SymptoGraph: Evaluating BiLSTM vs. CNN for Symptom-Based Disease Prediction with Knowledge Graph Integration

1st Shireesh Reddy Pyreddy
Dept. of Computer Science
SUNY Polytechnic Institute
Utica, NY, USA
pyredds@sunypoly.edu

***Abstract*—This study introduces SymptoGraph, a novel approach to disease prediction using symptom descriptions, focusing on a comparative analysis of BiLSTM and CNN neural network architectures. Previous efforts in this domain predominantly utilized conventional machine learning techniques. Our work extends this by integrating a knowledge graph construction, which facilitates a deeper understanding of the semantics and relationships between symptoms and diseases. The research systematically compares the predictive capabilities of BiLSTM and CNN, considering their ability to process and interpret complex symptom data. The incorporation of knowledge graphs aims to enhance the contextual understanding of diseases, potentially leading to more accurate and semantically rich disease predictions. This project not only provides insights into the relative strengths of BiLSTM and CNN in medical diagnosis but also illustrates the added value of knowledge graph integration in advancing the field of symptom-based disease prediction.**

***Index Terms*—SymptoGraph, Bidirectional Long Short-Term Memory, Convolutional Neural Network, Knowledge Graph, Medical Diagnostic AI, Neo4J.**

I. Introduction

In recent years, the application of artificial intelligence in healthcare, particularly for disease prediction and diagnosis, has witnessed significant advancements. The ability to accurately predict diseases based on symptom descriptions not only enhances patient care but also aids in early diagnosis and treatment planning [2]. Traditional approaches have primarily leveraged various machine learning techniques [1, 2, 3, 4], which, while effective, often fall short in handling the complexity and nuances of medical data.

With the advent of deep learning, new avenues have opened up in the realm of medical diagnostics. Neural networks architectures, such as Bidirectional Long Short-Term Memory (BiLSTM) and Convolutional Neural Networks (CNN) [2], have shown remarkable capabilities in processing and interpreting complex datasets. These models have the potential to capture intricate patterns in symptom progression and disease manifestation, far beyond the reach of traditional machine learning algorithms.

However, symptom-based disease prediction presents its own set of issues. Symptoms frequently overlap among disorders, and their presentation varies greatly from patient to patient [3]. Because of this diversity, models that are not only accurate but also capable of grasping the context and semantics of symptom data are required. Most current models struggle in this area, necessitating the need for a new strategy that overcomes this gap.

This study introduces SymptoGraph, a novel framework that not only utilizes the strengths of BiLSTM and CNN for disease prediction but also innovatively integrates knowledge graphs. Knowledge graphs offer a structured way to represent and understand the relationships and semantics between different medical entities, such as symptoms and diseases. By combining deep learning models with knowledge graphs, we aim to enhance the models' ability to make more informed and contextually aware predictions.

The primary objective of this research is to evaluate and compare the effectiveness of BiLSTM and CNN in the context of symptom-based disease prediction, with the integration of knowledge graphs. By doing so, we seek to not only advance the field of medical AI but also provide a comprehensive tool that could potentially transform diagnostic processes in healthcare [9]. The integration of knowledge graphs is expected to provide a deeper understanding of symptom-disease relationships, thereby improving the accuracy and reliability of predictions.

II. Motivation

The confluence of Natural Language Processing (NLP) and healthcare has proven to be a fertile ground for transformative advancements, opening avenues to revolutionize disease prediction and patient care. Our motivation stems from the recognition that harnessing the power of NLP techniques such as Text Classification, Feature Extraction, Named Entity Recognition and Knowledge Graph Generation can significantly enhance our understanding of healthcare data, particularly in the context of symptom-based disease prediction.

In the vast landscape of healthcare, where patient narratives are rich with valuable information, NLP offers a unique opportunity to extract meaningful insights from unstructured

textual data. By delving into the nuances of patient-reported symptoms, we aim to leverage NLP methodologies to decipher intricate patterns and subtle associations that may elude conventional analysis. This exploration is inherently driven by the belief that the narrative nature of symptom descriptions holds untapped potential for more accurate and personalized disease predictions [5].

The intersection of NLP with healthcare is not only timely but imperative, considering the growing volume and complexity of health-related data. Traditional methods often struggle to keep pace with the intricacies of textual information, necessitating the adoption of advanced computational models. In our pursuit, we focus on comparing two powerful neural network architectures—Bidirectional Long Short-Term Memory (BiLSTM) and Convolutional Neural Network (CNN)—to discern their effectiveness in unraveling the latent structures within symptom narratives.

Moreover, recognizing the importance of domain-specific knowledge, we introduce a novel dimension to our study by integrating a knowledge graph. This augmentation aims to encapsulate the contextual relationships inherent in healthcare data, providing a structured foundation for our predictive models. The amalgamation of NLP techniques with a knowledge graph framework reflects our commitment to developing not only accurate but also interpretable models that align with the intricacies of healthcare practice.

Ultimately, our venture into this intersection of NLP and healthcare seeks to contribute to a paradigm shift in disease prediction. By advancing our understanding of symptom-based patterns through the lens of sophisticated NLP models, we aspire to empower healthcare professionals with tools that can elevate diagnostic precision, improve patient outcomes, and foster a more proactive approach to personalized medicine. Through this endeavor, we aim to underscore the transformative potential of NLP in reshaping the landscape of healthcare analytics.

III. Background

The algorithms and tools employed in this research project are outlined in detail below, presenting a comprehensive overview of the methodology.

In the initial phase, we employ the Pandas library for efficient data handling, reading the dataset that serves as the cornerstone for our research. Moving forward, exploratory data analysis is conducted using Matplotlib and WordCloud libraries, offering insights into textual patterns within the dataset. With an emphasis on data preparation, we utilize label encoding via scikit-learn to transform categorical labels, ensuring compatibility with subsequent deep learning models.

Ensuring data quality is paramount, and to achieve this, Natural Language Toolkit (nltk) stopwords and the regex module (re) are employed for data cleaning. This involves the removal of stopwords, punctuation, numbers, and symbols, enhancing the text data for downstream analysis. For robust model evaluation, the dataset is split into training (80%), validation (10%), and test (10%) sets, establishing a comprehensive framework.

