T.C.

GALATASARAY ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

SAĞLIK VERİLERİNDE ÇİZGE TABANLI MODELLEME VE BÜYÜK DİL MODELLERİ İLE BİLGİ ÇIKARIMI

(GRAPH-BASED MODELING AND INFORMATION EXTRACTION WITH LARGE LANGUAGE MODELS ON HEALTH DATA)

DÖNEM PROJESİ

MUSTAFA DURMUŞ

Anabilim Dalı : MATEMATİK

Programı : VERİ BİLİMİ

Danışmanı : Dr. Öğr. Üyesi GÖNENÇ ONAY

ARALIK 2024

ÖNSÖZ

Bu çalışmanın hazırlanmasında, değerli bilgi birikimi ve rehberliğiyle bana yol gösteren değerli hocalarım Serap GÜRER ve Gönenç ONAY'a teşekkürlerimi sunarım.

Yüksek lisans eğitimim boyunca sağladığı hem maddi ve manevi destekle yanımda olan Albert Sağlık'a minnettarlığımı ifade etmek isterim.

Ayrıca, tüm süreç boyunca anlayışı ve desteğiyle yanımda olan, motivasyonumu ve çalışma azmimi her zaman yüksek tutmamı sağlayan değerli eşim Sedef SERT DURMUŞ'a ve aileme sonsuz teşekkür ederim.

Son olarak bu çalışmanın yapay zeka ve veri bilimi alanında bilgi birikimine katkı sağlamasını ve gelecekte yapılacak akademik çalışmalara ilham kaynağı olmasını temenni ederim.

MUSTAFA DURMUŞ

ARALIK 2024

İÇİNDEKİLER

KISALTMALAR

RAG	Retrieval Augmented Generation
NLP	Natural Language Processing
LLM	Large Language Models
RDF	Resource Description Framework
SQL	Search Query Language
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

SAĞLIK VERİLERİNDE ÇİZGE TABANLI MODELLEME VE BÜYÜK DİL MODELLERİ İLE BİLGİ ÇIKARIMI

Sağlık verileri, hastalar, doktorlar ve tedavi süreçleri gibi karmaşık ilişkiler içerir. Bu çalışmada, sağlık verilerinden anlamlı bilgi çıkarımı sağlamak için verilerin çizge tabanlı modellenmesi hedeflenmiştir. Bu amaç doğrultusunda, Neo4j ile sağlık verileri düğümler ve ilişkiler üzerinden modellenerek hastalar, doktorlar, teşhisler ve hastaneler arasındaki bağlantılar çizge yapıda temsil edilecektir. Bu yapı sayesinde sağlık verilerindeki karmaşık ilişkilerin daha anlaşılır ve analiz edilebilir hale gelmesi hedeflenmektedir.

Çizge modeli ile OpenAI'nin büyük dil modelleri (LLM) ve Retrieval-Augmented Generation (RAG) yöntemi kullanılarak sağlık verilerinde bilgi çıkarım sistemi geliştirilecektir. Bu sistem ile LLM'lerin doğal dilde kullanıcı sorularını anlamasına ve sağlık veritabanındaki güncel bilgileri kullanarak yanıt üretmesi sağlanacaktır. Elde edilen öncü sonuçlar sağlık verilerinin çizge tabanlı modelleme ve büyük dil modelleri ile entegre edildiğinde doğru ve kapsamlı analizlere imkan sağladığını göstermektedir.

Geliştirilen bu sistem, sağlık verilerinin analizinde kullanıcıların ihtiyaç duyduğu bilgiye daha doğru ve hızlı ulaşmasını sağlayan yenilikçi bir çözüm sunmaktadır.

ABSTRACT

GRAPH-BASED MODELING AND INFORMATION EXTRACTION WITH LARGE LANGUAGE MODELS ON HEALTH DATA

Health data includes multidimensional and complex relationships such as patients, doctors and treatment processes. In this study, it is aimed to use graph-based modeling in order to extract meaningful information from health data. For this purpose, health data is modeled with nodes and relationships using Neo4j, and the connections between patients, doctors, diagnoses and hospitals are represented in a graph structure. With this graph structure, it is aimed to make complex relationships in health data more understandable and analyzable.

In addition, a graph-based information extraction system has been developed in health data using OpenAI's large language models (LLM) and the Retrieval-Augmented Generation (RAG) method. With this system, LLMs are enabled to understand user questions in natural language and produce answers using up-to-date information in the health database. The results obtained show that health data enables accurate and comprehensive analysis when integrated with graph-based modeling and large language models.

This developed system offers an innovative solution that allows users to access the information they need more accurately and quickly in the analysis of health data.

BÖLÜM 1
GİRİŞ

1.1 Literatür Özeti

Çizge Veritabanlarının Sağlık Verilerinde Kullanımı

İlişkisel veritabanları gibi geleneksel yaklaşımlar sağlık gibi karmaşık ilişkiler içeren veri kümelerini saklamakta yetersiz kalabiliyor. Günümüzde bu ilişkileri verimli bir şekilde modellemek için çizge veritabanları kullanımı yaygınlaşmıştır. Neo4j gibi çizge veritabanları, bu karmaşık ilişkileri düğümler (örneğin, hasta veya doktor) ve bu düğümler arasındaki ilişkiler (örneğin, bir hastanın doktoruyla olan ilişkisi) olarak temsil eder. Bu sayede karmaşık ilişkiler çizgeler üzerinde daha anlaşılır halde kullanıma sunulmaktadır.

Doğal Dil İşleme (NLP) ve Büyük Dil Modelleri (LLM)

Doğal Dil İşleme (NLP) bilgisayarlara insan dilini öğretmeyi hedefleyen bir yapay zeka alt çalışma alanıdır. Büyük Dil Modelleri (LLM) NLP tekniklerini kullanarak insan dilini anlayıp üretim yapma yeteneğine sahip modellerdir ve sağlık verilerinde bilgi çıkarımı, teşhis önerileri gibi alanlarda kullanılır.

Örneğin, bir doktorun raporunda belirtilen semptomlara dayanarak hastalık tahmini yapma veya hasta geri bildirimlerinden hastane memnuniyetini ölçme gibi görevlerde NLP modelleri ile beraber LLM'ler yaygın olarak kullanılmaktadır. Özellikle BERT, GPT-4 gibi modeller, sağlık alanında hem yapılandırılmış hem de yapılandırılmamış verilerden bilgi çıkarımı yapma potansiyeline sahiptir.

Vektör Veritabanları

Vektör veri tabanları, metin verisini vektörler halinde saklayarak sonraki kullanımlar için hızlı erişim sağlar. Vektör veri tabanları, veriyi matematiksel vektörler olarak sakladığı için sorgulamalarda anlamsal benzerliklere göre en yakın sonuçları bulur. Bu teknoloji, LLM'ler ile kullanıldığında, dil modeli tarafından oluşturulan vektörlerin hızlı bir şekilde taranması ve en uygun sonuçların döndürülmesi ile kullanım kolaylığı sağlar.

RAG (Retrieval-Augmented Generation)

RAG, LLM ve bilgi erişim yöntemlerini birleştirerek doğru ve veriye dayalı cevaplar sunabilen bir tekniktir. RAG, bir veritabanı veya veri kaynağından alınmış spesifik bilgi parçacıklarını kullanarak LLM'in cevaplarını desteklemeyi sağlar. Bu sayede sadece modelin öğrenmiş olduğu bilgilerle değil, güncel veya projeye özel verilerle de cevaplar üretilir. Birçok LLM üzerinde görülen halüsinasyon problemine çözüm olarak sıkça kullanılmaktadır.

1.2 Projenin Amacı

Bu projenin amacı, sağlık verilerinden anlamlı bilgi çıkarımı yapabilen bir sistem geliştirmektir. Bunun için, veriler önce çizge tabanlı bir yapı olan Neo4j ile modellenerek ilişkisel analizler yapılacaktır. Ardından, LLM ve RAG yöntemleri kullanılarak, veriye dayalı soru-cevap sistemi oluşturulacaktır.

BÖLÜM 2
ÇİZGE VERİTABANLARI

2.1 Çizge Veritabanı

Çizge veritabanları geleneksel veritabanlarının tabloda veya belge üzerinde sakladığı verileri düğümler ve ilişkiler aracılığıyla bir çizge üzerinde saklar [1]. Düğümler varlıkları (örneğin, kişiler ve objeler) temsil ederken, ilişkiler bu düğümler arasındaki bağlantıları belirtir.

