Analisis Jumlah dan Hubungan Streamer pada Twitch dengan Algoritma Degree Centrality dan Strongly Connected Components

Titanio Meiga Batry [1]***, Ananda Khibran***[2]***, Khansa Al Faiziy***[3]***,***

Nur Aini Rakhmawati [4]

Institute Teknologi Sepuluh Nopember

Email : titaniokuliah@gmail.com [1]*, nur.aini@is.its.ac.id* [4]

Abstrak

Pesatnya perkembangan teknologi web memberikan kemudahan bagi masyarakat untuk mengakses berbagai informasi. Layanan Internet berbasis web generasi ketiga (Web 3.0) yang diperkenalkan sebagai Semantik Web. Hal ini bertujuan untuk membantu komputer memahami konten di web. Penerapan Semantic dapat dilakukan untuk mengambil dataset dari Neo4j, yaitu daftar streamer di Twitch. Tujuan analisis data ini adalah untuk mengetahui jumlah pemasukkan chat berupa teks dari user pada tiap streamer aktif dan jumlah moderator yang saling berhubungan pada satu kelompok. Penerapan ini menggunakan algoritma Degree Centrality dan Strongly Connected Components, serta melakukan visualisasi dengan menggunakan Neo4jBloom dari Neo4j itu sendiri. Pada penelitian ini menemukan bahwa riotgames memiliki jumlah pemasukkan chat paling tinggi pada twitch streamer dengan algoritma Degree Centrality serta terdapat dua yang memiliki jumlah moderator yang saling berhubungan yang hasilnya sama dengan algoritma Strongly Connected Components. Pada perhitungan kedua algoritma tersebut dilakukan komputasi dua kali untuk memastikan apakah yang dihasilkan kedua algoritma tersebut memiliki hasil yang konsisten dengan dataset tersebut.

Kata Kunci : Neo4j, Degree Centrality, Strongly Connected Components

1) Pendahuluan

Streamer adalah suatu aktivitas yang dapat melakukan membuat konten seperti game online berupa *pc* atau *mobile*. Sedangkan menjadi game streamer bukanlah hal yang tidak mudah. Selain masalah mengenai peralatan apa saja yang dibutuhkan seorang streamer agar channel streaming terus berkembang. Aplikasi streaming yang sering digunakan adalah Twitch. Twitch adalah suatu aplikasi yang digunakan sebagai streamer untuk melakukan recording permainan *(game online)* berbasis *pc* atau *mobile* dan mayoritas pengunaan aplikasi stream twitch bertaraf internasional hingga penjuru dunia. Maka dari itu rata-rata streamer menggunakan twitch untuk membuat konten game online berbasis *pc*[1].

Para Streamer sendiri tidak sekedar membikin konten secara individu tetapi kebanyakan sekarang para streamer memiliki komunitasi tersendiri agar mencapai tujuan secara kebersamaan. Pada streamer di twitch, streamer tersebut memiliki peran seperti moderator, editor, akses vip, dll. Di platform twitch sendiri memiliki fitur chat sebagai penonton untuk melakukan chat secara teks.

Pada penelitian ini melakukan analisis algoritma dengan Semantic Web. Semantic Web adalah cara untuk merevolusi Internet dengan menggabungkan interaktivitas pengguna, kolaborasi informasi, dan kecerdasan buatan ke dalam satu situs web. Dengan menggunakan Semantic Web maka data dan informasi yang tersebar di Internet dan membentuk jaringan dengan berbagai jenis data dan informasi. Data grafik masih berupa data mentah, sehingga perlu diolah dan dianalisis agar data mentah tersebut menjadi pengetahuan yang dapat membantu manusia[2].

Data yang digunakan perhitungan algoritma ini diambil dari dataset Neo4j, yaitu daftar streamer di Twitch. Neo4j merupakan sebuah alat sandbox yang digunakan basis data graph dengan standar SQL serta bisa menampilkan dataset pada relasi antara tiap node secara visual graph.

Algoritma yang digunakan perhitungan dataset ini yaitu *Degree Centrality* dan *Strongly Connected Components*. *Degree Centrality* digunakan untuk mengetahui berapa total pemasukkan chat pada tiap seoarang *streamer* dari penonton atau *user*. Sedangkan *Strongly*

Connected Components digunakan untuk mengetahui berapa jumlah streamer aktif yang saling terhubung terhadap relasi moderator pada kelompok. Kedua algoritma tersebut akan dijalankan dua kali untuk memastikan apakah dari kedua algoritma tersebut mengeluarkan hasil yang konsisten.