Text feature extraction is performed using TensorFlow's text-to-sequence functionality, followed by padding for consistent input dimensions. Subsequently, two neural network architectures—Convolutional Neural Network (CNN) and Bidirectional Long Short-Term Memory (BiLSTM)—are trained on the preprocessed text data, facilitating a comparative analysis of their performance.

Introducing semantic understanding into the analysis, a Named Entity Recognition (NER) model is trained using Spacy to extract entities from the text, enriching the dataset with valuable information. Entities identified through NER are then utilized to generate triples in the form of subject-relation-object, providing a structured representation of semantic relationships.

To leverage the benefits of a graph-based representation, the generated triples are inserted into Neo4j, a graph database. This integration enables efficient querying of relationships, offering deeper insights into the underlying semantic structure of the data.

IV. Approach

Fig. 1 represents the workflow or the architecture of the project. Each part is explained in detail in the following paragraphs.

A. Data Collection

The dataset under consideration is a curated collection of 1200 datapoints, each comprising two essential columns: "label" and "text." The "label" column encapsulates disease labels, while the "text" column contains detailed natural language descriptions of symptoms. This diverse dataset spans 24 distinct diseases, each represented by 50 symptom descriptions, resulting in a comprehensive set of 1200 datapoints. Covering a spectrum of medical conditions, the dataset encompasses diseases shown in Fig 2.

The source of this dataset is Kaggle, a prominent platform for machine learning datasets and competitions. While Kaggle provides a valuable resource for diverse datasets, the challenges in leveraging this particular dataset lie in the intricacies of interpreting natural language symptoms and the nuanced distinctions among various diseases. The dataset's richness in covering a wide array of medical conditions also presents the challenge of ensuring balanced representation for each disease category, a factor critical for model training robustness.

Moreover, the natural language aspect of the dataset introduces potential challenges related to unstructured text data, including variations in symptom descriptions, linguistic nuances, and potential inconsistencies. Also, classifying among 24 distinct disease classes in a text classification problem is inherently challenging. Furthermore, the dataset's limited size poses a constraint, especially when training deep learning models such as Bidirectional Long Short-Term Memory (BiLSTM) and Convolutional Neural Network (CNN). The inherent complexity of these models demands substantial amounts of

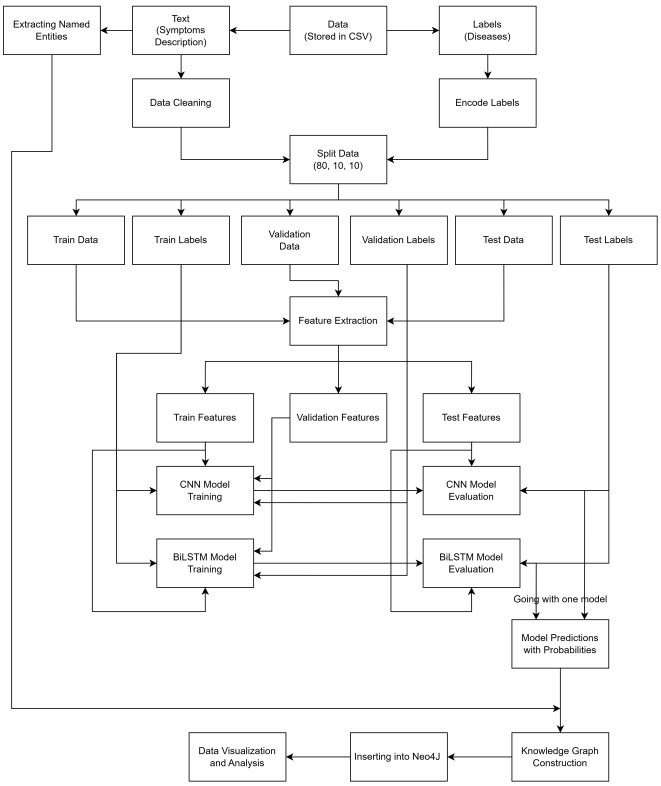

Fig. 1. The Architecture.

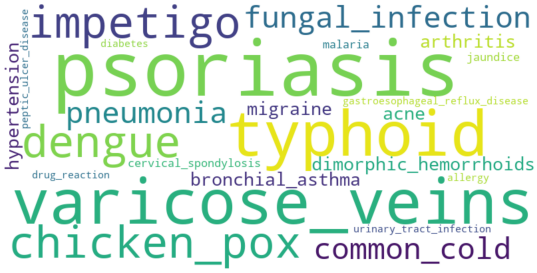

Fig. 2. Distribution of Different Diseases

The image depicts in Fig 3. is a histogram showcasing the distribution of text lengths in a dataset. The x-axis, representing text length, ranges from 0 to 50, while the y-axis, indicating count, extends from 0 to 160. The most frequent text length, signified by the highest bar, is approximately 30 characters. The histogram is right-skewed, suggesting a few texts are significantly longer than the majority. This skewness, along with potential outliers represented by bars separated from the main distribution, could pose challenges in data analysis. To tackle these issues, we applied data cleaning methods such as removing stopwords, unwanted symbols like punctuation etc. to reduce the text size, data transformation techniques, such as padding to make the sequence of each text to be equal, logarithmic or square root transformations, to reduce skewness.

data for effective training, and the relatively smaller dataset size may lead to challenges in achieving optimal model performance. As we delve into the complexities of disease prediction based on symptom descriptions, navigating these challenges becomes integral to the success of our approach. To address these issues, preprocessing steps such as cleaning, standardization, and careful consideration of models design will be implemented to ensure the model's robust performance across the diverse array of diseases present in the dataset. Also, to handle the ambiguity of the symptoms we introduce the Knowledge Graph to analyse the disease more effectively.

The graphical representation of the distribution of the diseases is shown in Fig. 2.

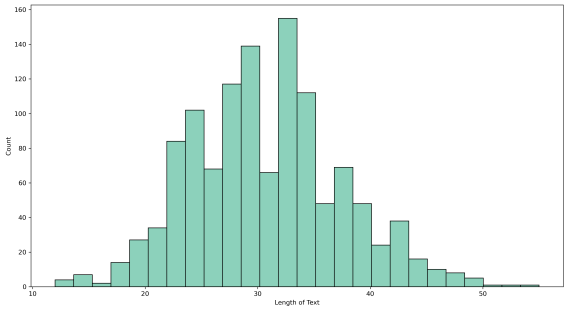

Fig. 3. Analyzing Symptom Descriptions Length

B. Exploratory Data Analysis

The next step includes performing exploratory data analysis on descriptions of the symptoms to get insights about the data. The analyses that were performed are, analyzing the text to understand the range of topics that are covered in the dataset, this is helpful to extract different entities from text across various diseases. Some other analyses were, checking the frequency of words used in the descriptions to remove them in the cleaning phase, and the word count distribution with regards to the text to check the outliers.

