Drug Recommendation System: based on user's allergies using Ontology

Venkata Vamsi Krishna Bhuvanam[1] ID: 5, Priyadarsini Nidadavolu[1] ID: 26, Bhulakshmi Makkena[1] ID: 16, Tej Kumar Yentrapragada[1] ID: 48.

[1] Graduate Student CSEE Department, University of Missouri Kansas City, Kansas City, Missouri

Abstract:

The paper presents a workflow and ontology based architecture for recommending drugs or medicine compositions based on user allergies. For enabling the closing of gap between the two different entities like Patient to Diseases prediction concept. Drugs.com and other database from European medical dataset which is open to all, helped to create a backbone to the knowledge base which we tend to create. Once the knowledge base is created, a rule based interface will help to communicate from Front-End to knowledge base which we have created using machine learning libraries. This framework or architecture provides a strong base and efficient way for recommending medicines or drugs to patient based on his own preferences like allergies to medical compositions.

Introduction:

Studies show that some diseases will show symptoms slowly and keeps on change or add new symptoms which are sometimes difficult to track and relate to a disease. And delay can deteriorate the condition of the patient based on the stage of the disease. So this system will take the symptoms as input and the allergies of the user which consists of some medical composition. Based on the symptoms and the provided user data the machine will recommend the user with expected diseases (which calculated by comparing different categories collected from drugs.com and other data sources) where we can select the number of diseases to predict and get a final output with drug compositions suggested for that particular drug as well.

Getting accuracy more than 60 % is a great challenge since health industry or medical industry is a sensitive field where even 99 % is sometimes not acceptable when suggesting a drug for Neuropsychiatric conditions and infectious or parasitic diseases and chronic diseases. But this framework or architecture can be used as informational purpose which can replace doctors if the diseases are in initial stages or normal medical condition. This solution will provide somewhere around 60 to 70 % of accuracy based on the size of the data set used for knowledge base creation (ontology).

The main objective of the recommendation system for E-drug is to recommend a drug to the user based on the symptoms that he provided. Our system takes user symptoms as input and it analyses the input with the types of diseases and produces a dataset with various possible diseases.

It analysis the type of diseases and ultimately recommends an appropriate drug based on the following criteria:

- Past Behavior of the patient
- Historical data of the past similar symptoms of the patients
- Drug allergies
- Drug similarities that the doctor recommended previously

Our recommendation system would return the dataset of list of recommended drugs with a hyperlink which leads to the composition, availability of the drug in various stores, the cost etc. This would help the user to view the list and can go and check for nearby pharmacy and buy it.

Machine learning and ALS algorithms run in the background and the result would be an recommended drug dataset.

Related Work:

The data set was taken from drugs.com where they have distinct categories of diseases with related symptoms and related articles published on that particular disease. Here we collected data based on URL by providing different condition or disease in the URL filler and get the data related to particular condition. From there system downloaded disease references, care notes, medication list and the entire encyclopedia collected on that medical condition. Also some of the articles published by Mayo clinic and Harvard Health Guide. This information will be analyzed by machine and divide into topics or categories and then compared to the symptoms provided.

Proposed Solution:

Based on the data collected we used two approaches which are creating a Training set at first and later creating a testing set based on the symptoms or user input, which are done using different algorithms to filter the data and recognize the relations between the entities identified.

