan pada bab tujuan, peneliti yang berkolaborasi diasumsikan memiliki ketertarikan pada bidang yang sama. Dengan demikian, komunitas peneliti (peneliti-peneliti yang saling terhubung) mungkin memiliki ketertarikan pada bidang yang sama. Hubungan antarkomunitas dapat menggambarkan dua kemungkinan:

- Bidang yang digeluti berhubungan--sehingga dapat disarankan untuk dipelajari oleh seseorang yang sering membaca *paper* dari sekelompok peneliti; atau
- Kedua komunitas bergerak pada bidang yang sama, namun belum sempat berkolaborasi--sehingga peneliti dapat disarankan untuk melakukan kolaborasi; dan pelajar juga dapat disarankan mempelajari *paper* dari sekelompok peneliti lain dari yang sudah dia baca.

d. Saran Pengembangan

Beberapa hal dapat dilakukan untuk meningkatkan manfaat graf ini. Misalnya ditambahnya informasi tahun kolaborasi terakhir pada sisi. Semakin jauh dari saat ini kolaborasi terjadi, semakin lemah kekuatan sisinya. Dengan demikian, saran yang diberikan baik bagi pembaca *paper* maupun peneliti dapat lebih relevan terhadap zaman. Pasalnya, seorang peneliti dapat memiliki pergeseran minat/bidang dari waktu ke waktu.

Jumlah kolaborasi juga dapat disimpan pada sisi. Semakin sering peneliti berkolaborasi dengan peneliti lainnya, semakin kuat sisinya. Banyaknya jumlah kolaborasi diasumsikan berbanding lurus kesamaan ketertarikan peneliti.

5. Manfaat

Manfaat yang bisa didapat dari eksplorasi data graf ini adalah:

- Mengetahui komunitas-komunitas yang dimiliki antar peneliti sehingga dapat menjadi rekomendasi peneliti untuk dipelajari penelitiannya atau untuk diajak kolaborasi.
- Mengetahui tingkat pengaruh dari seorang peneliti yang dapat diketahui dengan melihat banyaknya peneliti lain terhubung dengannya. Semakin banyak relasi yang terhubung pada peneliti tersebut, maka pengaruhnya akan semakin besar. Dari informasi tersebut, dapat diketahui jalur pengaruh paling efektif dari satu komunitas ke komunitas yang lain dengan melihat keterhubungan suatu peneliti terhadap komunitas lain di luar komunitasnya sendiri. Sehingga apabila perlu dilakukan koordinasi antar komunitas, koordinasi tersebut bisa dilakukan antara peneliti-peneliti yang memiliki pengaruh tersebut.
- Mengetahui peneliti-peneliti dengan latar belakang yang serumpun, sehingga dari komunitas tersebut bisa dibentuk suatu kelompok kerja tertentu.
- Dari peneliti-peneliti yang belum pernah melakukan kolaborasi, bisa saja terbentuk suatu rumpun atau komunitas dengan topik yang baru.
- Komunitas-komunitas yang terhubung, jika mewakili rumpun ilmu yang berbeda, dapat digerakkan untuk melakukan riset yang membutuhkan kolaborasi antarrumpun.