C. Data Cleaning

The dataset's integrity and the subsequent success of our deep learning models hinge on the effectiveness of the data cleaning process. Given the nature of natural language symptoms descriptions, this phase becomes crucial in mitigating potential challenges arising from unstructured text. Our first step involves leveraging the Natural Language Toolkit (nltk) stopwords and the regex module (re) to systematically remove stopwords, punctuation, numbers, and symbols from the "text" column. This ensures that the subsequent analysis is grounded

in the relevant linguistic content, devoid of extraneous elements that may introduce noise into the models.

Beyond basic text preprocessing, we address the inherent variability in symptom descriptions. Synonymous expressions or minor phrasing differences may exist across the dataset, contributing to potential discrepancies. To homogenize the language, we employ techniques such as lemmatization and stemming, standardizing words to their root form. This not only streamlines the vocabulary but also aids in capturing the essence of symptoms consistently, fostering a more cohesive understanding for downstream processing.

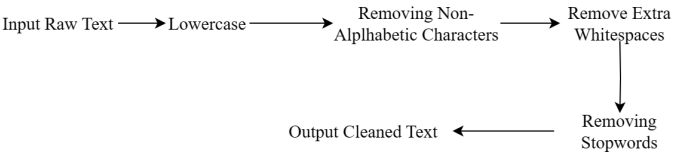

Fig. 4. Cleaning Process Pipeline.

A critical consideration is the potential ambiguity across the 24 diseases. Ambiguity samples among disease categories can bias model training, leading to suboptimal predictions for minority classes. Thus, we explore strategies like Knowledge Graph to achieve a more balanced representation, ensuring that our models are adept at discerning patterns across all diseases. The success of our data cleaning endeavors is pivotal in creating a robust foundation for subsequent feature extraction and model training, setting the stage for accurate and interpretable disease predictions in our comprehensive dataset.

D. Data Transformation

Once the data is cleansed and streamlined, the next pivotal phase involves transforming the raw textual information into a format suitable for machine learning models. This transformation is particularly crucial as it lays the groundwork for effective feature extraction and subsequent model training. In our approach, we leverage TensorFlow's text-to-sequence functionality to convert the natural language symptom descriptions into numerical sequences. This process involves mapping each unique word to a corresponding integer, facilitating the representation of textual data in a structured numerical format.

To address the varying lengths of symptom descriptions, we apply padding to ensure uniform input dimensions across all sequences. This not only simplifies the computational process but also allows for seamless integration with deep learning architectures, such as Bidirectional Long Short-Term Memory (BiLSTM) and Convolutional Neural Network (CNN), which require fixed-size inputs. The padded sequences serve as the input data for subsequent training, ensuring consistency and compatibility across the diverse symptom descriptions.

Furthermore, given the multiclass nature of our classification problem with 24 distinct diseases, we apply label encoder from scikit-learn library to represent the categorical "label" column. This technique transforms the disease labels into a numerical number, where each disease class is represented as a unique number between 0-24. This ensures that our models can effectively discern and learn from the diverse disease categories during training.

E. Algorithms and Data Modeling

The final step of the approach is performing data modeling on the extracted features. The modeling is broadly divided into two pipelines: Training CNN Model and Training BiLSTM Model and finally integrating the best model with Knowledge Graph.

a) Desiging and Training CNN Model: This pipeline undergoes designing the architecture of the CNN model and training it. The Convolutional Neural Network (CNN) model architecture depicted in the **Algorithm 1** consists of several layers. The first layer is an Embedding layer with an input dimension of 2000, an output dimension of 128, and an input length equal to the maximum length of the input sequences. This is followed by a Dropout layer with a rate of 0.3, which helps prevent overfitting. The next layer is a Conv1D layer with 128 filters, a kernel size of 5, and a ReLU activation function. This is followed by a GlobalAveragePooling1D layer, which helps reduce the spatial dimensions of the input. Another Dropout layer with a rate of 0.3 is added for further regularization. The model then includes a Dense layer with 32 units and a ReLU activation function, followed by another Dropout layer with a rate of 0.1 for additional regularization. The final layer is a Dense layer with 24 units and a softmax activation function, which outputs a probability distribution over the 24 classes. The model is compiled with the Adam optimizer and the sparse categorical crossentropy loss function, making it ready for training.

Algorithm 1 Construction of Convolutional Neural Network (CNN) Architecture

```
def cnn_architecture(max_length):
    cnn_model = tf.keras.Sequential([
        tf.keras.layers.Embedding(2000, 128,
                                  input_length=max_length),
        tf.keras.layers.Dropout(0.3),
        tf.keras.layers.Conv1D(128, 5, activation='relu'),
        tf.keras.layers.GlobalAveragePooling1D(),
        tf.keras.layers.Dropout(0.3),
        tf.keras.layers.Dense(32, activation='relu'),
        tf.keras.layers.Dropout(0.1),
        tf.keras.layers.Dense(24, activation='softmax')])

    print(cnn_model.summary())

    cnn_model.compile(loss='sparse_categorical_crossentropy',
                      optimizer='adam',
                      metrics=['accuracy'])

    return cnn_model
```

b) Desiging and Training BiLSTM Model: This pipeline undergoes designing the architecture of the BiLSTM model and training it. The provided code snippet in **Algorithm 2** is a Python implementation of a Long Short-Term Memory (LSTM) model using the Keras library, which is a high-level neural networks API capable of running on top of TensorFlow. The function 'bi_lstm_architecture' takes 'max_length' as an argument, which represents the maximum length of the input sequences. The model is initialized as a Sequential model, indicating that the layers are linearly stacked. The first layer is an Embedding layer with an input dimension of 2000, an output dimension of 128, and an input length equal to 'max_length'. This layer transforms integer-encoded vocabulary into dense vectors of fixed size.

