T.C.
GEBZE TEKNİK ÜNİVERSİTESİ

Bilgisayar Mühendisliği Bölümü

Neo4j Veritabanı'nda Sosyal Medya Verisi İçin Benzerlik Metriklerinin Geliştirilmesi

Yunus ÇEVİK

Danışman
Dr. Öğr. Üyesi Burcu YILMAZ

Mayıs, 2019
Gebze, KOCAELİ

T.C.
GEBZE TEKNİK ÜNİVERSİTESİ

Bilgisayar Mühendisliği Bölümü

Neo4j Veritabanı'nda Sosyal Medya Verisi İçin Benzerlik Metriklerinin Geliştirilmesi

Yunus ÇEVİK

Danışman
Dr. Öğr. Üyesi Burcu YILMAZ

Mayıs, 2019
Gebze, KOCAELİ

Bu çalışma ..../..../2019 tarihinde aşağıdaki jüri tarafından Bilgisayar Mühendisliği Bölümü'nde Lisans Bitirme Projesi olarak kabul edilmiştir.

Bitirme Projesi Jürisi

Danışman Adı	Dr. Öğr. Üyesi Burcu YILMAZ	
Üniversite	GEBZE TEKNİK ÜNİVERSİTESİ	
Fakülte	MÜHENDİSLİK FAKÜLTESİ	

Jüri Adı	Dr. Murat ŞEKER	
Üniversite	GEBZE TEKNİK ÜNİVERSİTESİ	
Fakülte	MÜHENDİSLİK FAKÜLTESİ	

ÖNSÖZ

Bitirme projemin planlanmasında, araştırılmasında, yürütülmesinde ve oluşumunda ilgi ve desteğini esirgemeyen, engin bilgi ve tecrübelerinden yararlandığım, yönlendirme ve bilgilendirmeleriyle çalışmamı bilimsel temeller ışığında şekillendiren sayın danışman hocam Dr. Burcu YILMAZ' a ve bu çalışmayı destekleyen Gebze Teknik Üniversitesi'ne sonsuz ve içten teşekkürlerimi sunarım.

Ayrıca eğitimim süresince bana her konuda tam destek veren aileme ve bana hayatlarıyla örnek olan tüm hocalarıma saygı ve sevgilerimi sunarım.

Mayıs, 2019 **Yunus ÇEVİK**

İÇİNDEKİLER

ŞEKİL LİSTESİ

KISALTMA LİSTESİ

CQL : Cypher Query Language

ML : Machine Learning (Makine Öğrenmesi)

ÖZET

Projenin amacı Neo4j veritabanında tutulan sosyal medya verilerinin, benzerlik metrikleri geliştirilerek, çeşitli makine öğrenmesi algoritmaları ile karşılaştırılmasıdır. Günlük hayatta insanlar, sosyal medya yardımı ile birbirlerine onlarca hatta yüzlerce mesaj gönderir. Örnek olarak, Twitter üzerinde insanlar anlık iletişim için oldukça önemli bir yere sahiptir. Twitter yorum, mesaj, bilgi, eleştiri, satış ve tanıtım yapılan etkili bir sosyal alandır.

Bu proje ile MySQL veritabanında tutulan Tweet bilgilerinin Jupyter Notebook programında Python programlama dili ile bağlantı kurularak erişilmesi sağlanmıştır. Tweet bilgileri Neo4j Graph Database' ine ilişkili bir şekilde çizgeler olarak kaydedilir.

Neo4j de yer lan Tweet bilgileri içerisinden herhangi ikisi arasında çizge-izomorfiği (graph – isomophism), çizge düzenleme mesafesi (graph - edit - distance) ve çizgelerdeki Tweet metinlerinin "Cosine Similarity" algoritması ile karşılaştırılır.

SUMMARY

The aim of the project is to compare social media data held in the Neo4j database with similar machine learning algorithms by developing similarity metrics. People in daily life send tens or even hundreds of messages to each other with the help of social media. Twitter is very important for people to be able to communicate instantly. Twitter is an effective social space where comments, messages, information, criticism, sales and publicity are made.

With this project, Tweet information held in MySQL database has been accessed by contacting Python programming language in Jupyter Notebook program. After the Tweet information is also saved as graphs associated with the Neo4j Graph Database. It is compared with the "Cosine Similarity" algorithm of the Tweet texts in the graphs and isomophism between the two of the tweets included in the Neo4j, as well as the graph - isomophism.

1. GİRİŞ

Günümüzde insan ilişkilerinde çok yoğun bir şekilde kullanılan internet üzerinde çeşitli paylaşımlara izin veren sistemleri içinde barındıran çift taraflı paylaşım imkanı veren sistemlerin tamamını içinde barındıran sisteme sosyal medya denilmektedir. Sosyal medya denilince aklımıza hemen eski sistemler gelmektedir. Bu sistemlere bakacak olursak gazete, dergi, radyo ve tv gibi eski iletişim araçları aklımıza gelmektedir. Bunların insanlara tanımış olduğu tek taraflı bilgi paylaşımı ile tek taraflı içerik paylaşımı sistemi artık günümüzde internetin gelmesi ve bu eski sistemleri içine almasıyla birlikte tek taraflı paylaşımlardan çift taraflı paylaşımlara geçmiştir. Bu sayede insanlar karşılıklı olarak hem görüntü hem ses ve yazı anlamında birçok yönde içerik paylaşımına girmiş ve sosyal medyanın faydalı olmasını sağlamışlardır.

Sosyal medya, tüm dünyanın bir arada iletişime geçtiği; yorum, mesaj, bilgi, eleştiri, satış ve tanıtım yaptığı etkili bir sosyal mecradır. Sosyal medya yardımı ile kurgulanan, doğru uygulanan bilgi ve içeriklerle kitlelere ulaşmak, katılımcı sağlamak, günümüzde sosyal medyanın en büyük gücüdür. Twitter bu sosyal alanda anlık iletişim için oldukça önemli bir yere sahiptir. Twitter kullanıcı sayısının gün geçtikçe artması ve kullanımının giderek yaygınlaşması da bunu destekler niteliktedir[1].

Sosyal medya aracılığı ile yapılan mesajlaşmaların ve mesajlaşan kişilerin bilgileri belirli bir çizge(graph) yapısı içerisinde veritabanlarında bulundurulur.