Manfaat pada melakukan penelitian ini yaitu untuk mengetahui apakah algoritma tersebut selalu memiliki hasil yang konsisten dalam komputasi kerja pada proses pengelolahan suatu data.

2) Tinjauan Pustaka

Pada bagian ini, definisi dari beberapa kosakata akan dijabarkan lebih detail melalui referensi dann penjelasanya

2.1 Neo4j

Neo4j adalah database grafik yang diperkenalkan pada tahun 2007. Pada Neo4j, graph memiliki properti yang terdiri dari node, relasi, direction, dan label. Node adalah aktor yang memiliki peran dalam jaringan, relasi adalah hubungan antar node, direction adalah arah hubungan dari satu node ke node lainnya, dan label digunakan untuk menunjukkan hubungan dengan node[3].

2.2 Degree Centrality

Menurut V Latora dan M Marchiori (2007) *Degree Centrality* adalah salah satu cara untuk mengukur sentralitas dalam suatu jaringan yang fokus terhadap seberapa banyak suatu node yang saling terhubung dengan node lainnya[4]. V Latora dan M Marchiori (2007) di dalam tulisannya juga mendefinisikan salah satu cara untuk mengukur sentralitas dalam suatu jaringan yang didasarkan terhadap kemampuan kelompok suatu node berhubungan dengan kelompok node lainnya. Rumus di bawah ini merupakan rumus untuk menghitung nilai Degree Centrality setiap node dalam jaringan[5].

$$C_i^{\mathrm{D}} = \frac{k_i}{N-1} = \frac{\sum_{j \in \mathbf{G}} a_{ij}}{N-1}, \quad \text{....[4]}$$

Dimana :

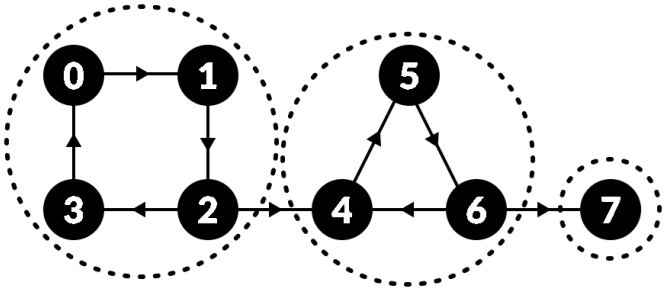
C_i^D = bobot nilai degree centrality

ki = derajat simpul ke- i

N = Jumlah node dalam suatu jaringan

2.3 Strongly Connected Components

Algoritma *Strongly Connected Components* (SCC) menemukan himpunan maksimal node terhubung dalam grafik berarah. Suatu himpunan dikatakan komponen terhubung kuat jika terdapat lintasan berarah antara setiap pasangan simpul dalam himpunan tersebut. Algoritma ini sering digunakan pada awal proses analisis grafik untuk membantu kita memahami struktur grafik[6]. Berikut contoh gambar *Strongly Connected Components* :



Gambar 1 Contoh Strongly connected components [7]

Pada Gambar 1 dapat dijelaskan bahwa node 0,1,2,3 mempunyai ikatan yang saling terhubung jadi node tersebut terbentuk satu komponen, begitu juga 5,4,6 terbentuk satu komponen, dan 7 terbentuk satu komponen meskipun node itu sendiri.

3) Dataset

Dataset yang digunakan pada penelitian ini yaitu dataset daftar streamer di twitch yang berasal dari Neo4j. Dataset ini memiliki 5 Node Labels yaitu disajikan pada tabel berikut :

Label	**Deskripsi**
Stream	Akun twitch yang aktif sebagai streamer dalam melakukan live streaming.
User	Akun twitch tapi tidak aktif dalam melakukan live streaming, mungkin hanya sebagai penonton atau moderator.
Game	Jenis game.
Language	Bahasa yang digunakan pada pemilik akun twitch
Team	Nama tim Streamer

Serta mempunyai 6 tipe Relasi *(Relationship)* yaitu :