Next, a Dropout layer is added with a rate of 0.3. This layer randomly sets a fraction of input units to 0 at each update during training time, which helps prevent overfitting. The model then includes an BiLSTM layer with 64 units. LSTM layers are a type of recurrent neural network (RNN) layer designed to learn order dependence in sequence prediction problems. This is followed by another Dropout layer with a rate of 0.2 for further regularization. The final layer is a Dense layer with 24 units and a softmax activation function. The softmax function outputs a vector that represents the probability distributions of a list of potential outcomes. It's often used in the final layer of a neural network-based classifier. The model is then compiled with the Adam optimizer and the categorical crossentropy loss function. The Adam optimization algorithm is an extension to stochastic gradient descent that has recently seen broader adoption for deep learning applications in computer vision and natural language processing. Categorical crossentropy is a loss function that is used in multi-class classification tasks. These are tasks where an example can only belong to one out of many possible categories, and the model must decide which one.

Algorithm 2 Construction of Bidirectional Long Short Term Memory (BiLSTM) Architecture

```
def bi_lstm_architecture(max_length):
    tf.random.set_seed(42)
    bi_lstm_model = tf.keras.Sequential([
        tf.keras.layers.Embedding(2000, 128,
                                  input_length=max_length),
        tf.keras.layers.Dropout(0.3),
        tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64)),
        tf.keras.layers.Dropout(0.2),
        tf.keras.layers.Dense(32, activation='relu'),
        tf.keras.layers.Dropout(0.1),
        tf.keras.layers.Dense(24, activation='softmax')])

    print(bi_lstm_model.summary())

    bi_lstm_model.compile(loss='sparse_categorical_crossentropy',
                          optimizer='adam',
                          metrics=['accuracy'])

    return bi_lstm_model
```

F. Knowledge Graph Generation

Beyond the realm of traditional machine learning and deep learning models, our research endeavors extend to the construction of a knowledge graph, introducing a layer of semantic understanding to the dataset. This phase is particularly pivotal in capturing and formalizing the intricate relationships among entities within the textual data. Leveraging Named Entity Recognition (NER), we extract entities representing diseases and symptoms from the symptom descriptions. The identified entities then serve as the foundational elements for the subsequent knowledge graph generation.

The provided Python code in **Algorithm 3** defines a function which trains a Named Entity Recognition (NER) model using the SpaCy library. The function takes as input a dataset ('TRAIN_DATA'), and the 'spacy' and 'DocBin' modules from the SpaCy library. The function begins by creating a blank English language model and initializing a 'DocBin' object. It then iterates over each example in the training data. For each example, it creates a 'Doc' object from the text of the example, and for each entity in the example, it creates a 'Span' object with the start and end indices and label of the entity. These spans are added to the 'Doc' object's entities. The function then saves the data into a spacy format named **training_data.spacy**, which contains all the 'Doc' objects, to disk. Later, it runs a command-line instruction to train the NER model using the saved 'DocBin' object as training data. The command is executed using the 'subprocess.run' function, and the function checks whether the command was executed successfully by checking the return code.

The process involves associating diseases and symptoms to construct triples in the general form of subject-[relation]-object which specifics to symptom-[is-linked-to]-disease Format. Here, the symptoms act as subjects, the relationships denote the manifestation of symptoms, and the diseases form the objects. This structured representation not only encapsulates the semantic relationships within the dataset but also lays the groundwork for a more interpretable understanding of the interplay between diseases and their associated symptoms.

To actualize the knowledge graph, we employ a graph database, Neo4j. This database facilitates the storage and querying of relationships, allowing for efficient traversal and exploration of the constructed knowledge graph. The incorporation of a knowledge graph enriches our research by providing a dynamic framework for uncovering hidden patterns, facilitating targeted queries, and offering a more holistic view of the complex interactions between diseases and symptoms. The resultant knowledge graph serves as a valuable resource, complementing deep learning models and paving the way for a comprehensive and interpretative analysis of our dataset.

Algorithm 3 Construction and Training of Named Entitiy Recognition using Spacy

```
def train_ner(TRAIN_DATA, spacy, DocBin):
    nlp = spacy.blank("en")
    doc_bin = DocBin()

    for training_example in tqdm(TRAIN_DATA):
        text = training_example['text']
        labels = training_example['entities']
        doc = nlp.make_doc(text)
        ents = []

        for start, end, label in labels:
            span = doc.char_span(start, end, label=label,
                                 alignment_mode="contract")

            if span is None:
              print("Skipping entity")
            else:
              ents.append(span)

        filtered_ents = filter_spans(ents)
        doc.ents = filtered_ents
        doc_bin.add(doc)

    doc_bin.to_disk("data/training_data.spacy")

    command = "python -m spacy train config.cfg –output
    ./models/ner-model1 –paths.train ./data/training_data.spacy
    –paths.dev ./data/training_data.spacy"

    result = subprocess.run(command, shell=True,
                            capture_output=True,
                            text=True)

    if result.returncode == 0:
        print("Command executed successfully.")
    else:
        print(f"Error: result.stderr")
```

V. Experimental Evaluation

The implementation details of this experiment involved the use of Python 3.10 as the primary programming language, along with several libraries and frameworks. This includes performing exploratory data analysis using Pandas and NumPy to get insights from data. Later, NLTK and Python regular expressions package named re are used to clean the raw data. Next, Tensorflow's text-to-sequence is used to extract textual features from the symptoms descriptions. Later, two pipelines are implemented, the first one uses CNN Architecture to train and second pipeline uses BiLSTM Architecture for training. Later, the NER model in trained on the symptoms descriptions to extract entities. Finally, the best of the both models are used to along with the NER model to integrate into the Knowledge Graph and visualize them using Neo4J.

The research questions of this experiment were aimed at understanding how symptoms impact on each disease. Specifically, the experiment sought to answer questions such as: What are the strengths and weaknesses of each model architecture concerning the nuances of symptom descriptions? How does the knowledge graph contribute to capturing and leveraging the semantic relationships between diseases and symptoms? To what extent can the constructed knowledge graph assist healthcare professionals in understanding the rationale behind disease predictions based on symptom descriptions? These research questions provided the framework for the experiment and guided the analysis of the results.

Keeping these research questions in mind, this research project began by gathering the data. The data set that is gathered consists of 24 different types of diseases, with each disease having 50 symptom descriptions.

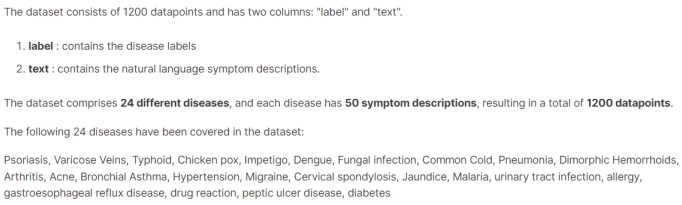

Fig. 5. Data Set Information.