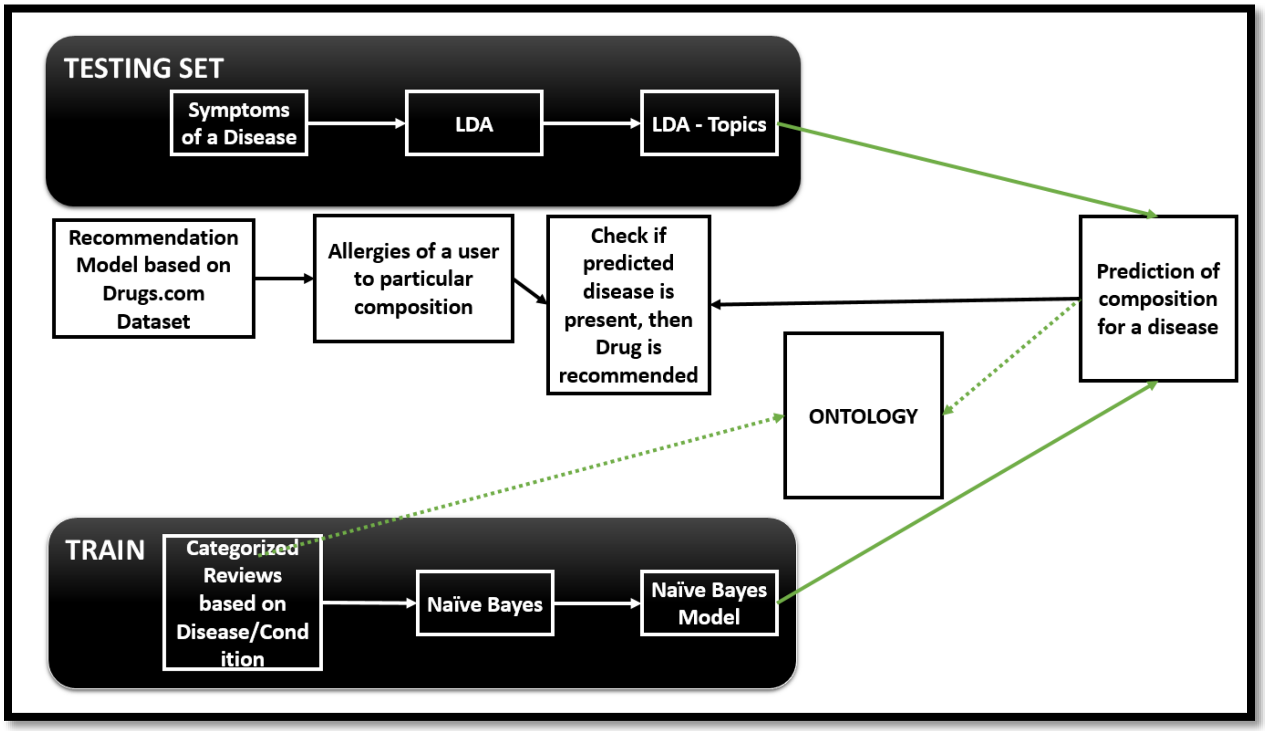

TESTING SET
Symptoms of a Disease
LDA
LDA - Topics
Recommendation Model based on Drugs.com Dataset
Allergies of a user to particular composition
Check if predicted disease is present, then Drug is recommended
Prediction of composition for a disease
ONTOLOGY
TRAIN
Categorized Reviews based on Disease/Condition
Naïve Bayes
Naïve Bayes Model

Fig 1: Workflow of the proposed solution.

Creation of Training Set – Here we apply Naïve Bayes algorithm on the data set collected, here the algorithm first calculates the probability of occurring particular group of texts and generates a set of RDF's. On these RDF's the machine will perform NLP and learns the meaning of each sentence and then after removal of stop words it further generates another set of rdf's.

In testing the model, the symptoms are given as input, LDA is another algorithm which is performed on the testing set to analyse the main topics of the given context of data. Later the documents are categorized and compared with the training model generated.

Tasks & Features:

This project definitely need a lot of data and to analyze that and get to a strong knowledge base. Using that knowledge, the machine will recommend the drug. Here there will be a lot of features to be considered. For example, first the machine will analyze the symptoms and different