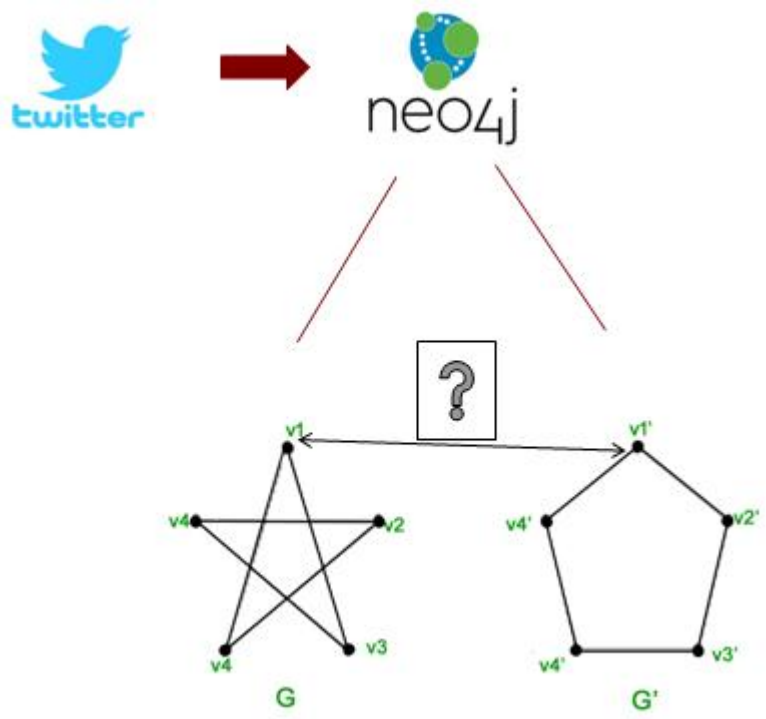
Projenin amacı, Neo4j veritabanında tutulan sosyal medya verilerinin, benzerlik metrikleri geliştirilerek, çeşitli makine öğrenmesi algoritmaları ile karşılaştırılmasıdır.

Neo4J, Neo4J Technology isimli firma tarafından açık kaynak olarak dağıtılan ve geliştirilen graph veritabanıdır.

Verileri ve verilerin arasındaki ilişkileri saklamak amacıyla graf (graph) yapısı kullanan, açık kaynak kodlu bir NoSql veritabanı altyapısıdır. Daha çok graf yapısına göre çalıştığından dolayı ağ, harita, sosyal ağ gibi bir birine bağlı düğümlerin saklandığı ve işlendiği alanlarda kullanılmaktadır.

1.1. PROJENİN TANIMI

İnsanlar, sosyal medya yardımı ile birbirlerine onlarca hatta yüzlerce mesaj gönderebiliyor. Sosyal medya aracılığı ile yapılan mesajlaşmaların ve mesajlaşan kişilerin bilgileri belirli bir çizge(graph) yapısı içerisinde veritabanlarında bulundurulur.



Şekil 1: Sistemin Genel Yapısı

.

1.2. PROJENİN NEDENİ VE AMAÇLARI

Sosyal medya verilerinin Neo4j veritabanına kaydedilmesi ve tutulan veriler üzerinden sorguların gerçekleştirilmesi, benzerlik metrikleri geliştirilerek, çeşitli makine öğrenmesi algoritmaları ile karşılaştırılmasıdır.

2. MALZEME VE YÖNTEM

Bu bölümde giriş bölümünde bahsedilen proje amaçları doğrultusunda projenin gereksinimleri ele alınacaktır. Projenin alt yapısını oluşturan makine öğrenmesi, Neo4j veritabanı, MySQL veritabanı, çizge benzerlik metrikleri, makine öğrenmesi gibi gereksinimlerden bu bölümde bahsedilecektir.

2.1. PROJE GEREKSİNİMLERİ

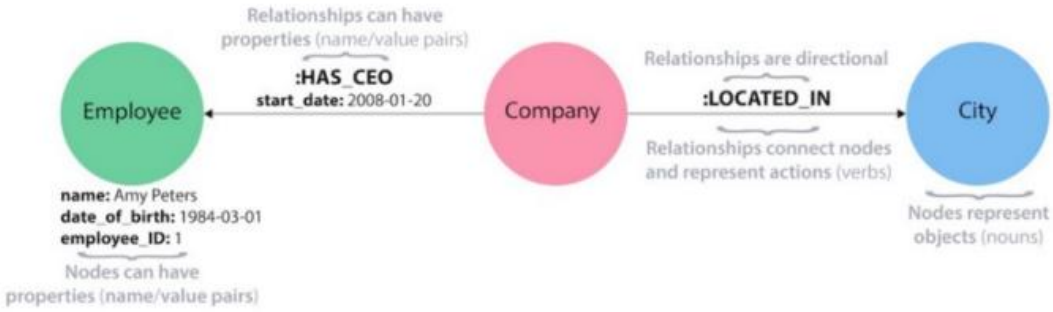
2.1.1. NEO4J VERİTABANI

Neo4j adının sonundaki 'j' harfinden çıkarım yapabileceği üzere Java ile yazılmıştır. Dolayısıyla eğer bir java uygulama geliştiriliyorsa uygulama içine Neo4j'yi entegre edip kullanılabilir. Bunun yanında bir RESTapi sağladığı için istenilen herhangi bir dille, istenilen bir platformda yazılan herhangi bir kodla da kullanılabilir. REST api'yi kullanırken karmaşık sorgular gönderebilk için "cypher" adında sql benzeri fakat sql'e göre fazlaca gelişmiş bir sorgu dili geliştirilmiştir. Cypher'ı öğrenmek ve güçlü bir şekilde kullanabilmek, çalışma alanını genişleterek özgürlük sağlayacaktır.

Neo4j'deki veri yapısı temel iki kavram üzerine oturtulmuştur; düğümler ve bağlantılar. Düğümler verileri, bağlantılar ise düğümler arasındaki ilişkileri gösterir. Herhangi bir düğümün veya bağlantının birden fazla özelliği olabilir.