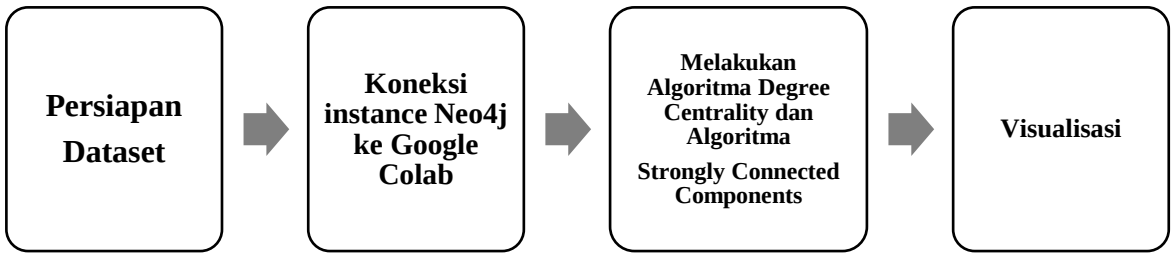
Relasi	**Deskripsi**
CHATTER	Relasi CHATTER yaitu dimana user twitch melakukan chat kepada streamer. Relasi ini ditemukan pada label Stream dan User
HAS_LANGUAGE	Relasi HAS_LANGUAGE yaitu penggunaan bahasa pada user twitch
HAS_TEAM	Relasi HAS_TEAM yaitu dimana seorang streamer miliki tim

MODERATOR	Relasi MODERATOR yaitu dimana user twitch memiliki peran moderator kepada streamer
PLAYS	Relasi PLAYS yaitu dimana streamer memiliki game yang kemungkinan sering dimainkan
VIP	Relasi VIP yaitu dimana user twitch memiliki peran akses vip kepada streamer

Dataset ini memiliki data dengan total sejumlah 4680871 node yang diketahui dengan query sendiri. Pada dataset tersebut terdapat satu node yang memiliki dua label yaitu *'Stream'* dan *'User'*. Maksud dalam gabungan dua label dalam satu node tersebut berarti tidak semua akun twich melakukan live streaming tetapi hanya sebagai penonton atau moderator jika node tersebut hanya memiliki label yaitu *'User'*. Jadi ada node yang memiliki satu label yaitu *'User'*, dan node yang memiliki label *'Stream'* selalu didampingin dengan label *'User'*.

4) Metode Penelitian

Penelitian ini dilakukan sesuai dengan alur yang ditunjukkan pada gambar dibawah ini.



Gambar 2 Alur Metode Penelitian

Tahapan pada alur tersebut meliputi Koneksi instance Neo4j ke Google Colab, perhitungan Algoritma Degree Centrality dan Algorithms Strongly Connected Components, dan Visualisasi

4.1 Persiapan Dataset

Dataset Twitch yang dipake berasal dari Neo4j. Dataset ini memiliki 5 Node Labels dan 6 tipe Relasi. Karena dataset ini dari Neo4j sudah lengkap tanpa melakukan modifikasi query pada dataset tersebut, dataset ini langsung bisa dilakukan query algoritma.

4.2 Koneksi instance Neo4j ke Google Colab

Berikut query python untuk koneksi dataset twitch yang sudah ada di Neo4j ke Google Colab:

```
#connect project/dataset from Neo4j

driver = GraphDatabase.driver(
 "bolt://34.207.70.27:7687",
  auth=basic_auth("neo4j", "condition-solders-spiral"))
```

Gambar 1. Query untuk koneksi instance Neo4j ke Google Colab

Setelah melakukan koneksi dataset tersebut, Google Colab bisa melakukan query SQL dan bisa melakukan modifikasi dataset di Neo4j

4.3 Algoritma Degree Centrality dan Algoritma Strongly Connected Components

Perhitungan algoritma dilakukan di Google Colab .

a. Algoritma Degree Centrality

Dalam analisis menggunakan algoritma ini, parameter yang digunakan untuk perhitungan yaitu *'User'* sebagai label dan **CHATTER** sebagai relasi pada sesama label *'User'*. Pemilihan parameter tersebut untuk mengetahui jumlah nilai pemasukan chatter pada seorang streamer di Twitch. Pada algoritma ini akan dilakukan dua kali query perhitungan untuk membandingkan apakah yang dihasilkan sama pada kedua query tersebut.