Fig. 5 shows the information of the data that is gathered. It has two columns: The label which describes each disease, the text columns which describes each symptom told by each patient.

The results of the experiment are conducted for each pipeline separately using CNN and BiLSTM and later compared with each other to find out which pipeline did better.

A. Evaluation Metrics

The confusion matrix and classification report is used to assess the performance of the CNN and BiLSTM models. The first metrics which is a confusion matrix is a matrix depicting the number of true positives, true negatives, false positives, and false negatives. It provides a detailed breakdown of model performance, enabling a nuanced understanding of classification errors and strengths.

True Positives (TP) are the instances where the model correctly predicts the positive class, indicating accurate identification of a specific disease. True Negatives (TN) are the instances where the model correctly predicts the negative class, indicating accurate identification of instances that do not belong to a specific disease. False Positives (FP) are the instances where the model incorrectly predicts the positive class, suggesting a misclassification of an instance as belonging to a specific disease when it does not. False Negatives (FN) are the instances where the model incorrectly predicts the negative class, indicating a misclassification of an instance as not belonging to a specific disease when it does.

The second metrics is a classification report. A classification report is a comprehensive summary of various evaluation

metrics for a classification model, providing detailed insights into its performance across different classes. It is particularly useful for understanding the precision, recall, and F1 score for each class, along with macro and weighted averages. The classification report is generated based on the information derived from the confusion matrix. The accuracy, precision, recall and F1 score can be computed using the formula:

$$Accuracy = TP + TN/TP + TN + FP + FN$$

$$Precision = TP/TP + FP$$

$$Recall = TP/TP + FN$$

$$F1 - Score = 2 * (Precision * Recall/Precision + Recall)$$

B. Convolution Neural Network (CNN) Architecture

First, the results of Convolution Neural Network (CNN) model are presented.

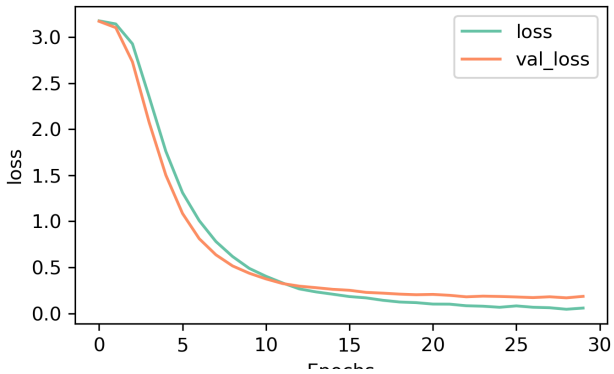

Fig. 6. CNN Training and Validation Loss

Fig. 6 describes a line graph that represents the training loss and validation loss of a model over 30 epochs. There are two lines on the graph: the orange line represents the training loss and the green line represents the validation loss. Both lines show a decreasing trend over time, which indicates that the model is learning and improving its performance with each epoch. Also, the training and validation loss seem to be converging towards the end of the epochs. This suggests that the model might be reaching its optimal performance on the training data. Since the validation loss is decreasing and closely follows the training loss, it's a good indication that the model is not overfitting. Towards the end of the epochs, the lines become smoother, suggesting that the changes in loss are becoming smaller and the model is stabilizing.

Fig. 7 describes a graph which provides several key insights about the model's learning process over 30 epochs. Both the training accuracy (green line) and the validation accuracy (orange line) increase over the 30 epochs. The training accuracy

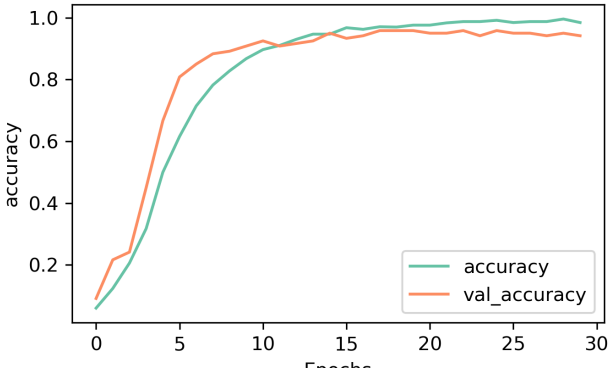

Fig. 7. CNN Training and Validation Accuracy

starts at around 0.0 and increases to around 0.9 at epoch 30. The validation accuracy starts at around 0.4 and increases to around 0.85 at epoch 30. This is a positive sign, indicating that the model is learning and improving its predictions with each epoch. This is a positive sign, indicating that the model is learning and improving its predictions with each epoch. It also indicates that the model is converging well and there's no possible overfitting that occurs in the model.

TABLE I
CNN MODEL PERFORMANCE METRICS ON VALIDATION SET

Disease	Precision	Recall	F1-Score
Acne	1.00	1.00	1.00
Arthritis	1.00	1.00	1.00
Bronchial Asthma	1.00	1.00	1.00
Cervical spondylosis	1.00	1.00	1.00
Chicken pox	1.00	1.00	1.00
Common Cold	1.00	1.00	1.00
Dengue	0.75	1.00	0.86
Dimorphic Hemorrhoids	1.00	1.00	1.00
Fungal infection	1.00	1.00	1.00
Hypertension	1.00	1.00	1.00
Impetigo	1.00	1.00	1.00
Jaundice	1.00	1.00	1.00
Malaria	1.00	1.00	1.00
Migraine	1.00	1.00	1.00
Pneumonia	1.00	1.00	1.00
Psoriasis	1.00	0.75	0.86
Typhoid	1.00	0.67	0.80
Varicose Veins	1.00	0.88	0.93
allergy	1.00	1.83	0.91
diabetes	1.00	1.00	1.00
drug reaction	1.00	1.00	1.00
gastroesophageal reflux disease	0.73	1.00	0.84
peptic ulcer disease	1.00	0.75	0.86
urinary tract infection	1.00	1.00	1.00

The Table I provides a detailed breakdown of the CNN model's performance metrics for each disease. The model achieved an overall accuracy of 0.96, indicating that it correctly predicted the disease 96% of the time across all cases. For most diseases, the model performed exceptionally well, achieving a precision, recall, and F1-score of 1.00. For instance, diseases such as Acne, Arthritis, Bronchial Asthma, and Cervical Spondylosis were predicted with perfect accu-

racy. However, there were some diseases where the model's performance was less than perfect. For example, Typhoid was predicted with a precision of 1.00, recall of 0.67, and F1-score of 0.80. Similarly, Dengue was predicted with a precision of 0.75, but a perfect recall and F1-score of 1.00 and 0.86, respectively.