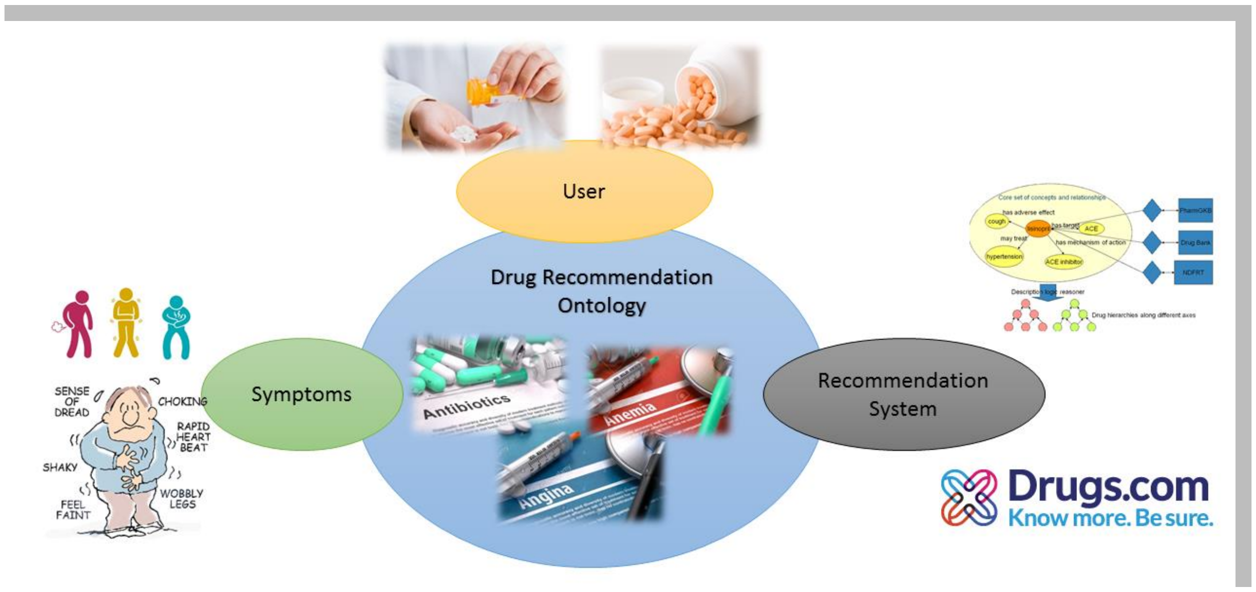


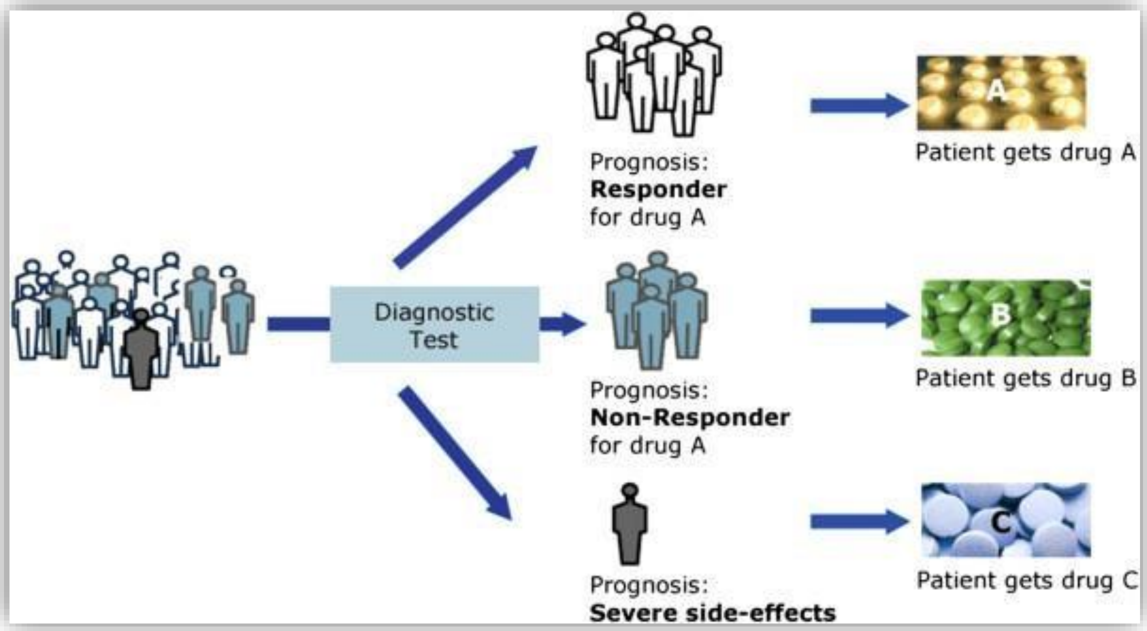
Fig 2: Architecture of proposed Recommendation system with ontology.