b. Algoritma Strongly Connected Components

Dalam analisis menggunakan algoritma ini, parameter yang digunakan untuk perhitungan yaitu *'Stream'* sebagai label dan **MODERATOR** sebagai relasi pada sesama label *'Stream'*. Pemilihan parameter tersebut untuk mengetahui berapa jumlah komponen yang saling berhubungan pada moderator yang aktif melakukan live streaming . Pada algoritma ini akan dilakukan dua kali query perhitungan untuk membandingkan apakah yang dihasilkan sama pada kedua query tersebut.

4.4 Visualisasi

Dalam melakukan ini hanya menampilkan visual graph dengan label dan relasi yang dibutuhkan dari melakukan kedua algoritma tersebut. Karena dataset yang didapatkan dari Neo4j, visualisasi graph pada dataset Twitch dengan menggunakan Neo4j Bloom dari Neo4j itu sendiri.

5) Hasil dan Pembahasan

5.1 Algoritma c

Pada algoritma ini akan dilakukan dua kali perhitungan query. Berikut hasil dari perhitungan pertama Algoritma Degree Centrality dengan parameter **User** sebagai label dan **CHATTER** sebagai relasi antara sesama Stream pada Tabel 1:

Tabel 1 hasil Pertama Algortima Degree Centrality

Streamer('user')	**chatterIN**('CHATTER')
riotgames	194546
xqcow	171902
itsbigchase	120726
esl_csgo	111467
enardo	109121

Pada tabel 1 dijelaskan bahwa jumlah *chatter* dari stream yang paling banyak pemasukan yaitu *'riotgames'*, lalu yang kedua yaitu *'xqcow'*, dan seterusnya yang didapatkan oleh label **User** lain.

Selanjutnya dilakukan perhitungan query kedua dengan query yang sama. Berikut hasil kedua perhitungan algoritma ini yang ditampilkan pada Tabel 2:

Tabel 2 hasil Kedua Algortima Degree Centrality

Streamer('user')	**chatterIN**('CHATTER')
riotgames	194546
xqcow	171902
itsbigchase	120726
esl_csgo	111467
enardo	109121

Pada Tabel 2 dapat dijelaskan bahwa perhitungan kedua memiliki hasil yang sama dengan hasil perhitungan pertama pada Tabel 1.

5.2 Algoritma Strongly Connected Components

Dalam melakukan algoritma Strongly Connected Components perlu mengetahui jumlah label *'Stream'* yang saling terhubung dalam satu kompleks pada relasi **MODERATOR**. Pada algoritma ini akan dilakukan dua kali perhitungan query. Berikut hasil pertama perhitungan algoritma ini yang ditampilkan pada Tabel 3:

Tabel 3 hasil Pertama Algoritma Strongly Connected Components

Component	**Jumlah Stream**
127	5
3757	5
212	4
1084	4
20	3

Pada Tabel 3 dijelaskan bahwa komponen yang memiliki jumlah label tertinggi yaitu pada nomer 127 dan 3757 yang memiliki 5 ***Jumlah Stream***. ***Jumlah Stream*** merupakan jumlah label Stream pada komponen. Selanjutnya menampilkan nama streamer yang terdapat pada komponen tersebut. Berikut hasil dari penampilan nama streamer pada komponen 127 dan 3757:

Tabel 4 hasil penampilan nama streamer pada komponen 127

Name	**Component**
f1ashko	127
romanovalera	127
c_a_k_e	127
dreadztv	127
just_ns	127

Tabel 5 hasil penampilan nama streamer pada komponen 3757

Name	**Component**
mjod	3757
drak0ola	3757
iinwafqht	3757
abuswe7l	3757
osamah	3757

Pada Tabel 4 dan 5 dapat dijelaskan bahwa nama streamer tersebut tercantum pada komponen tersebut yang sudah diperhitungkan oleh algoritma Strongly Connected Components.