Günümüzde çizge tabanlı modellemenin kullanımının artmasının sebebi veriler arasında doğrudan bağlantıların kurulmasını sağlayarak hızlı ve doğru sorgulama yapabilmesidir. Düğümler ve ilişkiler aracılığıyla veri üzerinde bağlantılar kurulur ve bu sayede en karmaşık veri yapılarında bile ilişkiler görselleştirilir ve analiz edilmesi kolaylaşır. Ayrıca çizgeler esnek yapıya sahiptir; yeni düğüm veya ilişkiler eklendiğinde model kolayca güncellenebilmekte ve yeni veri ihtiyaçlarına uyum sağlanabilmektedir.

Çizge veritabanlarının farklı modelleri bulunmaktadır. Bunlardan bazıları aşağıdaki gibidir [1].

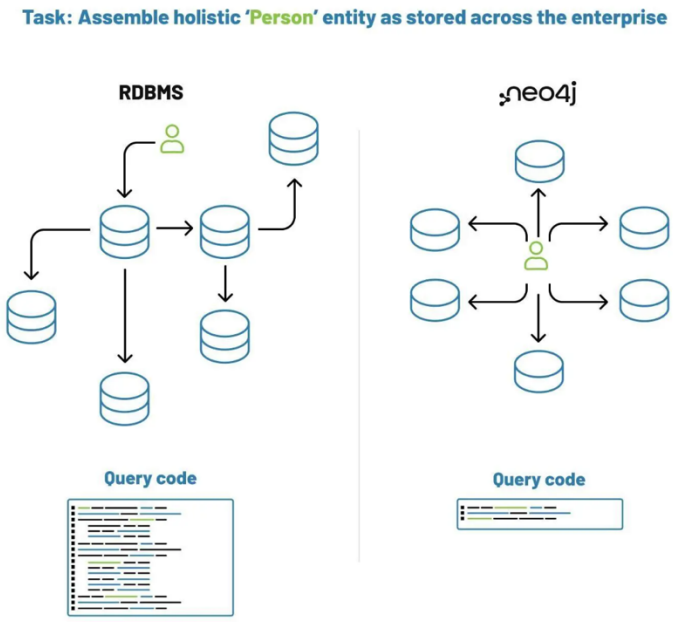
- **Varlık-İlişki (Property Graph) Modeli**: Her düğüm ve ilişkiye farklı özellikler atanır. Bu özellikler, her varlık veya ilişki hakkında daha fazla bilgi saklamayı sağlar. Özellikle sosyal ağ analizi, finansal veri analitiği gibi karmaşık ilişki içeren uygulamalarda kullanılır.

- **RDF (Resource Description Framework)**: Semantik web ile kullanılan bu modelde düğümler ve ilişkileri özne-özellik-nesne formatında saklar. Özellikle ontolojiler ve bilgi yönetim sistemlerinde tercih edilir.

Proje kapsamında **varlık-ilişki** çizge modeli kullanılacaktır.

2.2 Neo4j

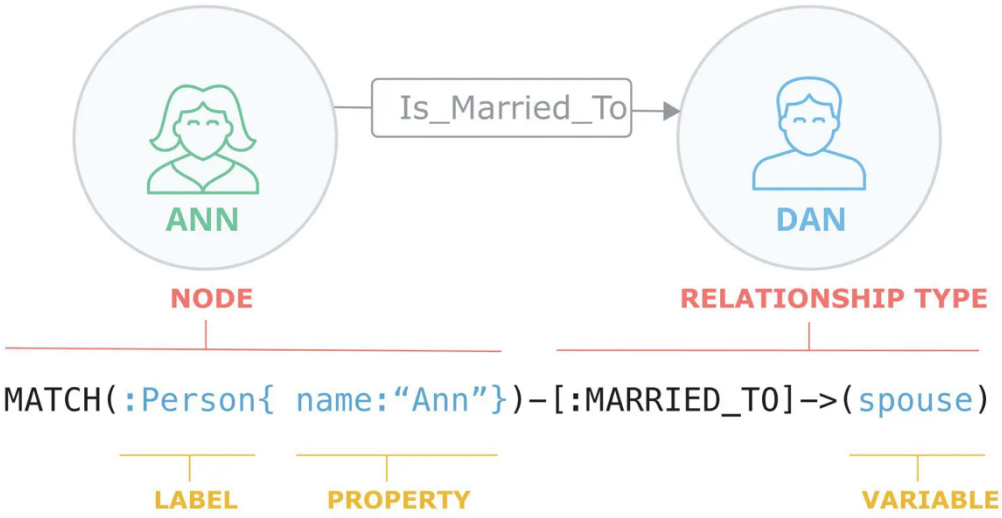
Neo4j, 2007 yılında geliştirilmiş ve günümüzde oldukça popüler bir çizge veritabanı olup veri yapısını düğümler ve ilişkiler üzerinden kurar [2]. Cypher adında bir sorgulama dili kullanarak veriler arasındaki ilişkileri tanımlamaya ve analiz etmeye olanak sağlar.



Şekil 2.1 Neo4j ile geleneksel veritabanlarının karşılaştırılması

Neo4j'nin en belirgin özelliklerinden biri, çizge verileri üzerinde işlem yapmayı kolaylaştıran Cypher sorgu dilidir. SQL gibi ilişki tabanlı dillerden farklı olarak Cypher çizge yapısına uygun bir syntax sunar. Neo4j doğrudan bağlantılar üzerinden veri sorgulaması yapabildiği için ilişkisel veritabanlarına kıyasla daha kısa sorgular kullanarak çok daha kısa sürelerde sonuç döndürür. Bunun yanı sıra, yeni düğüm ve ilişkilerin eklenmesiyle veri modelini dinamik bir şekilde genişletme imkanı sağlar. Bu esneklik ile değişen veri gereksinimlerine hızlıca uyum sağlanmasını mümkün kılmaktadır.

Aşağıda örnek 2 kişi düğümü ve aralarındaki ilişki ile cypher sorgusu gösterilmiştir.



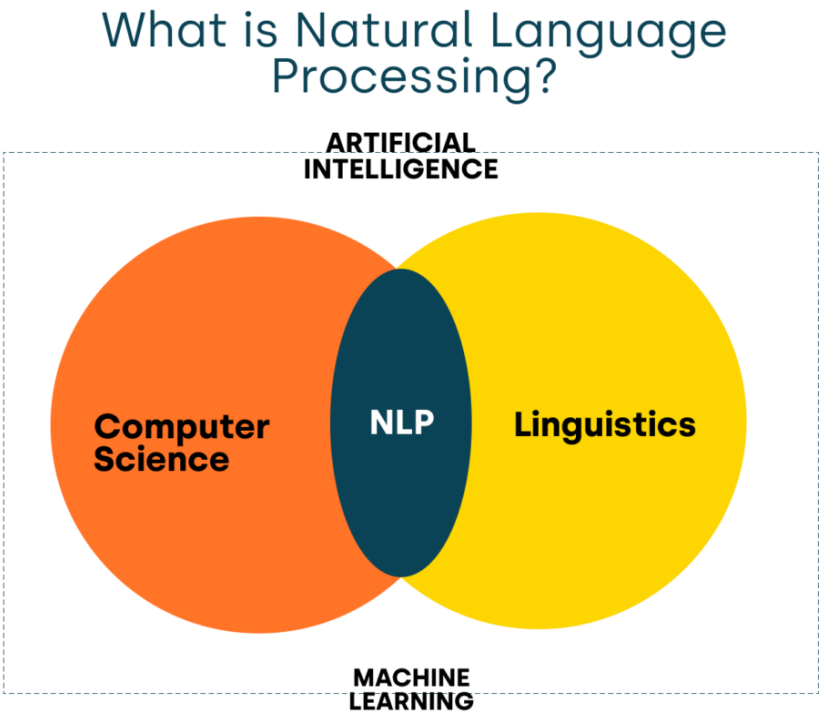
Şekil 2.2 Neo4j genel akışı[2]

BÖLÜM 3

BÜYÜK DİL MODELLERİ ve RAG

3.1 Doğal Dil İşleme

Doğal Dil İşleme (NLP), bilgisayarların insan dilini anlaması, işlemesi ve analiz etmesi için geliştirilmiş bir yapay zeka alt çalışma alanıdır. İnsanlar tarafından üretilen metin verilerini analiz ederek dilin yapısını öğrenir. NLP ile metin sınıflandırma, duygu analizi, makine çevirisi, soru-cevap sistemleri ve sohbet botları gibi çeşitli uygulamalarda kullanılır. Şekil 3.1 üzerinde görüldüğü gibi bilgisayar bilimleri ve dilbilim konularının ortak kümesi olarak tanımlanabilir.