Neo4j, nodes(kayıtlar), relationships(kayıtların ilişkileri) ve properties(node'ların özellikleri/içerdikleri veriler) gibi yapılardan oluşur. Node'lar ana kayıtlardır.

Node'lar içersinde isim, soy isim, tarih gibi özelliklere sahiptir. Labels(etiketler) ise node gruplarına verilen addır. Bir node hiç Label'a sahip olmayabilir veya birden fazla Label'a sahip olabilir. Label'lar herhangi bir özelliğe sahip olamaz. Sadece gruplama yapabilir. Graph Database'lerin kullanılmasındaki asıl güçlü unsur Node'lar arasındaki ilişkisel bağlantılardır[2].



Şekil 2: Genel olarak çizge veritabanlarının yapısı (medium.com' dan almtı yapılmıştır)

1. Nodes—Kayıtlar
2. Relationships—Kayıtların ilişkileri
3. Properties—Node'ların özellikleri/içerdikleri veriler

NODE

Node'lar ana kayıtlardır. Node'lar içersindeki özellikler name, from gibi… Properties olarak adlandırılır.Yani her Node, Properties'e sahiptir…

LABELS

Labels(etiketler) ise node gruplarına verilen addır. Örneğin name:"Emil" olan node aslında "Person(İnsan)" etiketiyle etiketlenmektedir. Person Label'ının rengi kırmızıdır.

Bir node hiç Label'a sahip olmayabilir veya birden fazla Label'a sahip olabilir. Label'lar herhangi bir özelliğe sahip olamaz. Sadece gruplama yapabilir.

Hız

Hız ile ilgili olarak birçok tartışma devam ediyor olsa da yapılan ölçümler bilgi vermeye yetecek düzeydedir. Neo4j ile bir sorgu yapldığında bu sorgu çok basit olsa bile 12 – 35ms aralıklarında sonuç döndürdüğü görülür. Oysa noSql ve RDBMS sistemlerde 1-2ms gibi süreler ölçmek mümkündür. Bu ilk başta can sıkabilir ancak büyük resme bakıldığında 3 veya 4 join attıldığında ve 200.000 civarı veri içeren mysql tablolarda ilişkili bir veriyi getirmek için harcanılan süre 200ms'nin üzerine çıkarken Neo4j'de 80ms civarında kalacaktır. Ayrıca veriyi indexleme işlemi cacheleme vs gibi işlemler Neo4j'de otomatik olarak yapılır (v2.0 ve sonrası)

Esneklik

Neo4j size noSql veritabanları kadar esnek bir yapı sunar. Yukarıda da bahsedilen gibi hem düğümlere (node) hem de bağlantılara (relation) istenildiği gibi veri girebilirsiniz. Herhangi bir şemaya bağımlı kalmadan yapı oluşturulabilir. Bir şemaya bağımlı kalmamak bazı yazılımcılarca hataya açık bir ortam oluşturmak gibi görünse de günümüz sistemleri artık internet ortamındaki sayısız veriyi taşımakta yetersiz kalmaktadır. Bir şemaya bağımlı kalmak da bu verilerin akışında ek işlemler ortaya çıkarıp akışı yavaşlatmaktadır. Ayrıca veri yapısında kısıtlamalara gidilmesine neden olmaktadır.

"Graph Database" felsefesinin çıkış tarihi çok eskidir fakat günlük kullanımda yeni yeni popüler olmaya başlamıştır. Bunun en büyük sebebi, artık RDBMS sistemlerin Solr, ElasticSearch gibi yan bir servis olmadan hatırı sayılır büyüklükte veriyi istenen sürelerde işleyememesidir. Her kullanılan servis de sunucuya ek bir maliyet getirir[13].

2.1.2. ÇİZGE BENZERLİĞİ METRİKLERİ

Çizge benzerliği, sosyal ağlar, görüntü işleme, ve bilgisayar görmesi gibi çeşitli uygulama alanlarına sahiptir ve bu nedenle birçok algoritma ve benzerlik ölçütü önerilmiştir. Önerilen teknikler üç ana kategoriye ayrılır: mesafenin düzenlenmesi/çizge izomorfizmi(edit distance/graph isomorphism), özellik çıkarımı (feature extraction) ve yinelemeli yöntemlerdir(iterative methods).

Edit distance/graph isomorphism:
Çizge benzerliğini değerlendirmede ki bir yaklaşım çizge izomorfizmidir. Eğer iki çizge izomorfikse ya da biri diğerinin alt-çizgesi (sub-graph) ile izomorfikse ya da bu iki çizge izomorfik alt-çizgelere sahipse onlar benzerdir.

Özellik çıkarımı(Feature extraction):
Bu yöntemlerin arkasındaki ana fikir, benzer çizgelerin muhtemelen derece dağılımı, çap, özdeğerler(eigenvalues) gibi belirli özellikleri paylaşmasıdır. Bu özelliklerin çıkarılmasından sonra, toplam istatistikler arasındaki benzerliği ve çizgeler arasındaki benzerliği değerlendirmek için bir benzerlik ölçüsü uygulanır.

Yinelemeli yöntemler(Iterative methods):
Yinelemeli yöntemlerin arkasındaki felsefe, “eğer iki düğümün komşuları benzerse o iki düğümde benzerdir”. Her yinelemede, düğümler benzerlik skorları değişiminde bulunurlar ve bu işlem yakınsama sağlandığında sona erer.