Selanjutnya dilakukan perhitungan kedua dengan query yang sama. Berikut hasil kedua perhitungan algoritma ini yang ditampilkan pada Tabel 6:

Tabel 6 hasil Kedua Algoritma Strongly Connected Components

Component	Jumlah Stream
126	5
3759	5
211	4
1075	4
19	3

Pada Tabel 6 dijelaskan bahwa komponen yang memiliki jumlah label tertinggi pada nomer 126 dan 3759 yang memiliki 5 label Stream *(Jumlah Stream)* berbeda dengan hasil pada Tabel 3. Nilai count pada perhitungan kedua memiliki nilai yang sama dengan perhitungan pertama pada Tabel 3. Selanjutnya menampilkan nama streamer yang terdapat pada komponen tersebut. Berikut hasil dari penampilan nama streamer pada komponen 126 dan 3759:

Tabel 7 hasil penampilan nama streamer pada komponen 127

Name	Component
f1ashko	126
romanovalera	126
c_a_k_e	126
dreadztv	126
just_ns	126

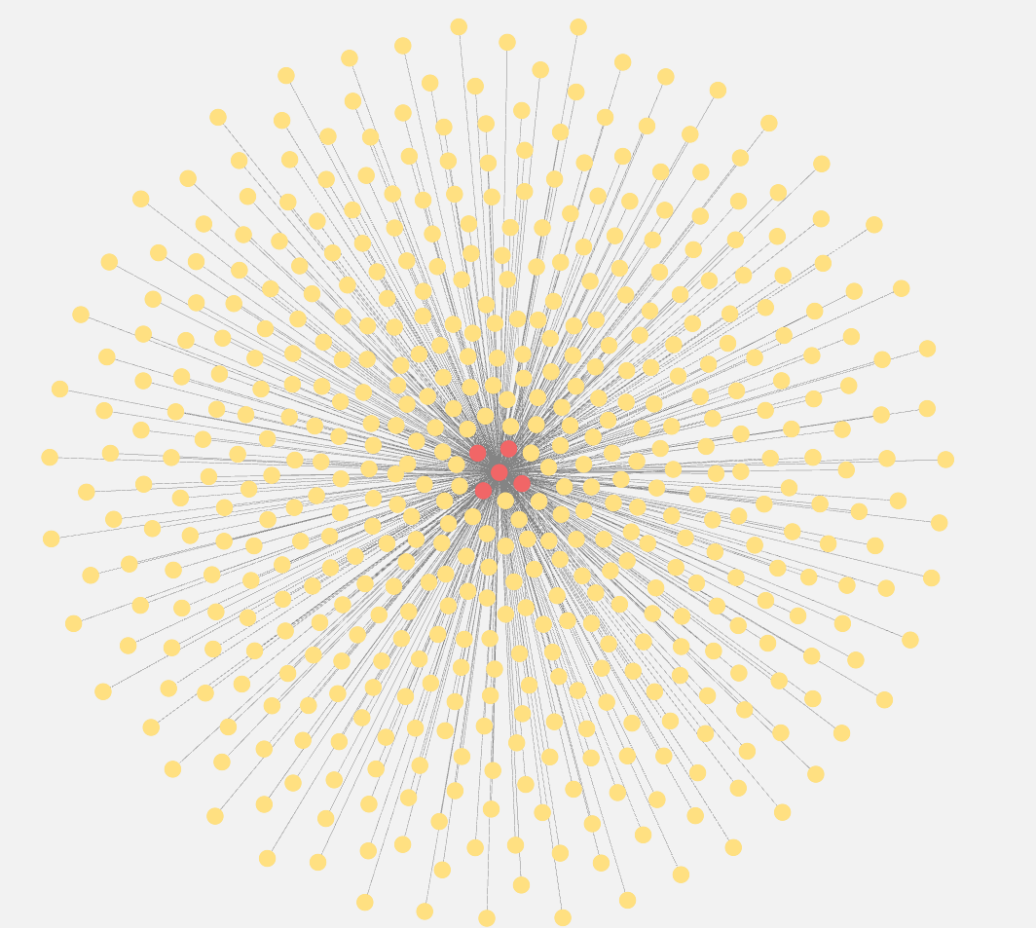
Tabel 8 hasil penampilan nama streamer pada komponen 3757

Name	Component
mjod	3759
drak0ola	3759
iinwafqht	3759
abuswe7l	3759
osamah	3759

Pada Tabel 7 dan 8 bisa dijelaskan bahwa dibandingkan dengan perhitungan pertama memiliki nama pengeluaran yang sama pada Tabel 4 dan 5 meskipun nomer komponen yang berbeda.