TABLE II
CNN MODEL PERFORMANCE METRICS ON TEST SET

Disease	Precision	Recall	F1-Score
Acne	1.00	1.00	1.00
Arthritis	1.00	1.00	1.00
Bronchial Asthma	1.00	1.00	1.00
Cervical spondylosis	1.00	1.00	1.00
Chicken pox	1.00	0.83	0.91
Common Cold	1.00	1.00	1.00
Dengue	0.60	0.50	0.55
Dimorphic Hemorrhoids	1.00	1.00	1.00
Fungal infection	1.00	1.00	1.00
Hypertension	1.00	1.00	1.00
Impetigo	0.75	1.00	0.86
Jaundice	1.00	1.00	1.00
Malaria	1.00	1.00	1.00
Migraine	1.00	0.75	0.86
Pneumonia	1.00	1.00	1.00
Psoriasis	1.00	0.50	0.67
Typhoid	0.62	0.83	0.71
Varicose Veins	1.00	1.00	1.00
allergy	1.00	0.83	0.91
diabetes	0.67	1.00	0.80
drug reaction	1.00	1.00	1.00
gastroesophageal reflux disease	0.50	1.00	0.67
peptic ulcer disease	1.00	0.71	0.83
urinary tract infection	1.00	0.83	0.91

The Table II provides a detailed breakdown of the CNN model's performance metrics on the test set. The model achieved an overall accuracy of 0.91. The generalization of the model on the test set is actually good and it does not overfit.

C. Bidirectional Long Short Term Memory (BiLSTM) Architecture

Second, the results of Bidirectional Long Short Term Memory (BiLSTM) model are presented.

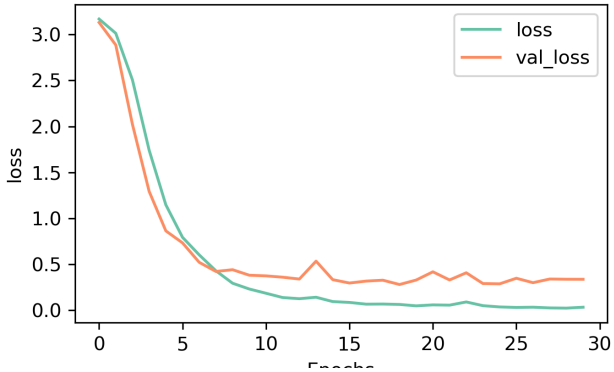

Fig. 8. BiLSTM Training and Validation Loss

Fig. 8 describes a line graph that represents the training loss and validation loss of a model over 30 epochs. There are two lines on the graph: the orange line represents the training loss and the green line represents the validation loss. Both lines show a decreasing trend over time, which indicates that the model is learning and improving its performance with each epoch. Also, the training and validation loss seem to be converging towards the end of the epochs. This suggests that the model might be reaching its optimal performance on the training data. Since the validation loss is decreasing and closely follows the training loss, it's a good indication that the model is not overfitting. Towards the end of the epochs, the lines become smoother, suggesting that the changes in loss are becoming smaller and the model is stabilizing.

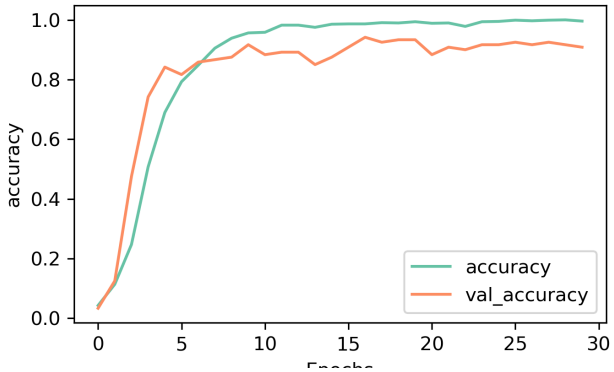

Fig. 9. BiLSTM Training and Validation Accuracy

Fig. 9 describes a graph which provides several key insights about the model's learning process over 30 epochs. Both the training accuracy (green line) and the validation accuracy (orange line) increase over the 30 epochs. The training accuracy starts at around 0.0 and increases to close to 1.0 at epoch 30. The validation accuracy starts at around 0.1 and increases to around 0.84 at epoch 30.

The Table III provides a detailed breakdown of the BiLSTM model's performance metrics for each disease. The model achieved an overall accuracy of 0.92, indicating that it correctly predicted the disease 92% of the time across all cases. For most diseases, the model performed exceptionally well, achieving a precision, recall, and F1-score of 1.00. For instance, diseases such as Acne, Arthritis, and Bronchial Asthma were predicted with perfect accuracy. However, there were some diseases where the model's performance was less than perfect. For example, Chicken Pox was predicted with a precision, recall, and F1-score of 0.83. Similarly, Dengue was predicted with a f1-score of 0.67, but a perfect precision and recall of 1.00 and 0.50, respectively.

The Table IV provides a detailed breakdown of the BiLSTM model's performance metrics on the test set. The model achieved an overall accuracy of 0.88. The generalization of the model on the test set is actually good and it does not overfit but it didn't do well like the CNN Model.