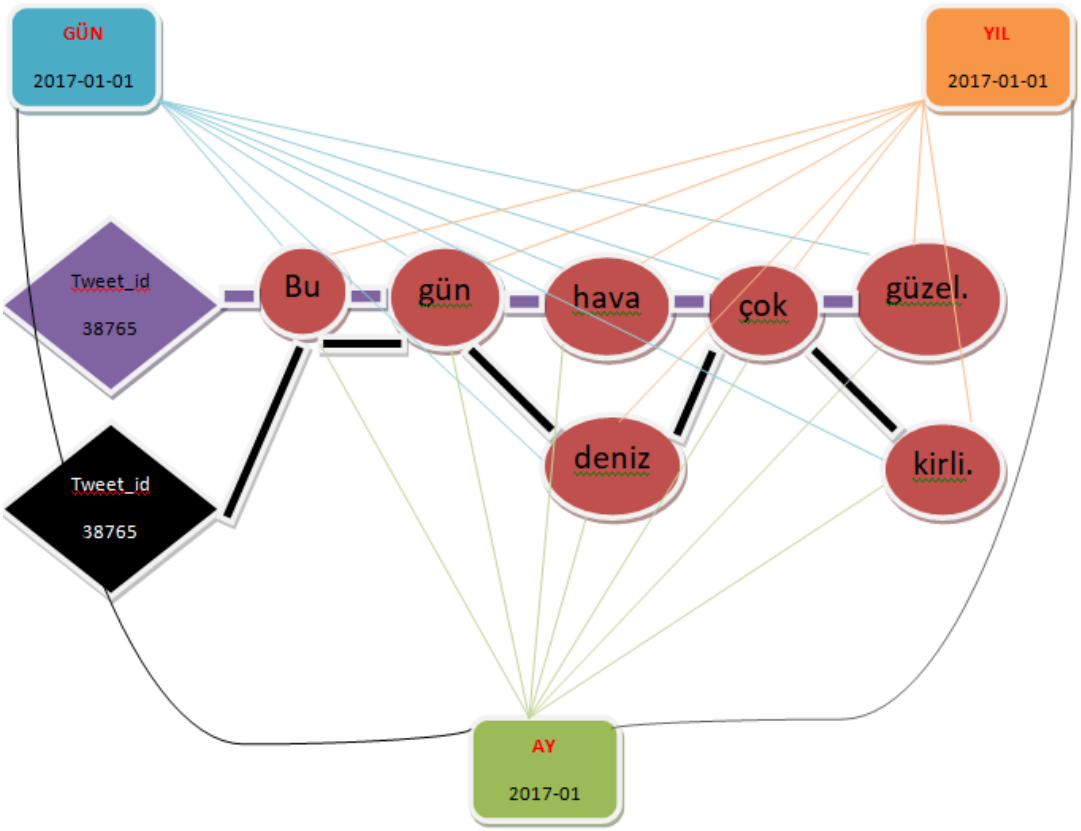
3. BULGULAR

Bu bölümde MySQL veritbanında yer alan Twitter bilgilerini Python programlama dili ile elde ederek, bu veriler ile Twitter çizgeleri oluşturulacaktır. Öncelikle elde ettiğimiz Twitter verilerini bir liste olarak oluşturduktan sonra "Tweet Class" yapısını oluşturarak Twitter listesindeki bilgileri Tweet Class' ının değişkenlererine atarak Tweet Classı listesi oluşturulmaktadır.Sonuç olarak MySQL veritabanında bulunan veriseti, Python programlama dili yardımıyla bir liste şeklinde elde edilmektedir. Bu veri kümesi üzerinde altı işlem gerçekleştirilmiştir.

1) Tweet Class' ı ile oluşturulmuş liste ile Python' ın "network" kütüphanesi kullanılarak çizge(graph) yapısı oluşturulması.
2) Tweet Class' ı içerisinde yer alan bilgileri çizge veritabanı olan "Neo4j Graph Database" ine kaydedilmesi.
3) Neo4j içerisinde yer alan çizgeleri Jupyter Notebook üzerinde görüntülenmesi. (graph visualization)
4) Neo4j veritabanı üzerinde sorgular oluşturulması.
5) Tweet Class' ının içerisinde yer alan "metadata" değişkeninde yer alan Tweet bilgileri ile Makine Öğrenmesi algoritmalarından "Doc2Vec" algoritması kullanılarak her Tweet verisinin vektörlerin çıkarılması ve "Doc2Vec" modelinin oluşturulması ve kaydedilmesi.
6) Kaydedilen "Doc2Vec" modelinin kullanılarak iki çizge yani Tweet verilerinin içeriklerinin karşılaştırılması ve bu karşılaştırmayı Makine Öğrenmesi algoritmalarından "Cosine Similarity" algoritması kullanılarak yapılması.

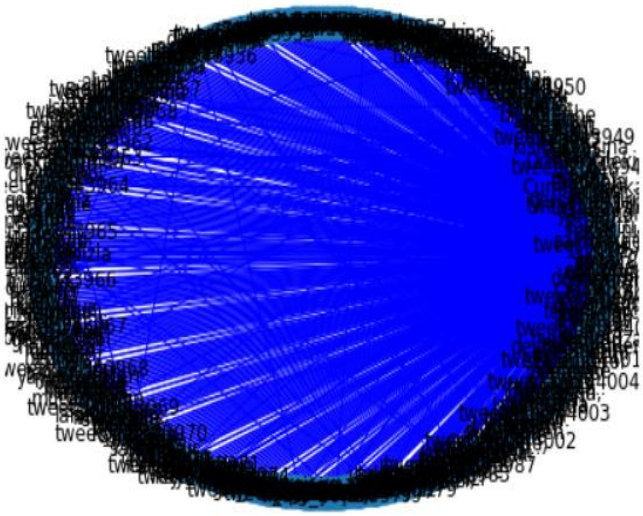
3.1. NETWORKX KÜTÜPHANESİ İLE ÇİZGE OLUŞTURMA

Tweet Class' ından oluşmuş liste "create_graph" fonksiyonuna parametre olarak verilir. Çizge modeli olarak belirlenmiş olan yapıda "node" ve "edge" ler oluşturulur. Benim kullandığım çizge modeli aşağıdaki gibi tasarlanmıştır.