types of diseases. This machine will take a lots of criteria into consideration like past behavior of the patient, historical data of other patients and their acceptance or rejection of the drug. Also this machine will analyze the drug allergies of a particular patient and based on the finalized knowledge created, the machine will recommend some drugs, so that the doctors can have a look and prescribe particular drugs to patients.

Implementation:

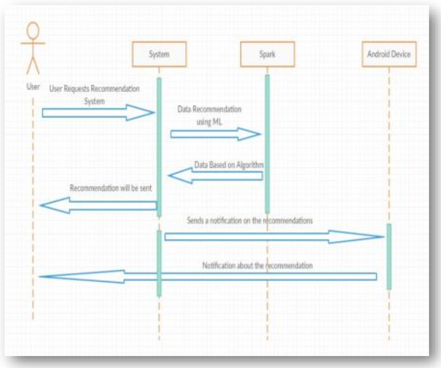
GITHUB Link: https://github.com/Summer2016-KDM/KDM-Summer-2016-Strikerz

Youtube Link: https://www.youtube.com/watch?v=0SZruUDKZBw

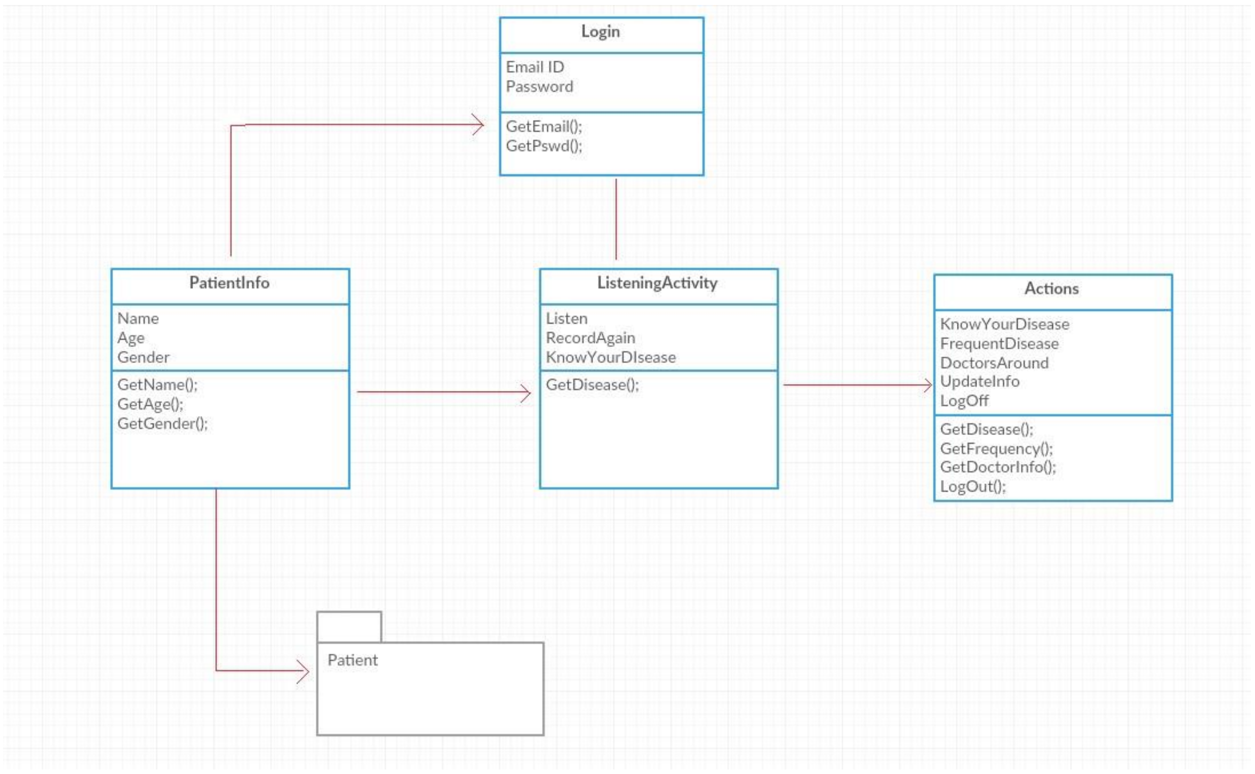

Diagnostic Test
Prognosis: Responder for drug A
Prognosis: Non-Responder for drug A
Prognosis: Severe side-effects
A
B
C
Patient gets drug A
Patient gets drug B
Patient gets drug C

In traditional way, diagnostic test will be done first and then a drug will be used to a person where if he responds to it, then doctor will recommend to continue that drug A for better outcome, if by any reason the patient will not respond to that drug A, then doctor will recommend drug B. At-last if patient