Şekil 3.1 Doğal Dil İşleme

3.2 Transformer

Transformers, büyük dil modellerinin en yaygın kullanılan mimarisidir. Bir kodlayıcı (encoder) ile bir kod çözücüden (decoder) oluşur ve girdi verilerini işleyerek belirteçler arasındaki ilişkileri keşfetmek için aynı anda matematiksel denklemler çalıştırır [3].

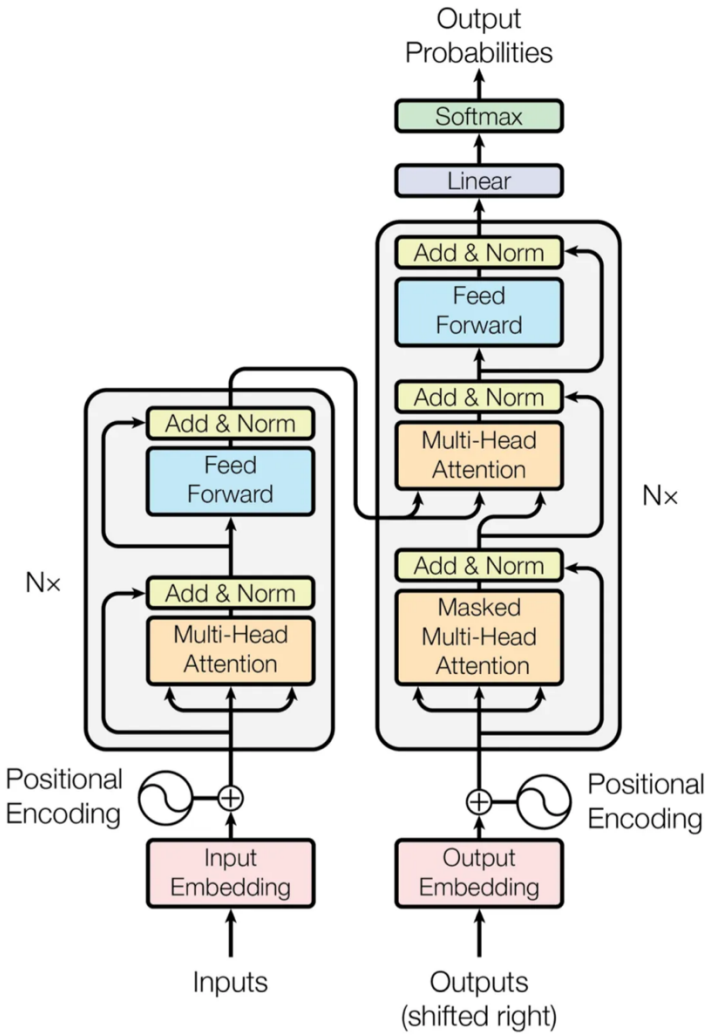


Figure 1: The Transformer - model architecture.

Şekil 3.2 Transformer Mimarisi [3]

Şekil 3.2'de görüldüğü gibi bir Transformer modeli, girdi metni aldıktan sonra her kelimeyi vektör formatına çevirir. Ardından, her bir kelimenin diğer kelimelerle olan ilişkisini öğrenmek için dikkat mekanizmasını (attention mechanism) kullanır. Attention ile her bir kelimenin cümlenin geri kalan kelimelerine ne kadar bağlı olduğunu belirlenir. Böylece model sadece kelimelerin sıralarına değil de her kelimenin bağlamına da odaklanarak metni derinlemesine anlamaya çalışır. Encoder kısmında metindeki tüm kelimelerin birbirleriyle olan ilişkisi analiz edilir ve bu ilişkiler bir vektöre çevrilir. Decoder kısmında ise bu vektör kullanılarak modelin anlamlı bir yanıt üretmesini sağlanır.

3.3 Büyük Dil Modeli

Büyük dil modelleri (LLMs - Large Language Models), transformer mimarisini temel

alarak metinleri anlamak, işlemek ve yeni metinler üretmek için büyük veriler üzerinde eğitilmiş modellerdir. Transformer'ın sağladığı attention mekanizması sayesinde LLM'ler büyük miktarda metin verisinde kelimeler arasındaki karmaşık ilişkileri öğrenir. LLM'ler genellikle büyük veri kümeleri üzerinde eğitilir ve milyarlarca parametreye sahip olabilirler.

Transformer mimarisine dayalı olan LLM'ler mimarinin yalnızca decoder tarafını kullanarak çalışır. Decoder kısmı, belirli bir metni tamamlamak veya yeni bir metin üretmek için uygundur çünkü her bir kelimenin önceki kelimelerle olan ilişkisini dikkate alarak anlamlı bir çıktı oluşturur. Decoder, dikkat mekanizmasıyla metindeki her kelimenin kendisinden önceki kelimelerle olan yakınlığını tespit edebilir ve akışa uygun şekilde kelime tahmini yapabilir.

3.4 Vektör Veritabanı

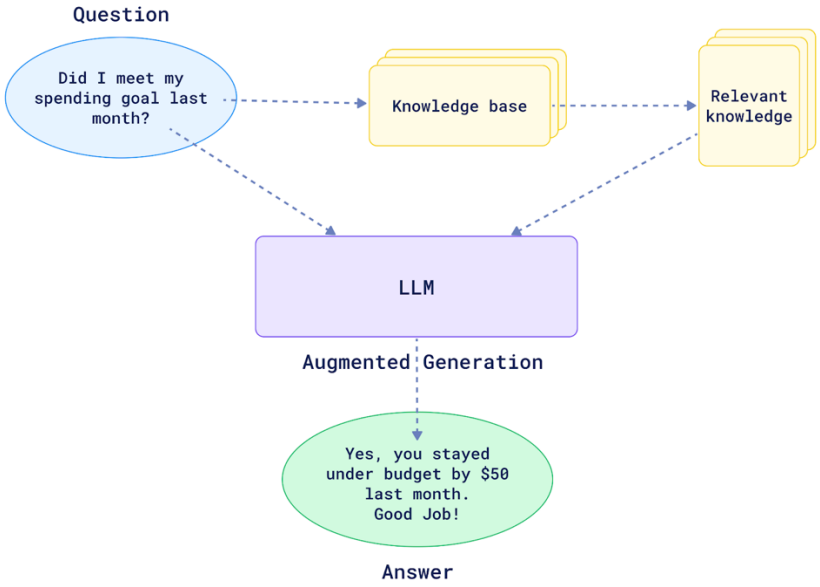
Vektör veritabanları, metin verilerinin vektör karşılıklarını saklayıp bu vektörler arasında hızlı ve anlamlı arama yapmayı sağlar. Özellikle LLM kullanılan uygulamalarda verilerden anlamlı bilgiler çıkarmada kullanılır [4].

Metinden elde edilen her veri parçası, model tarafından vektör formunda temsil edilir ve bu vektörler veritabanına kaydedilir. Vektör veritabanları, sorgulama sırasında bir kullanıcı girdisini vektöre dönüştürüp benzer anlamdaki diğer vektörlerle eşleştirerek en alakalı sonuçları yani vektörleri geriye döndürür. Bu eşleştirme işlemi sırasında genellikle öklid mesafesi, kosinüs benzerliği veya iç çarpım gibi metrikler kullanılmaktadır.

3.5 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG), büyük dil modellerini vektör veritabanında saklanan verilerle destekleyerek daha doğru ve güvenilir yanıtlar üreten bir tekniktir. RAG, kullanıcının girdi sorusuna en yakınsadığı bilgiyi LLM'lere sağlayarak, LLM'lerin veriye dayalı daha spesifik cevaplar üretmesine olanak tanır. RAG basit olarak 3 aşamadan oluştur.

- **Vektör Haline Çevirim** Kullanıcının sorusu vektörleştirilir.
- **Bilgi Çağırma (Retrieval)** Vektör veritabanında yer alan en alakalı bilgiler alınır.
- **Dil Modeliyle Yanıt Üretimi** Alınan bilgiler ve kullanıcı sorusu dil modeline aktarılır ve kullanıcı sorusuna uygun yanıt üretilir.



Şekil 3.3 RAG mimarisi [5]

BÖLÜM 4
VERİ

4.1 Veri Setinin Tanımı ve Yapısı

Proje kapsamında kullanılan sağlık verisi kaggle web sitesi üzerinden alınmıştır [6]. Veri toplam 49992 satır kayıt içermektedir. Bu veri seti, hastaların teşhis ve tedavi bilgilerini, hastane yatış tarihlerini ve doktorlarla olan etkileşimleri gibi birçok ilişkiyi kapsayan bilgileri içermektedir.