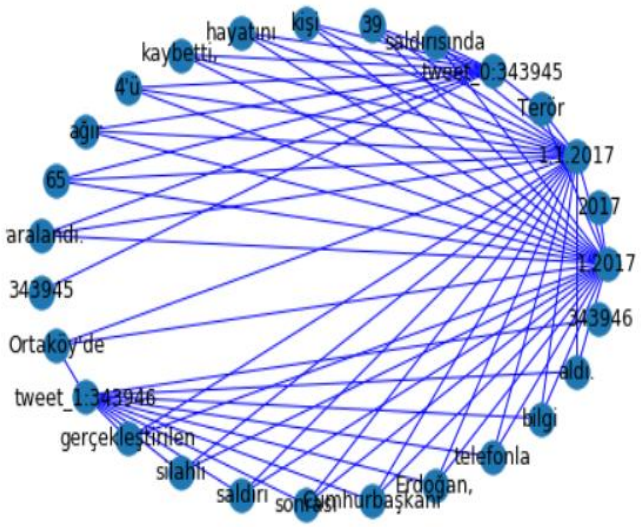
Şekil 3: Gün – Ay – Yıl' da yer alan Tweet' lerin çizge modeli

```
G = nx.Graph()
G = create_graph(tweets)
graph = nx.draw(G,with_labels=True,pos=nx.circular_layout(G),nodecolor='r', edge_color='b')
```



Şekil 4: Tweet' lerin networkx kütüphanesi ile gösterimi

```
G = nx.Graph()
G = create_graph(tweets[:2])
graph = nx.draw(G,with_labels=True,pos=nx.circular_layout(G),nodecolor='r', edge_color='b')
```



Şekil 5: İki Tweet' in networkx kütüphanesi ile gösterimi

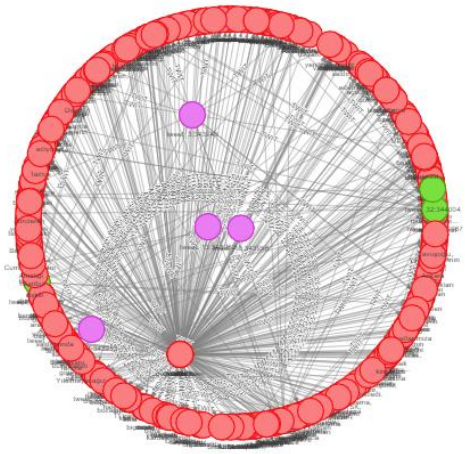
3.2. NEO4J ÇİZGE VERİTABANINA KAYIT İŞLEMİ

Neo4j çizge veritabanına kayıt işleminden önce "Neo4j Dekstop" veya "Neo4j Community" programını açmamız gerekmektedir. Açmamızın sebebi "Server – Client" ilişkisinden dolayıdır. Neo4j' yi açtıktan sonra Python programlama dilinde bulunan "py2neo" kütüphanesi yardımıyla Neo4j üzerinde bir çizge veritabanı oluşturabiliriz. Çizge veritabanı oluşturulduktan sonra içerisine "Label" lerden oluşmuş "Node" lar oluşturabiliz. Bu Node'ların "Property" leri bulunmaktadır. Bu "Property" alanında o Node hakkında bulunan özellikler yer almaktadır. Örneğin; Personal Label' ine sahip bir Node' un {name: Hakkı, surname: Bafralı, Age: 25} gibi özellikler bulunabilir.

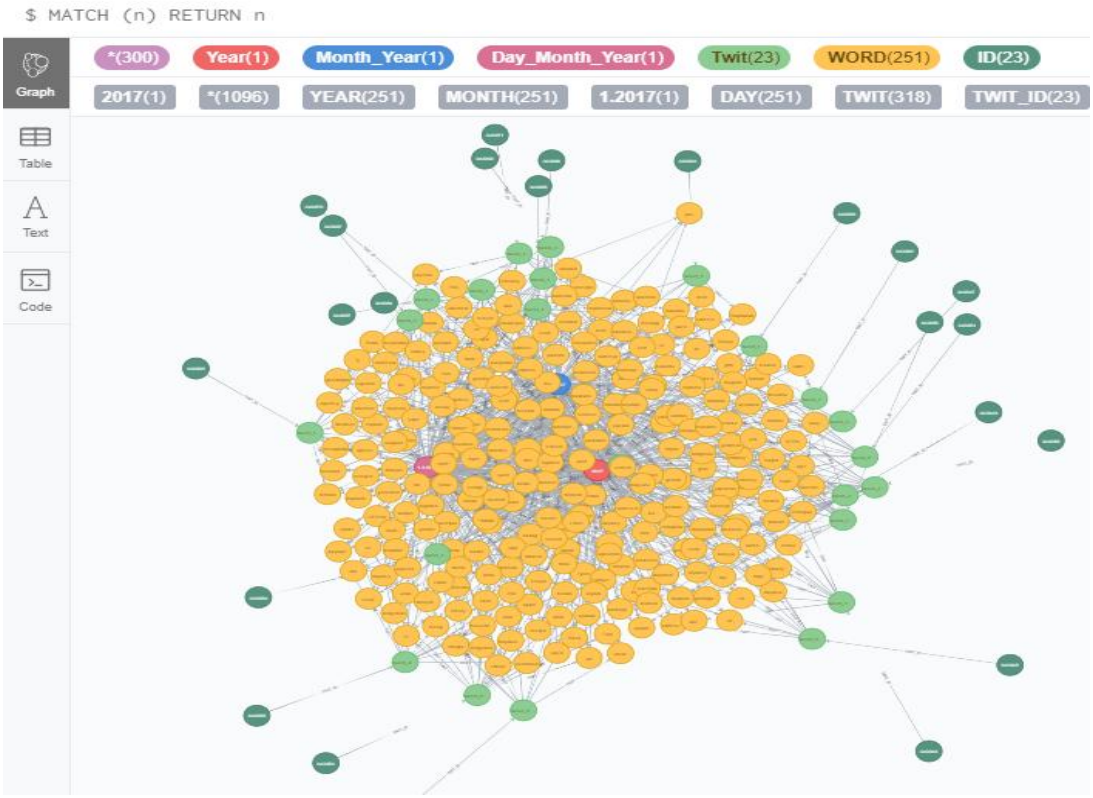
Ayrıca bu Node' ları birleştiren "Edge" yapıları vardır. Bunlar "Relation" lardır. Relation'lar iki Node arasında ne tür bir ilişki olduğunu belirtir.
Örneğin; "Hakkı" – KNOWS –-> "Mahmut".

Aşağıda MySQL veritabanından alınan Twitter bilgilerin Neo4j veritabanına ilişkili bir şekilde kaydedilmesi bulunmaktadır[4].