TABLE III
BILSTM MODEL PERFORMANCE METRICS ON VALIDATION SET

Disease	Precision	Recall	F1-Score
Acne	1.00	1.00	1.00
Arthritis	1.00	1.00	1.00
Bronchial Asthma	1.00	1.00	1.00
Cervical spondylosis	0.62	1.00	0.77
Chicken pox	0.83	0.83	0.83
Common Cold	0.86	1.00	0.92
Dengue	1.00	0.50	0.67
Dimorphic Hemorrhoids	1.00	1.00	1.00
Fungal infection	0.50	1.00	0.67
Hypertension	1.00	1.00	1.00
Impetigo	1.00	0.75	0.86
Jaundice	1.00	1.00	1.00
Malaria	1.00	1.00	1.00
Migraine	1.00	1.00	1.00
Pneumonia	1.00	1.00	1.00
Psoriasis	1.00	0.50	0.67
Typhoid	0.75	1.00	0.86
Varicose Veins	0.88	0.88	0.88
allergy	1.00	0.83	0.91
diabetes	0.80	1.00	0.89
drug reaction	1.00	1.00	1.00
gastroesophageal reflux disease	0.60	1.00	0.75
peptic ulcer disease	0.83	0.71	0.77
urinary tract infection	1.00	1.00	1.00

TABLE IV
BILSTM MODEL PERFORMANCE METRICS ON TEST SET

Disease	Precision	Recall	F1-Score
Acne	1.00	1.00	1.00
Arthritis	0.88	1.00	0.93
Bronchial Asthma	1.00	1.00	1.00
Cervical spondylosis	0.67	1.00	0.80
Chicken pox	0.67	0.67	0.67
Common Cold	1.00	0.83	0.91
Dengue	0.60	0.50	0.55
Dimorphic Hemorrhoids	1.00	1.00	1.00
Fungal infection	0.92	1.00	0.96
Hypertension	1.00	1.00	1.00
Impetigo	1.00	1.00	1.00
Jaundice	1.00	1.00	1.00
Malaria	1.00	1.00	1.00
Migraine	1.00	0.75	0.86
Pneumonia	1.00	1.00	1.00
Psoriasis	0.50	0.50	0.50
Typhoid	0.80	0.67	0.73
Varicose Veins	1.00	1.00	1.00
allergy	0.56	0.83	0.67
diabetes	1.00	0.75	0.86
drug reaction	0.00	0.00	0.00
gastroesophageal reflux disease	1.00	1.00	1.00
peptic ulcer disease	0.86	0.86	0.86
urinary tract infection	0.83	0.83	0.83

D. Named Entity Recognition (NER)

The Table V is the details of an NER model which is trained on 4000 samples with 57 epochs. The model's performance metrics (ENTS_F, ENTS_P, ENTS_R, SCORE) generally increase with each epoch. This suggests that the model is learning and improving its performance over time. Also, The LOSS NER metric generally decreases with each epoch. This is another indication that the model is learning and improving its performance, as a lower loss typically means the model's predictions are getting closer to the actual values. Towards the end of the training (from epoch 21 onwards), the model's performance metrics reach their maximum value of 100, and the LOSS NER metric decreases to a very low value. This suggests that the model has converged, i.e., further training is unlikely to improve the model's performance.

TABLE V
NAMED ENTITY RECOGNITION (NER) MODEL PERFORMANCE

E	#	LOSS NER	ENTS_F	ENTS_P	ENTS_R
0	0	40.34	0.00	0.00	0.00
0	200	2056.15	71.64	77.99	66.25
0	400	721.47	85.55	88.31	82.95
1	600	721.48	91.59	92.98	90.24
2	800	543.54	95.69	97.10	94.32
3	1000	506.91	96.88	97.37	96.40
4	1200	496.32	97.84	98.42	97.27
6	1400	394.02	98.76	98.95	98.57
8	1600	311.60	99.44	99.57	99.31
10	1800	258.72	99.76	99.83	99.70
13	2000	181.58	99.85	99.91	99.78
17	2200	103.09	99.98	99.96	100.00
21	2400	130.01	100.00	100.00	100.00
26	2600	63.07	99.96	99.96	99.96
30	2800	52.05	100.00	100.00	100.00
35	3000	27.21	100.00	100.00	100.00
39	3200	53.07	100.00	100.00	100.00
43	3400	40.22	99.80	99.78	99.83
48	3600	60.83	100.00	100.00	100.00
52	3800	75.92	99.96	99.91	100.00
57	4000	107.58	100.00	100.00	100.00

E. Result and Analysis

First, the comparision table between CNN and BiLSTM model are presented to pick the best model and integrate it with Knowledge Graph.

TABLE VI
CNN VS. BILSTM

Model	Accuracy	Precision	Recall	F1-Score
CNN	0.91	0.92	0.90	0.90
BiLSTM	0.88	0.84	0.84	0.83

Insights from the Table V represents that the CNN model performed well compared to the BiLSTM model by 2% accuracy. Finally, the insights from the Knowledge Graph Generated, visualized using Neo4J and are presented.

Fig 10. represents the common disease that are associated for a given set of symptoms. Here, we took two symptoms "fever" and "headache" to get diseases that may occur which tell use that these two symptoms are common across number of diseases like Dengue, Malaria, Chicken Pox etc. For example, When a patient presents early-stage symptoms like "fever" and "headache", a doctor can refer to this graph to explore potential diseases that are associated with these symptoms. By examining the other symptoms linked to each potential disease in the graph, the doctor can cross-check with the patient's symptoms. This could help narrow down the possible diseases and guide further diagnostic tests or treatments.

This approach could be particularly useful for diagnosing complex or rare diseases that might be overlooked in

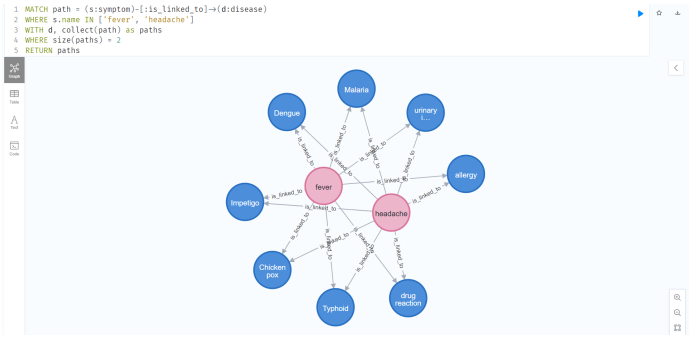

Fig. 10. Uncovering Common Diseases Associated with a Set of Symptoms

traditional diagnostic approaches. It also allows for a more holistic view of the patient's condition, taking into account the interconnected nature of symptoms and diseases.

Fig 11. represents most common symptoms that occur for a given disease. Extending the previously mentioned example, then doctor can then look into a graph like this for each disease and the cross-check with the patient symptoms to identify the potential disease. This could save huge amount of time and increase the early diagnosis of the disease and can take immediate action on the patients health.