4.2 Veri Tipleri ve Özellikleri

Veri setindeki temel veri tipleri ve içerdiği bilgiler şunlardır:

1. **İsim**: Ad ve soyad bilgisini içerir.
2. **Yaş**: Yaş bilgisini içerir.
3. **Cinsiyet**: Cinsiyet bilgisini içerir.
4. **Kan Grubu**
5. **Teşhis**: Sağlık durum bilgisini içerir.
6. **Hastane**: Hastanın başvurduğu hastane.
7. **Hastane Başvuru Tarihi**
8. **Doktor**: Hastaya bakan doktorun ad-soyadı.
9. **Sağlık Sigortası**: Hastanın sigorta bilgilerini.
10. **Fatura**: Hastaneye ödenen fatura.
11. **Oda Numarası**: Hastanede kalınan oda numarası.
12. **Hastane Ayrılış Tarihi**: Hastaneden ayrılma tarihi.
13. **İlaç**: Hastanın kullandığı ilaç bilgisini içerir.
14. **Test** Sonucu: Hastaya uygulanan testin sonuç bilgisini içerir.

4.3 Veri Ön İşleme ve Temizleme

Bu adımda, Neo4j üzerinde gerçekleştirilecek analiz için veriden bir alt küme çıkarılması hedeflenmiştir. Çizge veritabanında anlamlı olması için tekil kayıtlar olabildiği kadar veriden temizlenmiştir.

- **Minimum 5 Farklı Hastaya Hizmet Eden Hastaneler**: Veri setinde en az 5 farklı hastaya hizmet vermiş hastaneler kullanılmıştır. Bu filtre, sağlık hizmetlerinin yeterince kapsayıcı olup olmadığını ve hastanenin çeşitli hastalık ve hasta profilleriyle ilgili bilgi sağlaması hedeflenmiştir.

- **Minimum 2 Hastaya Bakım Yapmış Doktorlar**: Veri setinde en az iki hastaya bakmış doktorlar kullanılmıştır. Bu filtre ile doktorların hastalık ve hasta çeşitliliği konusunda yeterli bilgi sağlaması hedeflenmiştir.

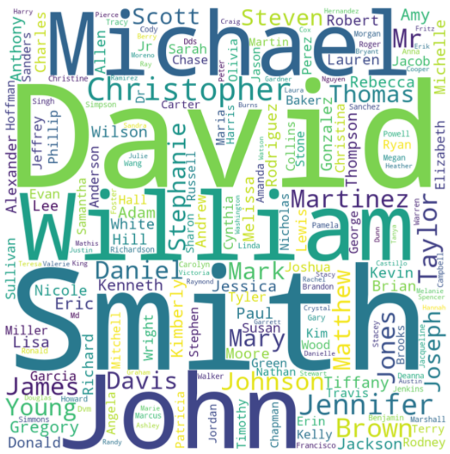
Yukarıdaki filtreler kullanılarak elde edilen veriseti toplamda 950 kayıtlık bir alt küme

elde edilmiştir. Sonraki bölümlerde yapılan veri analizleri ve Neo4j üzerinde veri modellemesi bu alt küme üzerinden yapılmıştır.

4.4 Veri Analizi

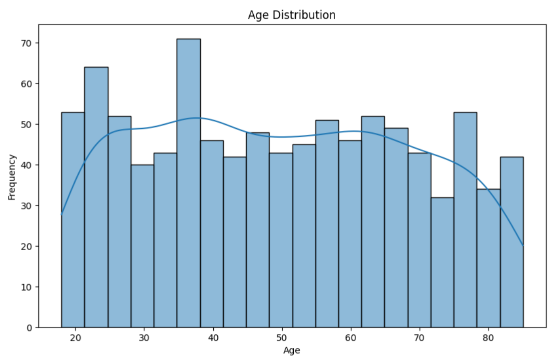
Veri üzerinde yapılan tanımlayıcı analizler aşağıda görülmektedir.

1. **İsim**



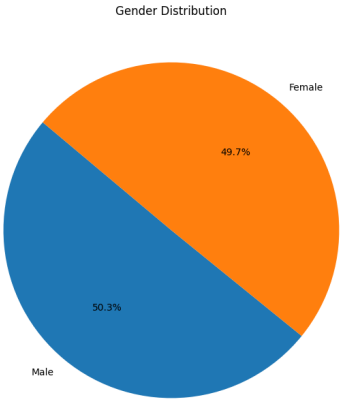
Şekil 4.1 Verideki isimlerin wordcloud örneği.

2. **Yaş**



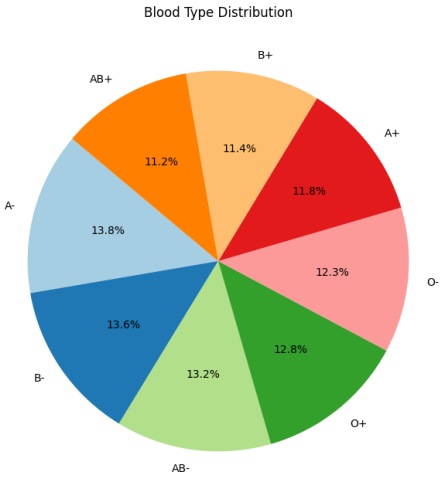
Şekil 4.2 Yaş dağılımı

3. **Cinsiyet**



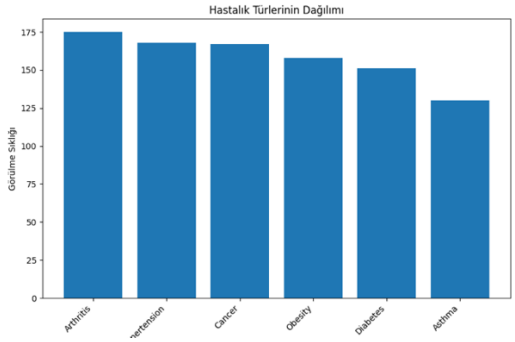
Şekil 4.3 Cinsiyet dağılımı

4. Kan Grubu



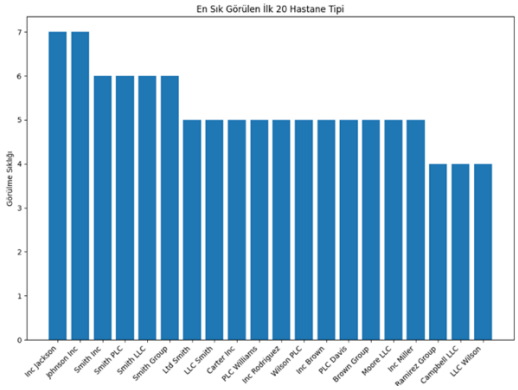
Şekil 4.4 Kan grubu dağılımı

5. Teşhis



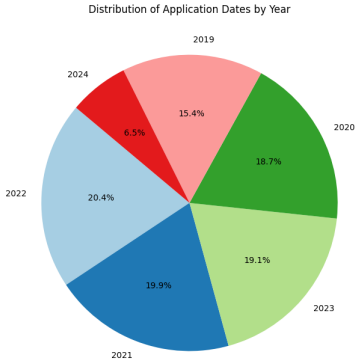
Şekil 4.5 Hastalık türlerinin dağılımı

6. Hastane



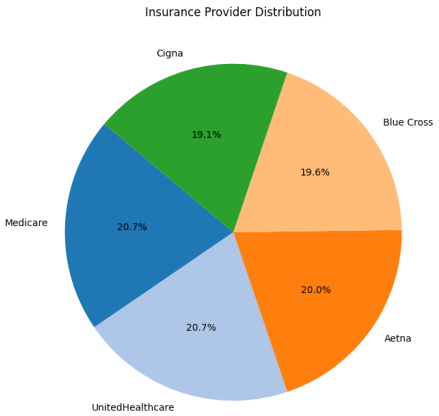
Şekil 4.6 En sık görülen 20 hastane dağılımı

7. Hastane Başvuru Tarihi



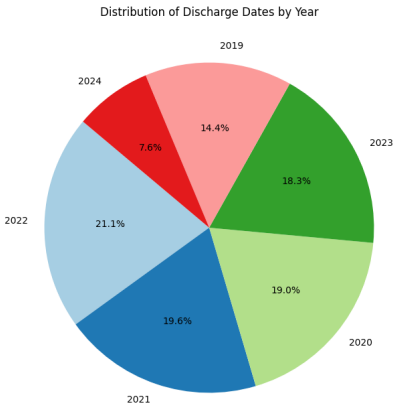
Şekil 4.7 Hastane yatış tarihlerinin yıllara göre dağılımı