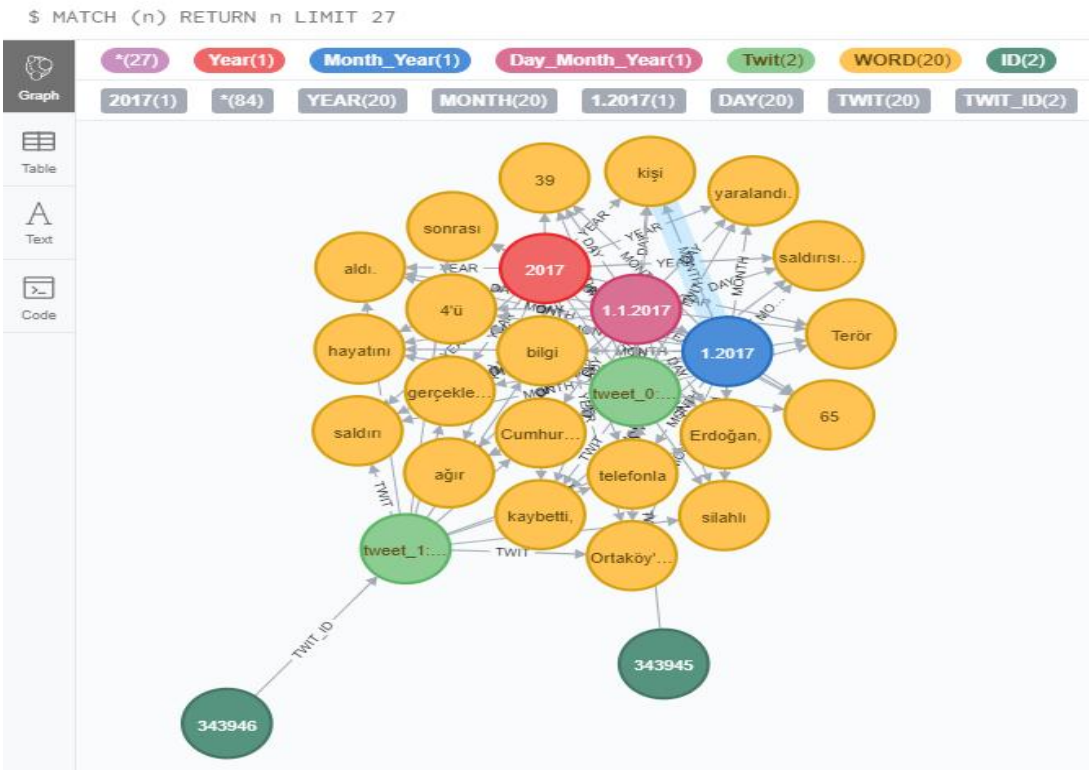
```
from scripts.vis import draw

options = {"Year": "name", "Month_Year": "name", "Day_Month_Year": "name", "Twit": "name", "WORD": "name"}
draw(graph, options)
```



Şekil 6: Neo4j veritabanında bulunan Tweet' lerin graph – visualization olarak Jupyter Notebook' ta gösterimi



Şekil 7: Neo4j veritabanında bulunan Tweet' lerin Neo4j Browser' da gösterimi



Şekil 8: Neo4j veritabanında bulunan iki Tweet' in Neo4j Browser' da gösterimi

3.3. NEO4J VERİTABANINDA SORGU İŞLEMİ CQL (CYPHER QUERY LANGUAGE)

Neo4j veritabanında bulunan veriler üzerine SQL soguları gibi sorgular yapmak mümkündür. Bunun için CQL (Cypher Query Language) kullanılır. Örneğin; hangi yılda hangi Tweet' lerin atıldığını bir tablo şeklinde görmek için Python programlama dilinde şu şekilde sorgular yazmamız yeterli olacaktır[7].

```
query = """
MATCH (year:Year)-[:YEAR]->(word:WORD)
RETURN word.name AS name, year.name AS year
"""

data = graph.run(query)
for d in data:
    print(d)
```

```
('name': '39', 'year': '2017')
('name': 'hesabı', 'year': '2017')
('name': 'uçaklar', 'year': '2017')
('name': 'Yıldırım', 'year': '2017')
('name': 'Bu', 'year': '2017')
('name': 'tutması', 'year': '2017')
('name': 'deprem', 'year': '2017')
```

Şekil 9: Cypher Query' lerin Python programlama dili ile yazılması sonucu sorgulama işlemi

```
%load_ext cypher
```

```
%%cypher
http://neo4j:123456@localhost:7474/db/data
MATCH (year:Year)-[:YEAR]->(word:WORD)
RETURN word.name AS name, year.name AS year
```

363 rows affected.

name	year
39	2017
hesabı	2017
uçaklar	2017
Yıldırım	2017
Bu	2017
tutması	2017
deprem	2017

Şekil 10: "%%cypher" keyword' u kullanılarak sorgulama işlemi

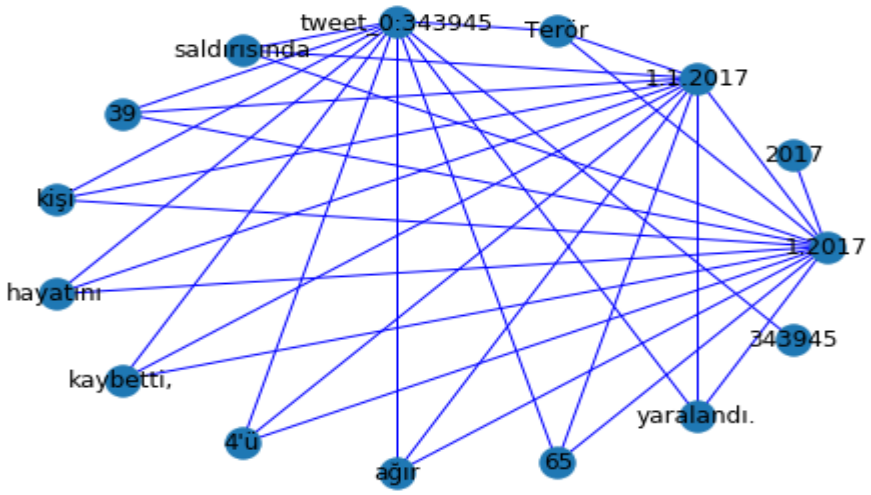
3.4. TWEET' LERİN DOC2VEC İLE VEKTÖRLERİNİN ÇIKARILMASI VE İKİ ÇİZGENİN KARŞILAŞTIRILMASI

