NLP approach for distributed health data management

Kashmira Lokhande, Varun Bhandari, Navani Udgaonkar, Aastha Chaudhary
Department of Computer Engineering and Computer Science
California State University Long Beach
Long Beach, USA
{kashmiraanant.lokhande01, varun.bhandari01, navani.udgaonkar01, aastha.chaudhary01}@student.csulb.edu

***Abstract*—**
In today's world, data is expanding by the day, especially in the healthcare arena, millions of patient records are created in both organized and unstructured formats. Electronic Health Records (EHR) are crucial in the development of digital health data. If this massive amount of data is not correctly managed, it can result in data loss, inaccuracies, and data overloading, all of which can have a severe impact on patient safety. Distributed and self-organizing algorithms are required to manage such a massive volume of data. We use Natural Language Processing (NLP) in this work, which is a field of artificial intelligence that examines the interaction between machines and humans using natural language and defines the basic stages of NLP. The semantic context of documents may be captured using NLP by modeling them in dense vectors known as word embeddings.

Health records may be represented as vectors by adopting the Doc2Vec paradigm. This method is used to create a vectorized representation of a collection of words as a whole. It uses the word2vec technique to generate vectors from words. The distributed system functions autonomously and locally, forming a sorted network. As a consequence, resources are used faster and more efficiently. As a consequence, user requests can successfully reach the server with the needed information.

***Index Terms*—Natural Language Processing, Distributed Computing, Electronic Health Records, Word Embedding, Word2Vec**

I. Introduction

The major portion of data is now produced by a variety of infrastructures that are geographically dispersed and have a dynamic source of information. In the field of health, the increased usage of digital infrastructures and data, such as electronic health records (EHRs) [5] as shown in **figure1**, has resulted in the formation of an exceptional amount of recorded data. Several management concerns have developed because of the information overload, potentially having a severe impact on clinical performance, such as omission errors, delays, and overall patient safety It is necessary to manage massive volumes of health data collected on a daily basis, as well as properly handle medical activities and processes. One of today's primary issues is developing a sufficient system to support all of the requirements of the health environment. Mechanisms for identifying valuable patient EHRs in a shared health system have been proposed, however many of them are dependent on a centralized approach and hence can only be dealt with in small-scale networks. Indeed, centralized methods have their limitations in big and dynamic systems, and it is better to build new and distributed ways. Several peer-to-peer technologies have been developed to organize and find resources in distributed systems due to their inherent resilience and scalability.

The goal of these tactics is to assist users in locating resources or services, whether hardware or software, that meet their requirements. In P2P networks, resources and services are usually represented by a syntactical/ontological description of their properties, known as metadata, and the discovery service must find the information that possesses the required attributes. In this paper's proposed method, Word2Vec is a conversion paradigm for words into vectors. Word2Vec is a two-layer artificial neural network used to analyze text in order to discover links between words a corpus of text Word2Vec takes a huge corpus as input text and generates a three-dimensional space (usually of several hundred dimensions), with each distinct word in the corpus being allocated a vector in the space.

Natural Language (NLP) is a subfield of artificial intelligence (AI) that examines how people and machines converse via natural language. The "word embedding" approach, which can identify various degrees of similarity between words, is one of the most significant recent advances in language processing. To develop the model of word connections, a specific grouping of text or documents, known as the training corpus, is given into the Word2Vec process. Word2Vec generates a vocabulary from a corpus and learns word representations by training a three-level neural network. Word2Vec presents two models: I Continuous Bag of Words (CBOW), which learns representations by predicting the target word based on its context words; and (ii) Skip-gram, which learns representations by predicting each context word based on the target word. To build word embeddings from a huge corpus of unlabeled data, one must select one of the designs and define values for key parameters such as embedding size, context size, and the

minimal frequency for a word to be included in the word vocabulary. NLP has its roots in linguistics and has been around for over 50 years. Among the uses are medical research, search engines, and corporate intelligence. Natural language processing (NLP) refers to a computer program's capacity to recognize human-spoken and written natural language. It's a feature of artificial intelligence.