8. Sağlık Sigortası Firması



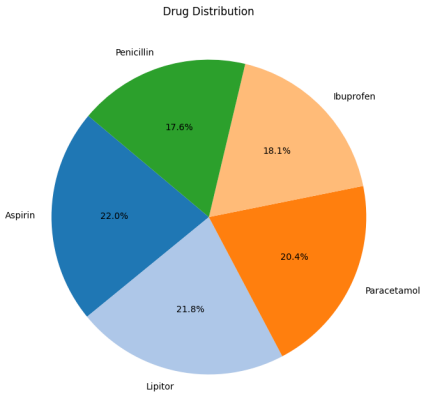
Şekil 4.8 Sigorta şirketlerinin dağılımı

9. Hastane Ayrılış Tarihi



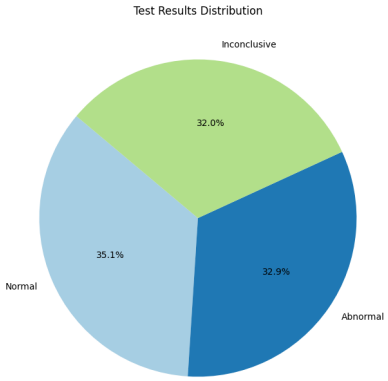
Şekil 4.9 Hastanelerden ayrılma tarihlerinin yıllara göre dağılımı

10. İlaç



Şekil 4.10 İlaç dağılımı

11. Test Sonucu



Şekil 4.11 Test sonuç dağılımı

BÖLÜM 5

NEO4J ile VERİ MODELLENMESİ

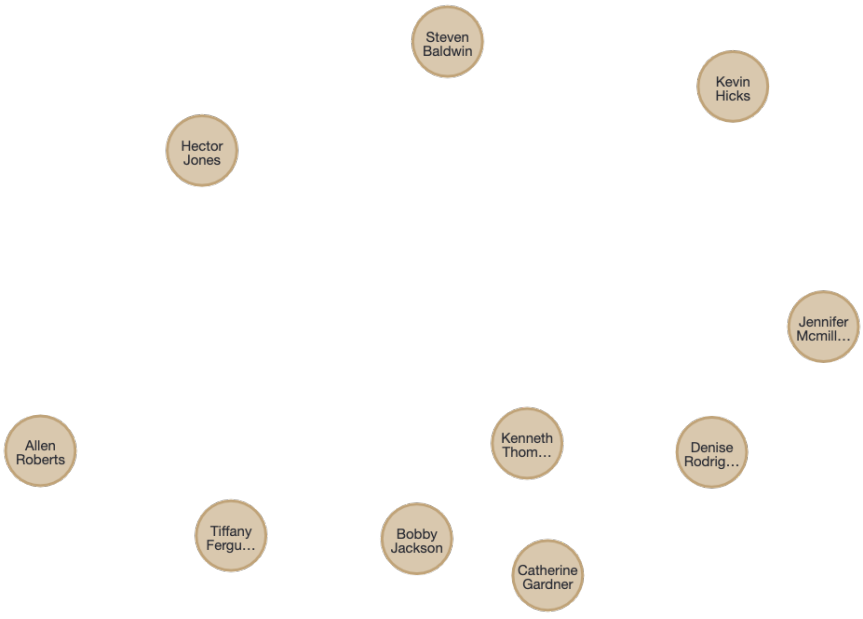
Bu adımda, veriler data.csv dosyası ile Neo4j'ye yüklenecektir.

5.1 Düğümler

Aşağıda tanımlanan düğümler, Neo4j üzerinde çalıştırılan Cypher kodları ve çıktıları görülmektedir.

- **Patient Düğümleri**

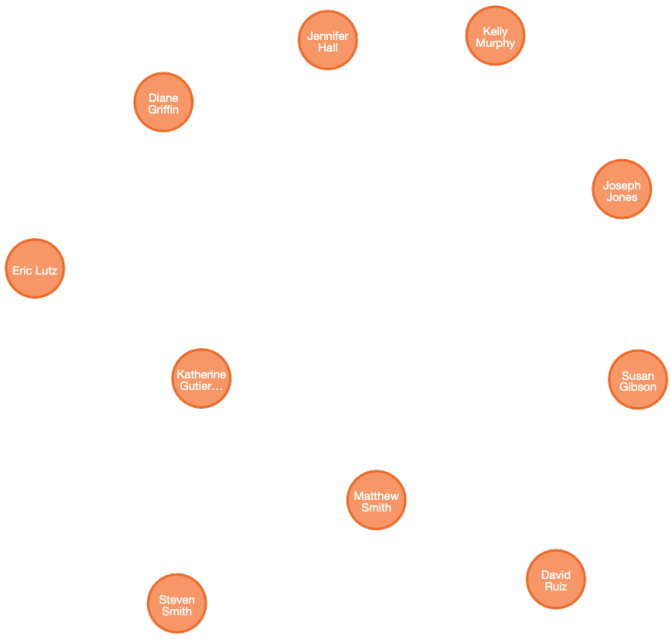
```
LOAD CSV WITH HEADERS FROM 'file:///data.csv' AS row
MERGE (p:Patient {name: row.Name})
SET p.age = toInteger(row.Age),
      p.gender = row.Gender,
      p.bloodType = row.Blood_Type;
```



Şekil 5.1 Hasta düğüm örnekleri

- **Doctor Düğümleri**

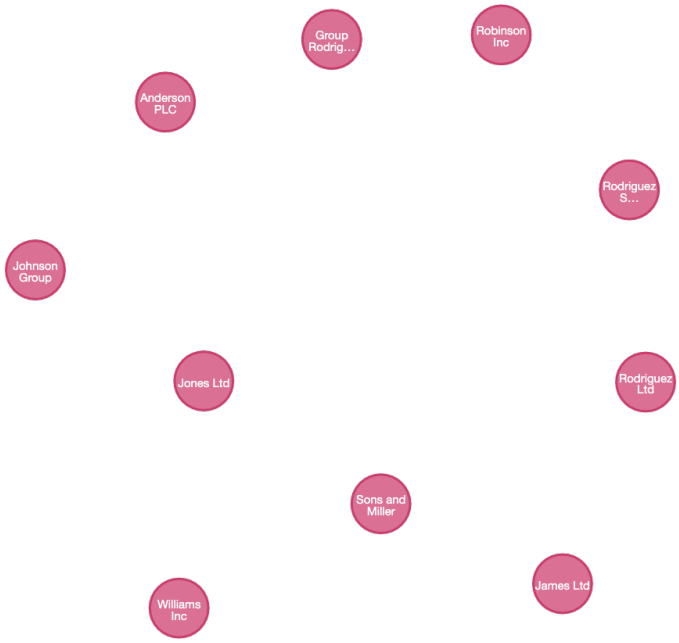
```
LOAD CSV WITH HEADERS FROM 'file:///data.csv' AS row
MERGE (d:Doctor {name: row.Doctor});
```



Şekil 5.2 Doktor düğüm örnekleri

- **Hospital Düğümleri**

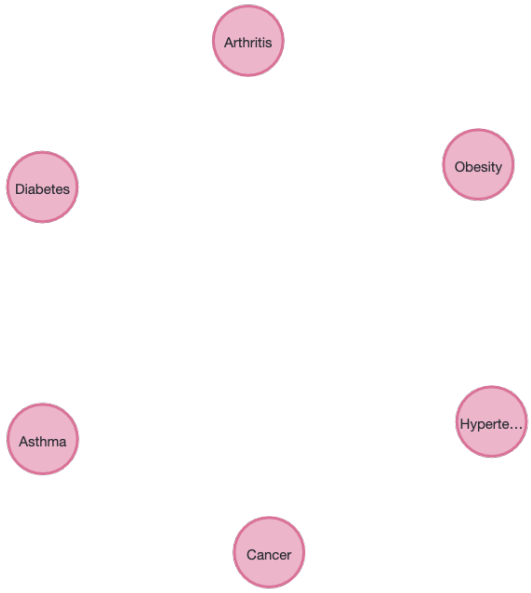
```
LOAD CSV WITH HEADERS FROM 'file:///data.csv' AS row
MERGE (h:Hospital {name: row.Hospital});
```



Şekil 5.3 Hastane düğüm örnekleri

- **Disease Düğümleri**

```
LOAD CSV WITH HEADERS FROM 'file:///data.csv' AS row
MERGE (disease:Disease {name: row.`Medical Condition`});
```