Makine Öğrenmesinde yer alan "gensim" kütüphanesinin algoritmalarından "doc2vec" ile Tweet bilgilerini vektörlere çevirmemiz gerekmektedir. Çevrilen bu vektörleri daha sonra doc2vec modeli olarak makine öğrenmesine tabi tutmalıyız. Bu modelin daha sonralarda da kullanılabilir olması için modeli ".model" uzantısı

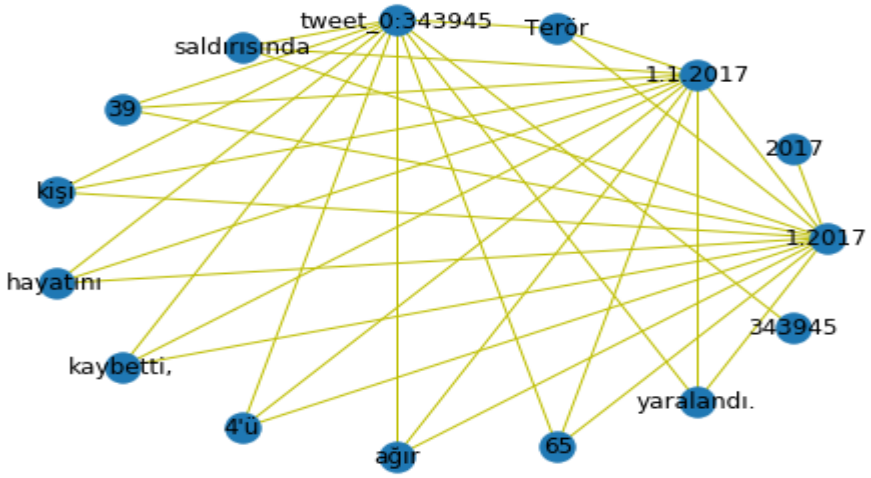
ile kaydetmemiz gerekmektedir. Vektörlere çevrilmiş ve eğitilmiş bu modeli "cosineSimilarity(modelFile, tweets, data1, data2)" fonksiyonuna vererek kullanabiliriz. Bu fonksiyon doc2vec modelini kullanarak her hangi iki Tweet arasında karşılaştırma yapmamıza benzerliklerinin sonucu hesaplamamıza yardımcı olur. Sonuç olarak bir "score" bilgisi döndürür ve iki Tweet arasında nasıl ne kadar bir benzerlik olduğu belirlenir.

Aşağıda iki Tweet arasında "graph_isomorphism, graph_edit_distance ve cosineSimilarity" olarak benzerlikler tespit edilmiştir.

Benzerlik Gösteren İki Çizge



Şekil 11: Tweet_Id: 343945 nolu 1. Tweet

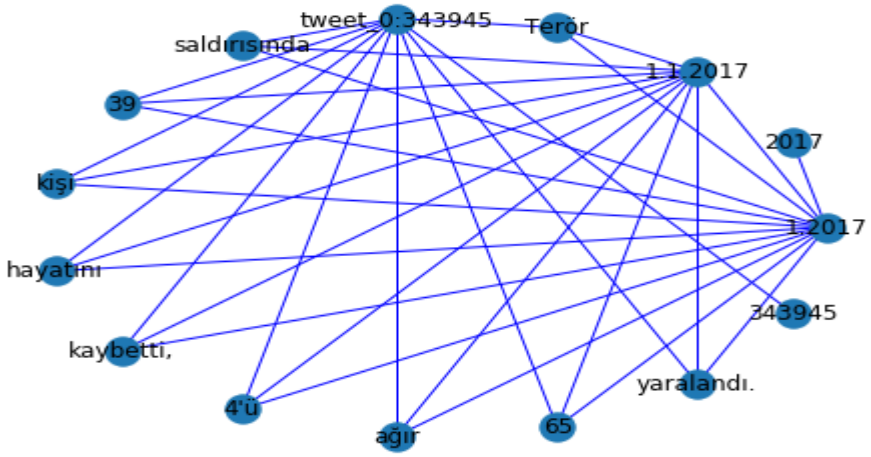


Şekil 12: Tweet_Id: 343945 nolu 2. Tweet

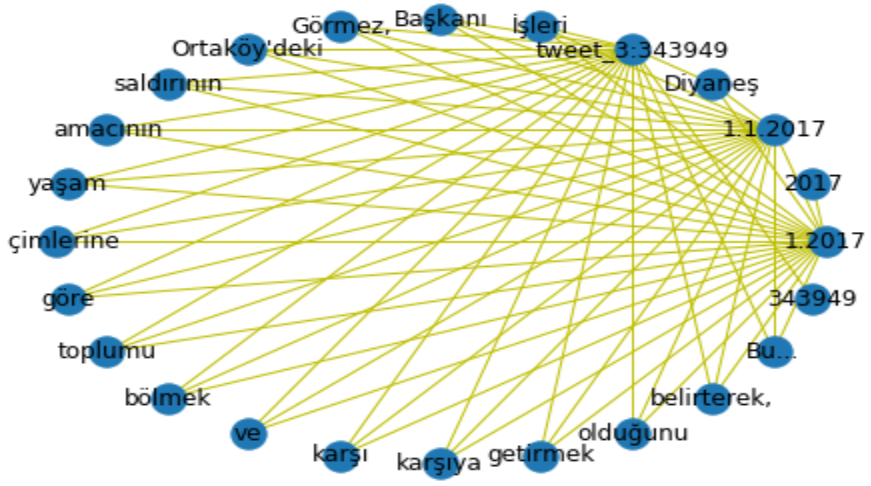
```
Cosine Similarity between two days =>    1.0
Cosine Similarity between two months =>  1.0
Cosine Similarity between two years =>   1.0
Cosine Similarity between two users =>   1.0000001
Cosine Similarity between two texts  =>  1.0

Weigthed Average Score of Cosine Similarities =>  1.000000023841858
-----------------------------------------------------------------------
Graph-isomorphism between two graphs =>  True
Graph-edit-distance between two graphs =>  0.0
```

Şekil 13: 1.Tweet ve 2. Tweet' in karşılaştırılma sonuçları

Benzerlik Göstermeyen İki Çizge



Şekil 14: Tweet_Id: 343945 nolu 1. Tweet



Şekil 15: Tweet_Id: 343949 nolu 2. Tweet

```
Cosine Similarity between two days =>    1.0
Cosine Similarity between two months =>  1.0
Cosine Similarity between two years =>   1.0
Cosine Similarity between two users =>   -0.13250376
Cosine Similarity between two texts  =>  0.5744927774241578

Weigthed Average Score of Cosine Similarities =>  0.6883978029165867
-----------------------------------------------------------------------
Graph-isomorphism between two graphs =>  False
Graph-edit-distance between two graphs =>  20.0
```

Şekil 16: 1.Tweet ve 2. Tweet' in karşılaştırılma sonuçları

GM.is_isomorphic: İki graph arasında Node ve Edge sayılarına bakarak içerik önemsiz bir şekilde bire bir karşılaştırma işlemi yapar. Eğer Node ve Edge sayıları birbirine eşit ise “True”, değilse “False” sonucunu döndürür[5].