shows some severe side effects even after using drug B, then doctor will recommend drug C which is a long and time taking process. This can be changed by the above proposed approach.

Sequence Diagram:


User
System
Spark
Android Device
User Requests Recommendation System
Data Recommendation using ML
Data Based on Algorithm
Recommendation will be sent
Sends a notification on the recommendations
Notification about the recommendation

Class Diagram:



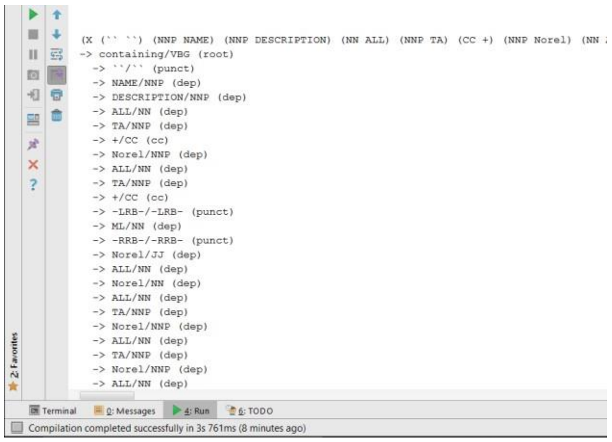
Services:

Following are the services and algorithms used to filter and analyze the data:

Collaborative filtering (CF) is a technique that we used to filter the information and produce an output Dataset. Collaborative filtering methods have been used on the input data which include the symptoms and identifies the type of diseases. Later it is used to analyze the drugs that are prescribed for such kinds of diseases based on the given criteria.

Natural Language Processing (NLP) is the technique that we have used to processes the input given by the User. Here the NLP takes the input as a symptoms and it processes the data to enable computer to derive meaning.