NLP combines linguistics and computer science to comprehend language structure and norms and to develop understandable models for breaking down and isolating crucial aspects from text and speech. The purpose is to design a system that can comprehend the contents of the documents, as well as the contextual aspects of the language, included within them. Natural language processing (NLP) is the ability of computers to comprehend natural language in the same manner that humans do. Natural language processing, whether spoken or written, employs artificial intelligence to analyze and interpret real-world data in a form that computers can comprehend.

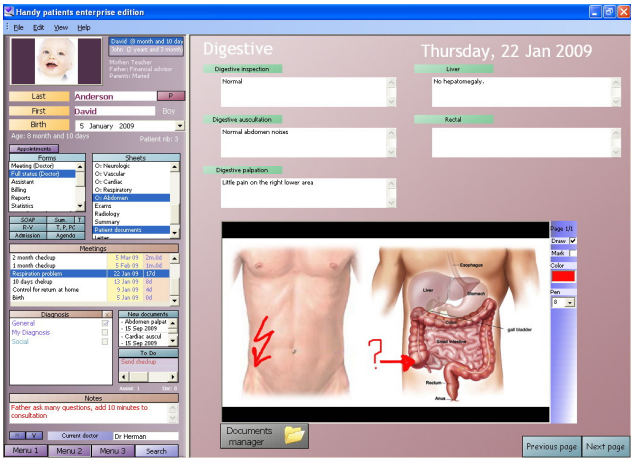

Fig. 1. **Electronic Health Records**

A. Phases of NLP

There are two main phases of Natural language processing that include: data preprocessing and algorithm development. Data Preprocessing includes preparing and "cleaning" text data so that machines can examine it. Preprocessing converts data into usable form and identifies language elements that an algorithm can use. An algorithm is created to process the data once it has been preprocessed.

There are many different types of natural language processing algorithms, but two of the most prevalent are: A system based on rules, that employs language rules that have been meticulously established. This method was pioneered in the development of natural language processing and is still in use today. Statistical techniques are used by the algorithms of a machine learning-based system. They learn how to do tasks by using training data and adjusting when fresh data is evaluated. Language processing algorithms use a combination of machine learning, deep learning, and neural networks to change their own rules through repeated processing and learning.

System based on rules, the system employs language rules that have been meticulously established. This method was pioneered in the development of natural language processing and is still in use today. Machine learning-based system, the algorithms employ statistical methodologies. They learn to complete tasks using training data and adapt when fresh data is processed. Natural language processing algorithms use a combination of machine learning, deep learning, and neural networks to modify their own rules through repeated processing and learning.

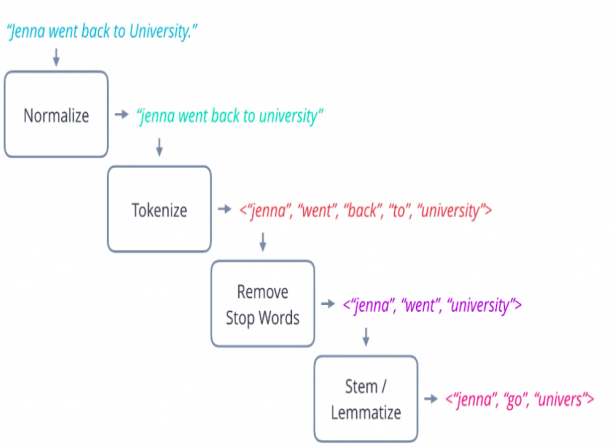

Fig. 2. **Types of Data Preprocessing**

B. Data Preprocessing and its types

Data preprocessing is the process of arranging and "cleaning" textual information so that computers can manage it. Data that has been turned into a useful format, with a concentration on textual components that an algorithm may use, is referred to as preprocessing. This can be accomplished in a variety of ways, including: The act of breaking down a whole text into its component phrases is known as segmentation. This may be accomplished by segmenting the material along punctuation lines like full stops and commas. Tokenization, For the algorithm to understand segmented phrases, we get the words in a phrase and explain them individually to our algorithm; hence, we split down our sentences into their component words and store them. We may shorten the learning process by eliminating non-essential phrases that don't bring any value to our messaging. The Stop-Word Removal method is used in this case. These words, which include – are and – the, are known as stop words.