Şekil 5.4 Teşhis düğüm örnekleri

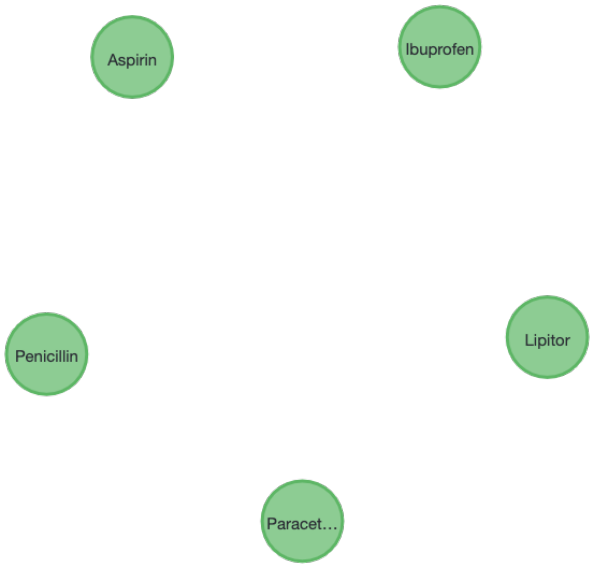
- **Insurance Provider Düğümleri**

```
LOAD CSV WITH HEADERS FROM 'file:///data.csv' AS row
MERGE (i:InsuranceProvider {name: row.`Insurance Provider`});
```

Şekil 5.5 Sigorta şirketi düğüm örnekleri

- **Medication Düğümleri**

```
LOAD CSV WITH HEADERS FROM 'file:///data.csv' AS row
MERGE (m:Medication {name: row.Medication});
```



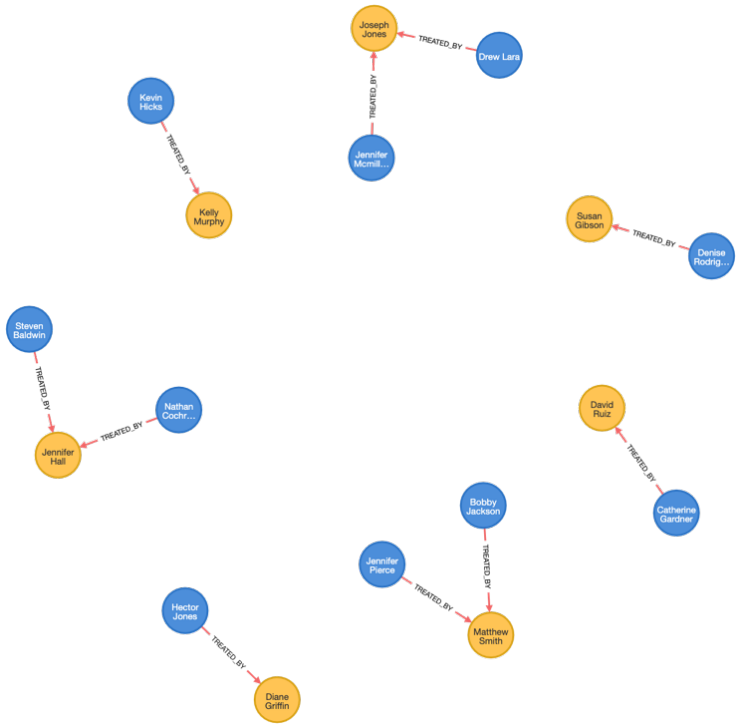
Şekil 5.6 İlaç düğüm örnekleri

5.2 İlişkiler

CSV dosyasındaki verileri kullanarak, düğümler arasındaki ilişkiler aşağıda gibi modellendi.

- **Patient-Doctor İlişkisi**

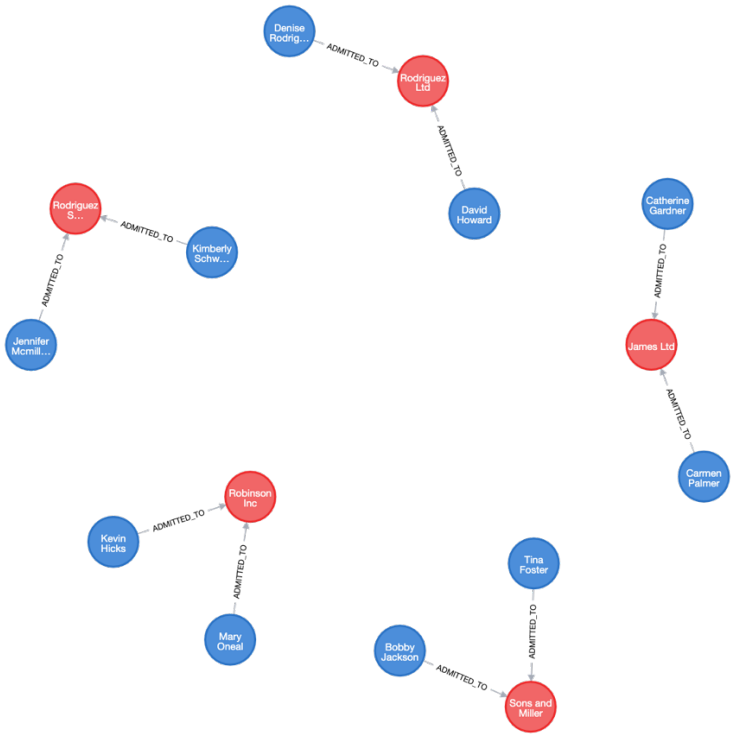
```
LOAD CSV WITH HEADERS FROM 'file:///data.csv' AS row
MATCH (p:Patient {name: row.Name})
MATCH (d:Doctor {name: row.Doctor})
MERGE (p)-[:TREATED_BY]->(d);
```



Şekil 5.7 Hasta-doktor modellenmesi

- **Patient-Hospital İlişkisi**

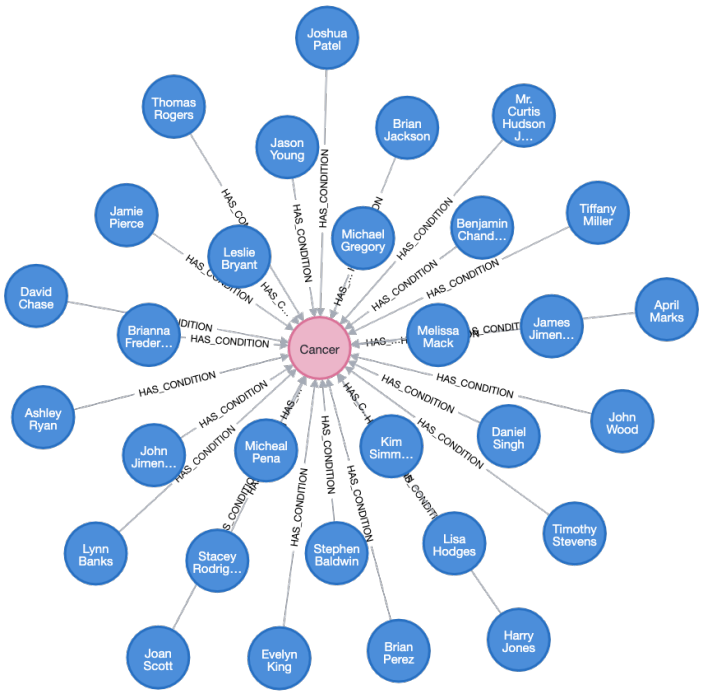
```
LOAD CSV WITH HEADERS FROM 'file:///data.csv' AS row
MATCH (p:Patient {name: row.Name})
MATCH (h:Hospital {name: row.Hospital})
MERGE (p)-[r:ADMITTED_TO]->(h)
SET r.dateOfAdmission = row.`Date of Admission`,
    r.dischargeDate = row.`Discharge Date`,
    r.billingAmount = toFloat(row.`Billing Amount`),
    r.roomNumber = row.`Room Number`,
    r.admissionType = row.`Admission Type`,
    r.testResults = row.`Test Results`;
```



Şekil 5.8 Hasta-hastane modellenmesi

- **Patient-Disease İlişkisi**

```
LOAD CSV WITH HEADERS FROM 'file:///data.csv' AS row
MATCH (p:Patient {name: row.Name})
MATCH (disease:Disease {name: row.`Medical Condition`})
MERGE (p)-[:HAS_CONDITION]->(disease);
```