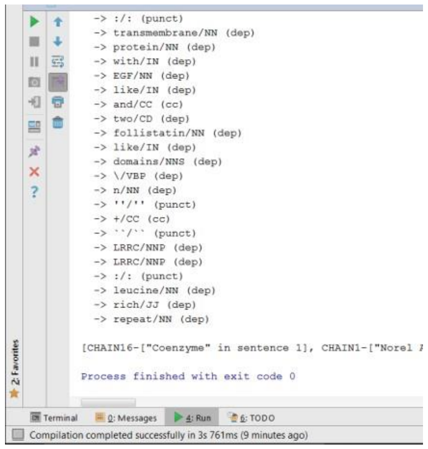
Parse tree:



```
(X (`` ``) (NNP NAME) (NNP DESCRIPTION) (NN ALL) (NNP TA) (CC +) (NNP Norel) (NN
-> containing/VBG (root)
  -> ``/`` (punct)
  -> NAME/NNP (dep)
  -> DESCRIPTION/NNP (dep)
  -> ALL/NN (dep)
  -> TA/NNP (dep)
  -> +/CC (cc)
  -> Norel/NNP (dep)
  -> ALL/NN (dep)
  -> TA/NNP (dep)
  -> +/CC (cc)
  -> -LRB-/-LRB- (punct)
  -> ML/NN (dep)
  -> -RRB-/-RRB- (punct)
  -> Norel/JJ (dep)
  -> ALL/NN (dep)
  -> Norel/NN (dep)
  -> ALL/NN (dep)
  -> TA/NNP (dep)
  -> Norel/NNP (dep)
  -> ALL/NN (dep)
  -> TA/NNP (dep)
  -> Norel/NNP (dep)
  -> ALL/NN (dep)
```

```
Compilation completed successfully in 3s 761ms (8 minutes ago)
```

Process Execution:

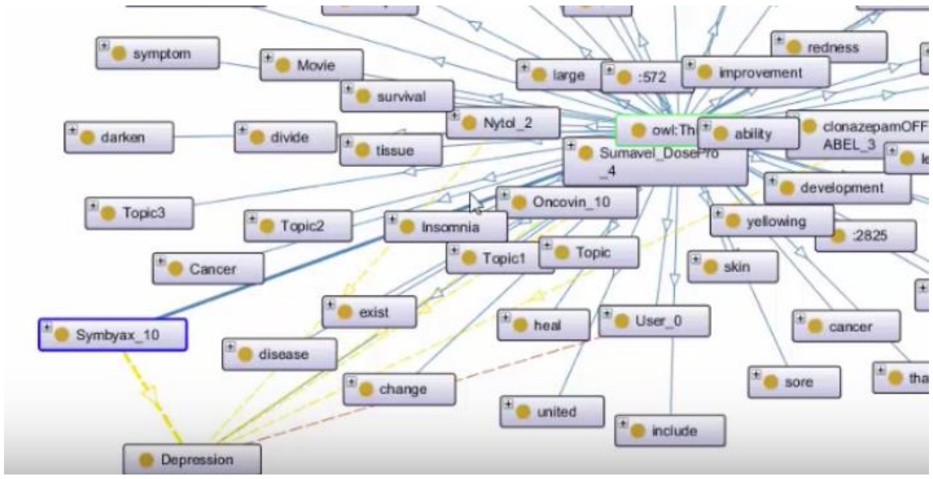


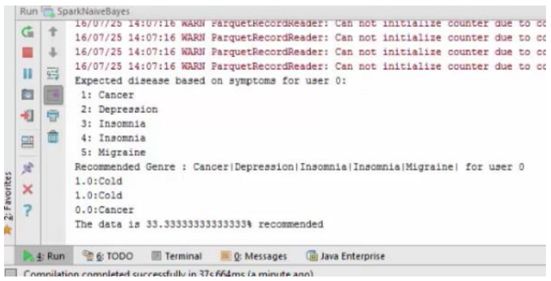
```
  -> :/: (punct)
  -> transmembrane/NN (dep)
  -> protein/NN (dep)
  -> with/IN (dep)
  -> EGF/NN (dep)
  -> like/IN (dep)
  -> and/CC (cc)
  -> two/CD (dep)
  -> follistatin/NN (dep)
  -> like/IN (dep)
  -> domains/NNS (dep)
  -> \/VBP (dep)
  -> n/NN (dep)
  -> ''/'' (punct)
  -> +/CC (cc)
  -> ``/`` (punct)
  -> LRRC/NNP (dep)
  -> LRRC/NNP (dep)
  -> :/: (punct)
  -> leucine/NN (dep)
  -> rich/JJ (dep)
  -> repeat/NN (dep)

[CHAIN16-["Coenzyme" in sentence 1], CHAIN1-["Norel

Process finished with exit code 0
```

```
Compilation completed successfully in 3s 761ms (9 minutes ago)
```