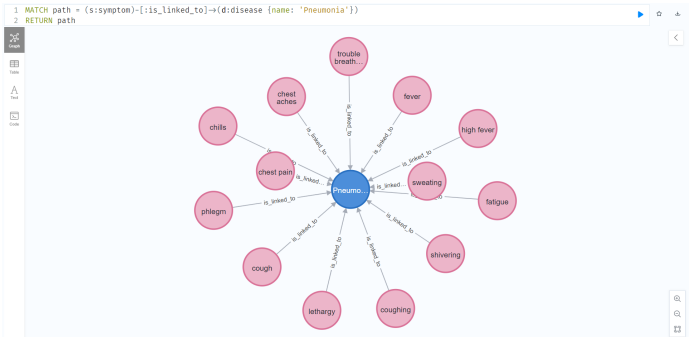

Fig. 11. Identifying Predominant Symptoms for a Specific Disease

Fig 12. represents a table which shows rare symptoms that occur for multiple diseases. For example, if a patient comes tells the doctor with the rare symptoms that he never heard of, a graph like this can help him to determine the specific disease.

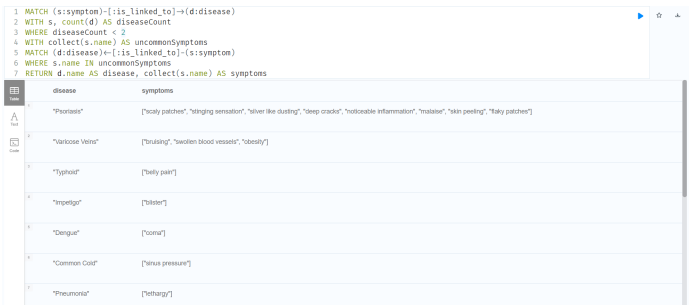

Fig. 12. Exploring Rare Symptom-Disease Associations

VI. LIMITATIONS

While SymptoGraph offers a promising avenue for advancing symptom-based disease prediction, several limitations need to be acknowledged to ensure a nuanced interpretation of the findings and guide future research endeavors. One of the limitation is the dataset encompasses 24 diseases, the limitation lies in its exclusivity to these specific diseases. The model's applicability to a broader range of diseases outside this set may require additional data and validation to ensure comprehensive coverage.

Another limitation is, the dataset, comprising 1200 datapoints distributed across 24 diseases, may be deemed relatively modest for training robust deep learning models like BiLSTM and CNN. This limitation has implications for the models' generalization capabilities, particularly for diseases with limited instances.

Regarding the Named Entity Recognition (NER), it's efficiency in accurately identifying disease and symptom entities is subject to the intricacies of symptom descriptions. Ambiguities, variations in language, and potential symptom overlap across diseases may pose challenges to precise entity extraction.

Finally, SymptoGraph, developed and trained on the 24 diseases in the dataset, may face challenges when applied to novel or rare diseases. Generalizing the approach to a broader spectrum of diseases may require additional training on diverse datasets to ensure adaptability.

VII. CONCLUSION

In conclusion, SymptoGraph marks a notable advancement in symptom-based disease prediction, employing sophisticated deep learning models and knowledge graph integration at the intersection of Natural Language Processing and healthcare. While the comparative analysis of BiLSTM and CNN models, along with the incorporation of a knowledge graph, enhances interpretability, acknowledging limitations is paramount. Challenges include the dataset's size, the inherent ambiguity in symptom descriptions, and ethical considerations. Addressing these limitations is crucial for refining SymptoGraph's robustness, with opportunities for dataset augmentation, improved entity recognition, and a deeper exploration of ethical implications. Despite current constraints, SymptoGraph lays a foundation for future innovations in leveraging NLP for nuanced disease prediction and contributing to more informed healthcare decision-making.

REFERENCES

[1] P. Hema, N. Sunny, R. Venkata Naganjani and A. Darbha, "Disease Prediction using Symptoms based on Machine Learning Algorithms," 2022 International Conference on Breakthrough in Heuristics And Reciprocation of Advanced Technologies (BHARAT), Visakhapatnam, India, 2022, pp. 49-54, doi:10.1109/BHARAT53139.2022.00021.

[2] Alanazi R. Identification and Prediction of Chronic Diseases Using Machine Learning Approach. J Healthc Eng. 2022 Feb 25;2022:2826127. doi: 10.1155/2022/2826127. PMID: 35251563; PMCID: PMC8896926.

[3] Keniya, Rinkal and Khakharia, Aman and Shah, Vruddhi and Gada, Vrushabh and Manjalkar, Ruchi and Thaker, Tirth and Warang, Mahesh and Mehendale, Ninad, Disease Prediction From Various Symptoms Using Machine Learning (July 27, 2020). Available at SSRN: https://ssrn.com/abstract=3661426 or http://dx.doi.org/10.2139/ssrn.3661426

[4] Divya, A., Deepika, B., Durga Akhila, C.H. et al. Disease Prediction Based on Symptoms Given by User Using Machine Learning. SN COMPUT. SCI. 3, 504 (2022). https://doi.org/10.1007/s42979-022-01399-0

[5] V. Chaurasia and S. Pal, "Data mining approach to detect heart diseases", International Journal of Advanced Computer Science and Information Technology, Vol.2, No.4, pp.56-66, 2014.

[6] K. Vembandasamy, R. Sasipriya, and E. Deepa, "Heart Diseases Detection Using Naive Bayes Algorithm", IJISET-International Journal of Innovative Science, Engineering Technology, Vol.2, pp.441-444, 2015.

[7] Khourdifi, Youness Bahaj, Mohamed. (2019). Heart Disease Prediction and Classification Using Machine Learning Algorithms Optimized by Particle Swarm Optimization and Ant Colony Optimization. International Journal of Intelligent Engineering and Systems. 12. 10.22266/ijies2019.0228.24.

[8] Chae S, Kwon S, Lee D. Predicting Infectious Disease Using Deep Learning and Big Data. Int J Environ Res Public Health. 2018 Jul 27;15(8):1596. doi: 10.3390/ijerph15081596. PMID: 30060525; PMCID: PMC6121625.

[9] Das, Subhalaxmi Pradhan, Sateesh Mishra, Sujogya Pradhan, Sipali Pattnaik, Pradyumna. (2022). DIAGNOSIS OF CARDIAC PROBLEM USING ROUGH SET THEORY AND MACHINE LEARNING. Indian Journal of Computer Science and Engineering. 13. 1112-1131. 10.21817/indjcse/2022/v13i4/221304070.