Şekil 5.9 Hasta-teşhis modellenmesi

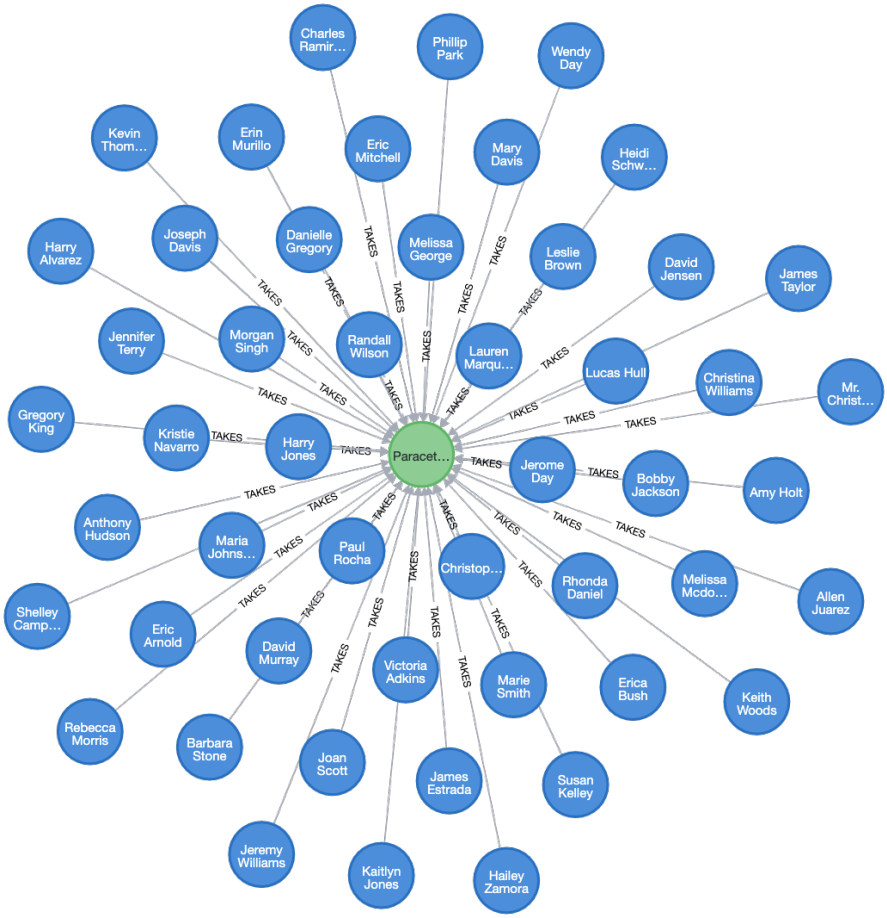
- **Patient-Insurance Provider İlişkisi**

```
LOAD CSV WITH HEADERS FROM 'file:///data.csv' AS row
MATCH (p:Patient {name: row.Name})
MATCH (i:InsuranceProvider {name: row.`Insurance Provider`})
MERGE (p)-[:INSURED_BY]->(i);
```

Şekil 5.10 Hasta-sigorta şirketi modellemesi

- **Patient-Medication İlişkisi**

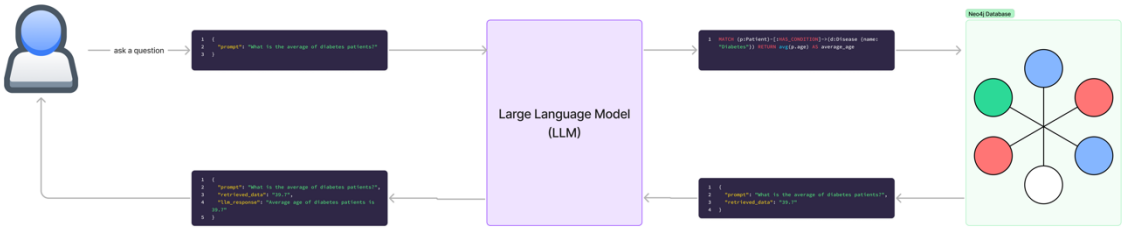
```
LOAD CSV WITH HEADERS FROM 'file:///data.csv' AS row
MATCH (p:Patient {name: row.Name})
MATCH (m:Medication {name: row.Medication})
MERGE (p)-[:TAKES]->(m);
```



Şekil 5.11 Hasta-ilaç modellemesi

BÖLÜM 6
MODEL

Neo4j üzerinde modellenen çizge veritabanı kullanılarak LLM ile bilgiye dayalı bir soru-cevap sistem geliştirilmiştir. Sistem, kullanıcının doğal dilde sorduğu soruları anlayarak, veritabanında yakınsadığı bilgileri kullanır ve cevaplar üretmek için RAG mimarisini kullanır. Aşağıda geliştirilen genel sistemin akışı görülmektedir.



Şekil 6.1 Geliştirilen sistem

1. **Kullanıcıdan Soru Alımı**
   Kullanıcı geliştirilen arayüz üzerinden metin kutusuna sorusunu girer. Örnek olarak Şekil 6.1 üzerinde de görüldüğü gibi *"What is the average age of diabetes patients?"* sorusunu sorar. Arayüz üzerinden soru JSON formatında LLM'e gönderir.

2. **Soruya Göre Cypher Sorgusunun Oluşturulması**
   LLM kullanıcının sorduğu soruyu analiz eder ve bağlama uygun bir şekilde Neo4j çizge veritabanına yönelik Cypher sorgusunu oluşturur. Örneğin, yukarıdaki soruya uygun olarak şu sorgu üretilir:

   ```
   MATCH (p:Patient)-[:HAS_CONDITION]->(d:Disease {name: "Diabetes"})
   RETURN avg(p.age) AS average_age
   ```

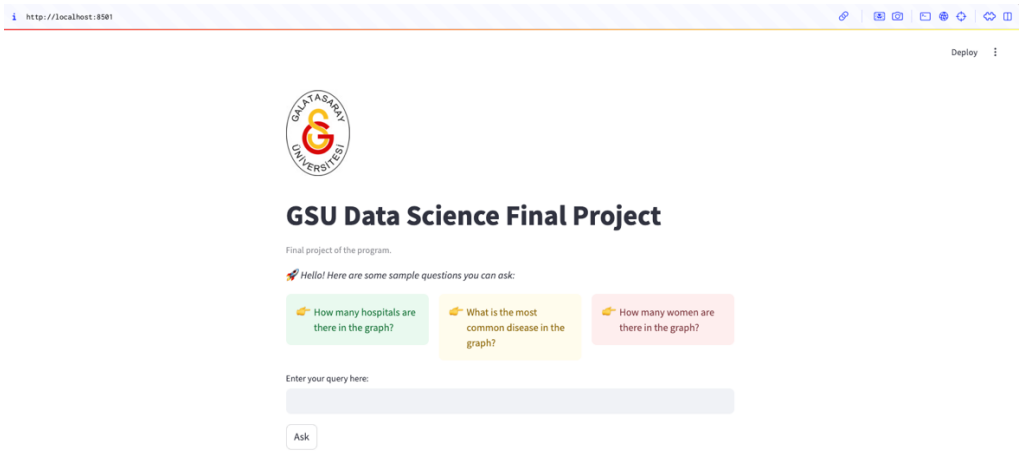
3. **Cypher Sorgusunun Veritabanına Gönderilmesi**
   Oluşturulan Cypher sorgusu ile Neo4j çizge veritabanına istek atılır ve cevap alınır.

4. **Elde Edilen Verinin İşlenmesi**
   Veritabanından dönen ham veri, tekrar LLM modeline iletilir. LLM, bu veriyi ve kullanıcının sorduğu soruyu alarak bağlama uygun bir yanıt üretir.

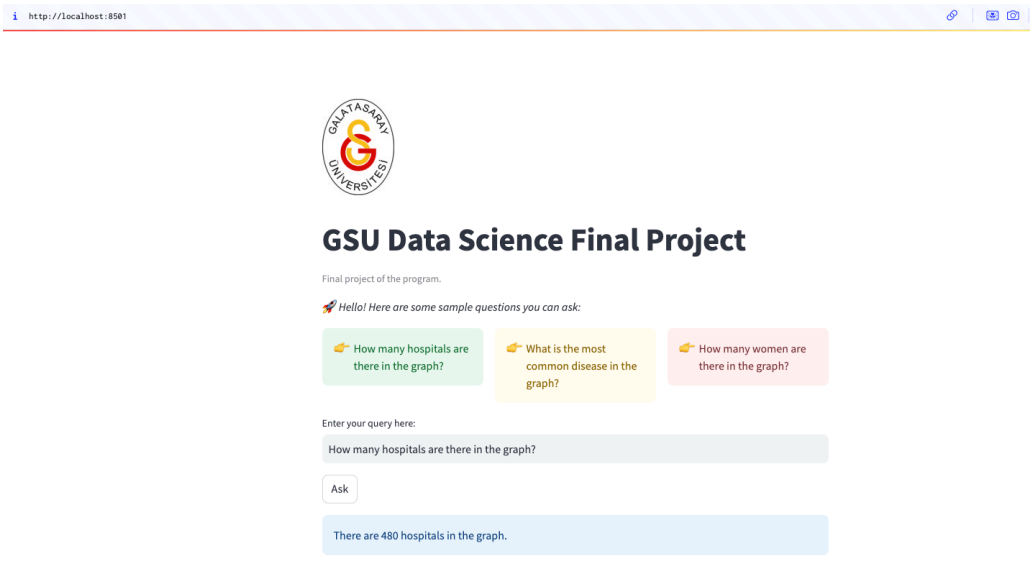
5. **Yanıtın Kullanıcıya Sunulması**
   LLM ile üretilen cevap metni Streamlit arayüzü aracılığıyla kullanıcıya iletilir.