Results and Evaluation:

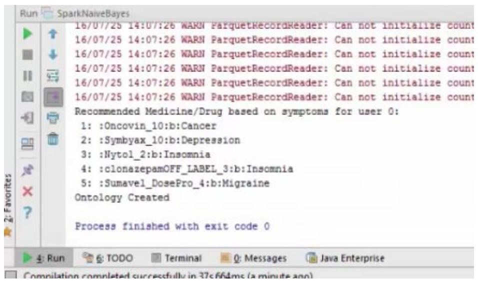
Ontology:





```
16/07/25 14:07:16 WARN ParquetRecordReader: Can not initialize counter due to cc
16/07/25 14:07:16 WARN ParquetRecordReader: Can not initialize counter due to cc
16/07/25 14:07:16 WARN ParquetRecordReader: Can not initialize counter due to cc
16/07/25 14:07:16 WARN ParquetRecordReader: Can not initialize counter due to cc
Expected disease based on symptoms for user 0:
 1: Cancer
 2: Depression
 3: Insomnia
 4: Insomnia
 5: Migraine
Recommended Genre : Cancer|Depression|Insomnia|Insomnia|Migraine| for user 0
1.0:Cold
1.0:Cold
0.0:Cancer
The data is 33.33333333333333% recommended
```

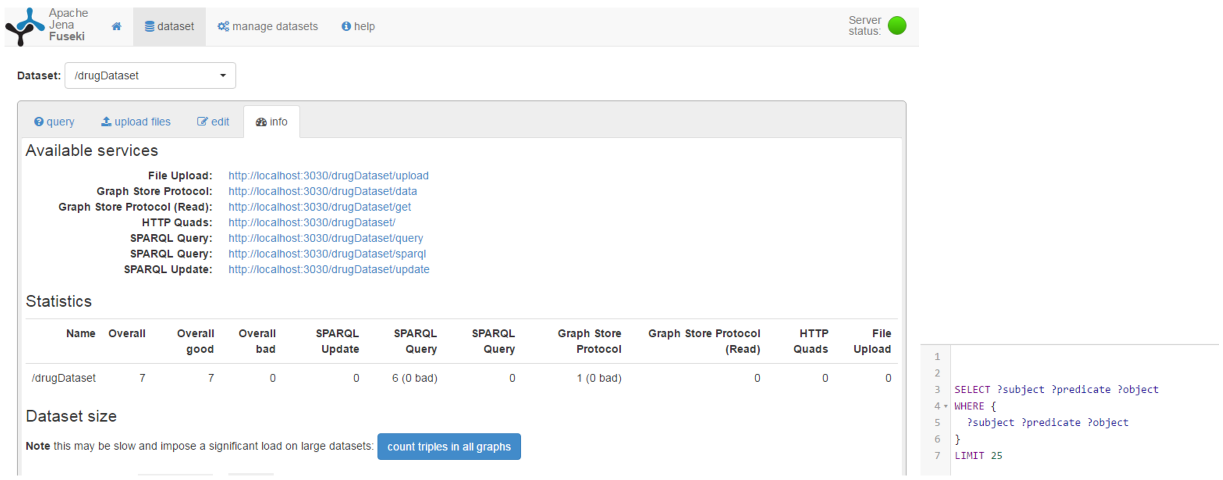
```
Compilation completed successfully in 37s 664ms (a minute ago)
```



```
16/07/25 14:07:26 WARN ParquetRecordReader: Can not initialize count
16/07/25 14:07:26 WARN ParquetRecordReader: Can not initialize count
16/07/25 14:07:26 WARN ParquetRecordReader: Can not initialize count
16/07/25 14:07:26 WARN ParquetRecordReader: Can not initialize count
16/07/25 14:07:26 WARN ParquetRecordReader: Can not initialize count
16/07/25 14:07:26 WARN ParquetRecordReader: Can not initialize count
Recommended Medicine/Drug based on symptoms for user 0:
 1: :Oncovin_10:b:Cancer
 2: :Symbyax_10:b:Depression
 3: :Nytol_2:b:Insomnia
 4: :clonazepamOFF_LABEL_3:b:Insomnia
 5: :Sumavel_DosePro_4:b:Migraine
Ontology Created

Process finished with exit code 0
```

```
Compilation completed successfully in 37s 664ms (a minute ago)
```