5.3 Visualisasi

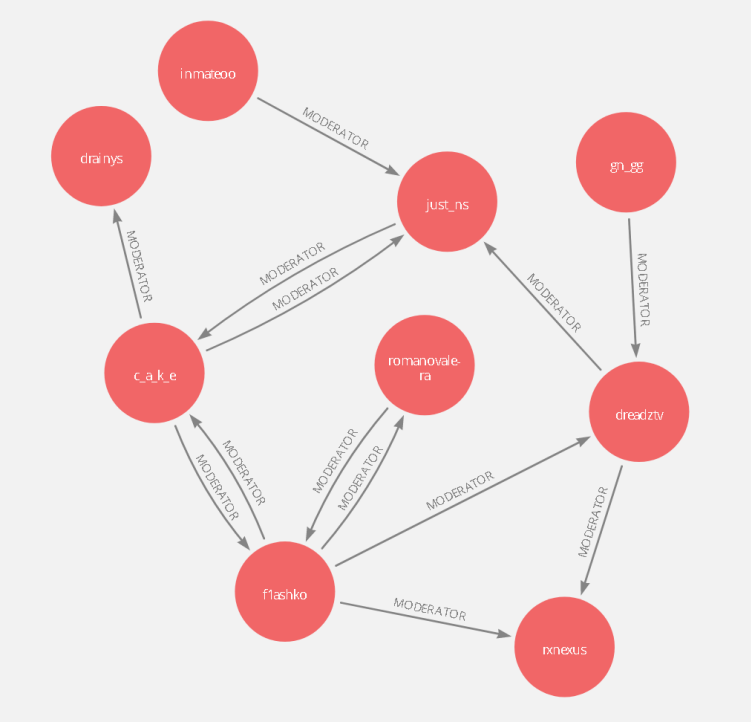
Pada bagian ini menampilkan visualisasi dari hasil kedua algoritma tersebut dengan Neo4jBloom. Berikut hasil visualisasi pada uji coba algoritma Degree Centrality :



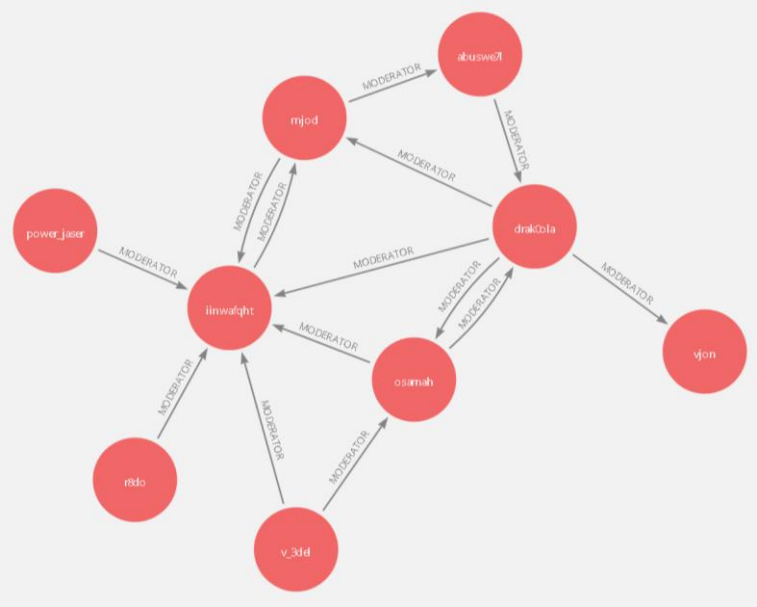
Gambar 3 Visualisasi dari hasil algoritma Degree Centrality

Pada gambar 3 menujukan warna kuning adalah sebagai label *'User'* yang melakukan chat ke streamer dengan relasi CHATTER yang dibatasi 500 node pada label *'User'* dan label *'Stream'* yang berwarna merah. Pada pengambilan label *'Stream'* yaitu hanya mengambil diatas urutan 5 terbesar dari pemasukkan chat yaitu *'riotgames'*, *'xqcow'*, '*itsbigchase'*, *'esl_csgo'*, dan *'enardo'*.