Proje kapsamında son kullanıcı kullanımına sunmak amacıyla Streamlit ile bir arayüz geliştirilmiştir [7]. Streamlit, kullanıcıların sorduğu sorularını yazabildiği bir metin kutusu ve yanıtları görüntüleyebildiği bir arayüz sağlar. Arayüz tüm süreci baştan sona

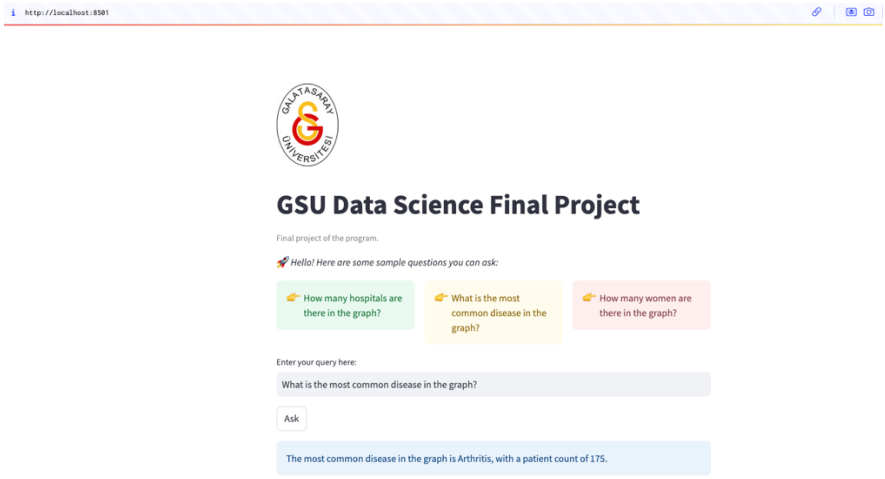
otomatize hale getirir ve son kullanıcının teknik bilgiye ihtiyaç duymadan geliştirilen sistemi kullanmasını sağlar. Aşağıda geliştirilen arayüz üzerinde sorulan bazı örnek sorular ve alınan cevaplar gösterilmiştir.



Şekil 6.2 Geliştirilen Streamlit arayüzü



Şekil 6.3 Streamlit arayüz ile örnek soru cevap



Şekil 6.4 Streamlit arayüz ile örnek soru cevap

BÖLÜM 7
SONUÇ

Proje kapsamında Neo4j çizge modelleme, OpenAI LLM, LangChain destekli RAG ve Streamlit arayüzü kullanılarak sağlık verisi üzerinde bilgi çıkarımı sistemi geliştirilmiştir. Çizge modelleme sayesinde hasta-doktor-hastane ilişkileri gibi karmaşık yapılar anlaşılır hale getirilmiş, LLM ve RAG entegrasyonlarıyla sağlık verilerinden doğru ve hızlı bilgi çıkarımı yapılmıştır.

Geliştirilen sistemin başarısını test etmek için bir test veriseti kümesi oluşturulmuştur. Modelin performansı, **100 adet çeşitli sorudan oluşan bir test kümesi oluşturularak** değerlendirildi. Sorular içerisinde basit ve karmaşık olarak çeşitlendirilmiş, hasta bilgileri, doktor-hastane ilişkileri, kullanılan ilaçlar ve finansal veriler gibi tüm düğüm ve ilişkilere değinen kısımlar bulunmaktadır. Bunun yanında veritabanında yer almayan bazı bilgiler de sorulara eklenmiştir. Bu sayede modelin halüsinasyon olasılığı olan sorularda performansı gözlemlenmiştir. Değerlendirme için kullanılan metrikler aşağıda açıklanmıştır.

True Positive: Sistemin verdiği cevap ground truth cevap ile eşleşir.
- Soru: "Which hospital admitted Kevin Hicks?"
- Sistem Yanıtı: "Robinson Inc"
- Ground Truth: "Robinson Inc"

True Negative: Sisteme veritabanında yer almayan bir bilginin sorulduğunda cevap vermez.
- Soru: "Which hospital admitted a patient named John Doe?" (Veritabanında John Doe bulunmuyor.)
- Sistem Yanıtı: "No relevant information found."
- Ground Truth: "No relevant information."

False Positive: Sistemin verdiği cevap ground truth cevap ile eşleşmez.
- Soru: "Which hospital admitted Kevin Hicks?"
- Sistem Yanıtı: "City Clinic"
- Ground Truth: "Robinson Inc"

False Negative: Sisteme veritabanında yer alan bir bilgi sorulduğunda cevap vermez.
- Soru: "Which hospital admitted Kevin Hicks?"
- Sistem Yanıtı: "No relevant information found."
- Ground Truth: "Robinson Inc"

Precision: Sistemin verdiği doğru cevapların, verdiği toplam cevaplara oranıdır.

Recall: Sistemin verdiği doğru cevapların, veritabanında yer alan tüm doğru cevaplara oranıdır.

F1-Score: Precision ve recall'un harmonik ortalamasıdır, doğruluk ve duyarlılık arasında dengeyi ölçer.

Accuracy: Sistemin doğru verdiği cevapların (hem TP hem TN) toplam cevaplara oranıdır.

Değerlendirme metriklerinin sonuçları aşağıdaki tabloda görülmektedir.

Tablo 7.1 Değerlendirme sonuçları

Metrik	**Sonuç**
TP	59
TN	23
FP	3
FN	15
Precision	0.8
Recall	0.95
F1-Score	0.86
Accuracy	0.82

Modelin performans analizi özellikle %79.73 precision ve %95.16 recall değerleri üzerine yapılmıştır. Yüksek recall değeri modelin doğru cevapları büyük ölçüde yakaladığını ancak %20 oranında yanlış bilgi ürettiğini göstermektedir. Bu durum, özellikle retrieve edilen bilginin bağlama uygun olmayan düğümlerden bilgi çekilmesi veya yanlış sonuç üreten Cypher sorgularının sonucu olabilir. Ayrıca, FP oranının düşük olması, doğru bilgi çıkarımında modelin doğru çalıştığına işaret etmektedir. Accuracy ve F1-score değerleri ise modelin yapılacak iyileştirmelerle yüksek performansa ulaşabileceğini göstermektedir.

Bu çalışmada geliştirilen sistem, belirli bir doğruluk oranının üzerine çıksa da ve bazı eksiklikler ve geliştirmeye açık alanlar dikkat çekmektedir. FP sayısının fazla olması modelin bazı durumlarda bağlama uygun olmayan yanıtlar üretme eğiliminde olduğuna işaret eder. Bu sorun özellikle veri çıkarımı aşamasında kullanılan Cypher sorgusu oluşturmanın daha bağlamsal hale getirilmesiyle iyileştirilebilir. Diğer bir eksiklik, modelin bazı karmaşık çok adımlı ilişkilerde doğru çıkarım yapamaması olarak gözlemlenmiştir. Veri setindeki çeşitliliğin artırılmasıyla modelin daha geniş kullanım senaryolarıyla test edilmesi, geliştirilmesi gereken bir diğer alandır.

KAYNAKÇA

[1] "Graph Database", 2024, https://en.wikipedia.org/wiki/Graph_database

[2] "Neo4j Graph Database & Analytics", 2024, https://neo4j.com

[3] "Attention Is All You Need", 2017, https://arxiv.org/pdf/1706.03762

[4] "Unleashing the Power of Vectors and Embeddings with Vector Databases", 2024, https://www.linkedin.com/pulse/unleashing-power-vectors-embeddings-vector-databases-huaping-gu/

[5] "What is RAG: Understanding Retrieval-Augmented Generation", 2024, https://qdrant.tech/articles/what-is-rag-in-ai/

[6] "Healthcare Dataset", 2024, https://www.kaggle.com/datasets/prasad22/healthcare-dataset

[7] "Streamlit Main Concepts", 2024, https://docs.streamlit.io/get-started/fundamentals/main-concepts