Apache Jena Fuseki (SPARQL):



Dataset: /drugDataset

Available services

File Upload:	http://localhost:3030/drugDataset/upload
Graph Store Protocol:	http://localhost:3030/drugDataset/data
Graph Store Protocol (Read):	http://localhost:3030/drugDataset/get
HTTP Quads:	http://localhost:3030/drugDataset/
SPARQL Query:	http://localhost:3030/drugDataset/query
SPARQL Query:	http://localhost:3030/drugDataset/sparql
SPARQL Update:	http://localhost:3030/drugDataset/update

Statistics

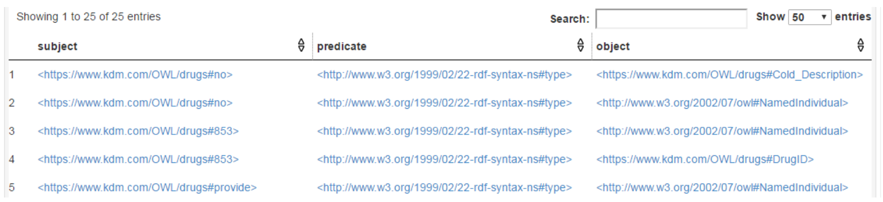
Name	Overall	Overall good	Overall bad	SPARQL Update	SPARQL Query	SPARQL Query	Graph Store Protocol	Graph Store Protocol (Read)	HTTP Quads	File Upload
/drugDataset	7	7	0	0	6 (0 bad)	0	1 (0 bad)	0	0	0

Dataset size

Note this may be slow and impose a significant load on large datasets: count triples in all graphs

```


SELECT ?subject ?predicate ?object
WHERE {
  ?subject ?predicate ?object
}
LIMIT 25
```

Sample Output:



Showing 1 to 25 of 25 entries

	subject	predicate	object
1	<https://www.kdm.com/OWL/drugs#no>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<https://www.kdm.com/OWL/drugs#Cold_Description>
2	<https://www.kdm.com/OWL/drugs#no>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<http://www.w3.org/2002/07/owl#NamedIndividual>
3	<https://www.kdm.com/OWL/drugs#853>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<http://www.w3.org/2002/07/owl#NamedIndividual>
4	<https://www.kdm.com/OWL/drugs#853>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<https://www.kdm.com/OWL/drugs#DrugID>
5	<https://www.kdm.com/OWL/drugs#provide>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<http://www.w3.org/2002/07/owl#NamedIndividual>

Conclusion:

This project is still not fully implemented in view of front end and back end connections. Also the data set is still getting updated and developed using many other data sources since medical industry is more sensitive and accuracy is more important when predicting disease based on symptoms without any further diagnosis and suggesting drugs.

Future Work:

One can still implement front end which will connect to back-end which is OWL file using SQWRL which is a Semantic Web Query Rules Language. Also we can convert the created owl to rdf using a simple java code or online owl to rdf converters and the converted RDF file can be used as backend at Apache Jena Fuseki and using SPARQL and POST and GET methods, one can get the output using knowledge base which was created.

References:

http://cs.stanford.edu/people/widom/paper-writing.html

https://en.wikipedia.org/wiki/Disease

https://www.drugs.com/medical_conditions.html

http://nlp.stanford.edu/software/

http://nlp.stanford.edu/software/CRF-NER.html

http://spark.apache.org/docs/latest/mllib-guide.html

http://spark.apache.org/mllib/

https://spark.apache.org/docs/1.2.1/mllib-guide.html

http://spark.apache.org/docs/latest/

http://spark.apache.org/docs/latest/streaming-programming-guide.html