Hasil algoritma Strongly Connected Components ada 2 komponen yang memiliki jumlah label tertinggi yaitu 5. Berikut hasil visualisasi dari kedua hasil algoritma Strongly Connected Components :



Gambar 4 hasil visualisasi pada komponen no.127/no.126



Gambar 5 hasil visualisasi pada komponen no.3757/no.3759

Pada gambar 4 dan 5 bahwa kelima label tersebut memiliki relasi yang saling terhubung antara sesama label. Pada gambar 4 menujukan bahwa kelima label tersebut memilki ikatan

moderator yang saling berhungan. Dilihat pada keempat label tersebut memiliki hubungan yang berotasi antara *‘c_a_k_e’* , *‘f1ashko’, dreadztv*, dan *‘just_ns’* meskipun *‘romanovalera’* tidak mengikuti rotasi pada keempat label tersebut tetapi masuk pada komponen tersebut karena *‘rommanovalera’* hanya saling berhubungan pada komopenen tersebut. Pada gambar 5 menujukan bahwa kelima label tersebut memilki ikatan moderator yang saling berhungan. Dilihat pada kelima label tersebut memiliki hubungan yang berotasi antara *‘drak0ola’*, *‘osamah’, ‘iinwafqht’, ‘mjod’*, dan *‘abuswe7l’*.

6) Kesimpulan

Berdasarkan perhitungan algoritma Degree Centrality bahwa nilai pemasukkan chatter yang paling tinggi yaitu ‘riotgames’ yang didapatkan oleh label Stream lain, sehingga ‘riotgames’ memiliki pemasukkan chat yang paling banyak di Twitch. Dalam perhitungan algoritma Degree Centrality menghasilkan pengeluaran terakurat pada dataset Twitch dari Neo4j saat melakukan query ulang dengan hasil yang sama dengan algoritma.

Berdasarkan perhitungan algoritma Strongly Connected Components bahwa ada dua komponen memiliki nilai jumlah label stream tertinggi yaitu berjumlah 5 label. Namun memiliki nomer komponen berbeda pada perhitungan pertama dan kedua setelah melakukan perhitungan query tetapi memiliki hasil yang tetap sama pada pengularan nama streamer dengan algoritma ini. Kemungkinan dataset Twitch dari Neo4j ini memiliki data yang sangat besar sehingga menggunakan algoritma Strongly Connected Components tidak bisa konsisten pada dataset ini.

7) Daftar Pustaka

[1] Abian Widyadhana, “Ingin Jadi Game Streamer Sukses? Pastikan Kamu Mengikuti Tips Ini,” *https://www.duniagames.co.id/*. [Online]. Available: https://www.duniagames.co.id/discover/article/ingin-jadi-game-streamer-sukses-pastikan-kamu-mengikuti-tips-ini.

[2] A. Perwiradewa, A. N. Rofiif, and N. A. Rakhmawati, “Visualisasi Pemain Sepak Bola Indonesia pada DBPedia dengan menggunakan Node2Vec dan Closeness Centrality,” *J. Buana Inform.*, 2020.

[3] K. Mufidah, N. Syahputra, and N. A. Rakhmawati, “ANALISIS AKTOR POPULAR DAN SUTRADARA BERPENGARUH BERDASARKAN DATA DBPEDIA MENGGUNAKAN ALGORITMA CLOSENESS CENTRALITY DAN NODE2VEC,” *Maj. Ilm. UNIKOM*, 2020.

[4] V. Latora and M. Marchiori, “A measure of centrality based on network efficiency,” *New J. Phys.*, vol. 9, no. 6, pp. 188–188, Jun. 2007.

[5] A. Rahman, M. A. Naufal, and N. A. Rakhmawati, “Analisis Pengaruh Wilayah dan Partai Politik Terhadap Hubungan Para Aktivis di Indonesia Menggunakan Algoritma Degree Centrality,” *J. Inform. Upgris*, 2020.

[6] Neo4j, “Strongly Connected Components.” [Online]. Available: https://neo4j.com/docs/graph-data-science/1.8/algorithms/strongly-connected-components/. [Accessed: 02-Dec-2021].

[7] “Strongly Connected Components,” *Programiz.com*. [Online]. Available: https://www.programiz.com/dsa/strongly-connected-components.