Graph_edit_distance: İki çizge arasındaki Node uzaklıkları arasındaki mesafeyi karşılaştırır. Eğer iki çizge de Node ve Edge’ ler birbirine eşit ise ve mesafelerde eşit durumda ise sonuc olarak “0.0” döndürülür. Bu durum iki çizgenin bire bir benzer olduğunu göstermektedir. Eğer mesafeler farklı ve birbirine eşit Node ve Edge yapısı yoksa iki çizge arasındaki mesafelerin farkları döndürülür[6].

CosineSimilarity: İki Tweet çizgelerinin gün, ay, yıl ve user bilgileri Node2Vec algoritması ile vektörize edilir. Ayrıca Tweet bilgilerinin tutulduğu text kısmı ise Doc2Vec algoritması ile vektörize edilmektedir. Makine Öğrenmesi algoritmalarından “Cosine Similarity” iki kelime arasında benzerliği verirken, çizgeler için implement etmiş olduğum “myGraphCosineSimilarity” fonksiyonu ile iki Tweet çizgesinden elde edilen vektörler yardımı ile ayrı - ayrı Cosine Similarity’ leri alınmaktadır. Daha sonra gün, ay, yıl, user ve text bilgilerinin Cosine Similarity sonuçları “weightedAverageCalculation” fonksiyonu ile ağırlıklarına göre ortalamaları alınarak benzerlikleri hesaplanır.

4. TARTIŞMA VE SONUÇ

İnsanlar, sosyal medya yardımı ile birbirlerine onlarca hatta yüzlerce mesaj gönderebiliyor. Sosyal medya aracılığı ile yapılan mesajlaşmaların ve mesajlaşan kişilerin bilgileri belirli bir çizge(graph) yapısı içerisinde veritabanlarında bulundurulur. Sosyal Medya Verisinin Neo4j Veritabanına kaydedilmesi ve sorguların gerçekleştirilmesi ve Benzerlik Metriklerinin Geliştirilmesi projesi ile Twitter' daki mesajlaşmaların birbirlerine benzerlikleri bazı Makine Öğrenmesi algoritmalarına benzer algoritmalar tasarlanarak hesaplanmıştır. Bu algoritmalar çizgelerde bulunan gün, ay, yıl ve user node bilgilerinin Node2Vec algoritması ile vektörize edilmesi ile birlikte text bilgisinin Doc2Vec algoritması ile vektörize edilmesidir. Bu vektörlerin daha sonra ayrı – ayrı Cosine Similarity' leri alınmıştır. Tüm sonuçların ağırlıklı ortalamaları ile iki çizgenin ne kadar benzediği belirtilmiştir. Ayrıca mevcut sistemin kurulması aşamasında birçok problem ile karşılaşılmıştır. Bu problemlerden ilki Python programlama diline eklenen kütüphanelerden "py2neo" kütüphanesinin versiyon farklılığı olmuştur. Neo4j verilerinin Jupyter Notebook üzerinde gösterilmesi için kullanılan graph visualization' ın Web Browser' lardaki ayarlarının problemler oluşturması ise diğer bir sorundur. Ayrıca çizgelerin benzerliğini yüzde yüz ölçebilen bir algoritma ile araştırmalarım dahilinde karşılaşmadım. Şekil13 ve Şekil16' deki resimler bu durumu örnekler niteliktedir. Bu resimlerde Graph-isomorphism, bütün node' lar ve edge' ler metin içerikleri önemsiz olmakla birlikte birbirine eşit sayılarda ise "True", değil ise "False" değerini vermektedir. Aynı şekilde Graph-edit-distance, bütün node' lar ve edge' ler metin içerikleri önemsiz ve birbirini örtüyorsa aralarındaki fark "0.0", eğer örtme durumu bire bir değilse aralarındaki fark çıkan sonuca göre değişecek niteliktedir.

KAYNAKLAR

[1] http://cgnyazilim.com/blog/google-plus-twitterin-onemi/

[2] https://medium.com/5bayt/neo4j-nedir-e7160602211e

[3]https://www.ahmetiscan.web.tr/mysql-nedir-nerelerde-kullanilir-ozellikleri-nelerdir/

[4] https://nicolewhite.github.io/neo4j-jupyter/hello-world.html

[5]https://networkx.github.io/documentation/networkx-1.10/reference/algorithms.isomorphism.html

[6]https://networkx.github.io/documentation/latest/reference/algorithms/generated/networkx.algorithms.similarity.graph_edit_distance.html

[7] https://www.tutorialspoint.com/neo4j/neo4j_quick_guide.htm

[8]https://medium.com/@mishra.thedeepak/doc2vec-simple-implementation-example-df2afbbfbad5

[9] https://kanoki.org/2019/03/07/sentence-similarity-in-python-using-doc2vec/

[10] https://www.kaggle.com/currie32/predicting-similarity-tfidfvectorizer-doc2vec

[11] Giovanni Da San Martino, Nicolò Navarin, Alessandro Sperduti, **"Graph Kernels Exploiting Weisfeiler-Lehman Graph Isomorphism Test Extensions"**, Apr 2014

[12] Marc Wörlein,Thorsten Meinl, Ingrid Fischer, Michael Philippsen, **"A Quantitative Comparison of the Subgraph Miners MoFa, gSpan, FFSM, and Gaston"**, May 2005

[13] https://netvent.com/yepyeni-bir-veritabani-sistemi-neo4j/