The process of transforming a term into its root form is known as stemming and lemmatization. For instance, in the line "The dog barked," the algorithm would determine that the root of the word "barked" is "bark." This would be handy if a user was looking for all instances of the word bark, as well as all of its conjugations, in a document. Even if the letters are different, the algorithm recognizes that they are basically the same term. We now teach the machine the notion of nouns, verbs, articles, and other components of speech by adding these tags to our machine; this is known

as speech tagging. Named entity recognition is a technique for categorizing sentences. The goal of NER is to allow the computer to extract entities such as persons, places, items, locations, monetary numbers, and more.

II. Word Embedding Techniques

Word embedding is defined in NLP to explain how words are represented for text analysis, which is often in the form of a vector of real numbers that preserves the meaning of words and predicts that words which are nearby in the vector space will have similar meanings. Word embedding have three goals: the first is to minimize dimensionality, the second is to utilize a word to anticipate the words around it, and the third is to capture inter word semantics. They're made by applying a set of language modeling and feature learning algorithms to map words or phrases from a dictionary to real-number vectors. Word embeddings are also useful for avoiding the dimensionality curse, which is a common difficulty in artificial intelligence. The unique identifiers encoding the words generate dispersed data, isolated points in a huge sparse representation without word embedding. With word embedding, on the other hand, the dimensionality of the space is much reduced while the amount of semantic information is greatly increased. It is easier for a computer to do different mathematical operations like matrix factorization, dot product, and other operations that are required to apply shallow and deep learning approaches with such numerical characteristics. There are many word embedding techniques from which we have reviewed Bag-Of-Words, TF-IDF, Word2Vec and Doc2Vec.

A. Bag-Of-Words

A bag of words is a basic and widely used technique for extracting features from text. The bag of word model analyzes the text to determine the number of times each word appears in the sentence. Vectorization is another term for this. The model is just interested in the frequency of words in the text and is unconcerned with their order. It can be used for natural language processing, document classification, and information retrieval from documents. Cleaning raw text, tokenization, vocabulary development, and vector generation are all part of the normal BOW operation. The following are the steps to make a BOW:

- Tokenize the text so that it may be broken down into sentences.
- Tokenize phrases to make them into words.
- Remove any punctuation or words that aren't necessary.
- Lower the text by converting the words.
- Make a word frequency distribution chart.

For example, there are two statements:

Statement1: "The cat in the hat"

Statement2: "The cat sat on wall"

The vocabulary of these two statements is:

Vocabulary: ["The", "cat", "in", "hat", "sat", "on", "wall "]

This vocabulary generates feature vectors with a length of 7. These two statements are represented by a bag-of-words feature:

BOW(statement1): [2,1,1,1,0,0,0]

BOW(statement2): [1,1,0,0,1,1,1]

This approach is used to generate the bag-of-words representation of each document in the dataset. If our dataset contains a large number of unique terms, some of them are unlikely to be used frequently, which is the case most of the time. As a result, we chose N of the most common words and created a feature vector with the dimensions Nx1. These feature vectors are then applied to any machine learning task.

B. Term Frequency-Inverse Document Frequency

Another technique to represent a document based on its words is the Term Frequency-Inverse Document Frequency. It assigns weights to terms based on their TF-IDF relevance rather than their frequency. It's a statistical metric for determining how relevant words in a corpus are to the document. The weight is proportional to the number of times the term appears in the manuscript. It also fluctuates depending on the word's frequency in the corpus.

Term Frequency (TF) is equal to the number of times a word appears in a document, divided by the total number of words in that document and Inverse Document Frequency (IDF) is calculated by the logarithm of the number of documents in the corpus divided by the number of documents where the specific term appears.Thus TF informs us about the frequency with which a term appears in a document, while IDF informs us about the relative rarity of a term in the collection of documents. By multiplying these figures together, we can get our final TF-IDF value. The more important or relevant the term is, the higher the TF-IDF score; the less relevant the term is, the lower the TF-IDF number.

Machine learning, information retrieval, and text summarization/keyword extraction are the three main applications for TF-IDF. The most significant advantages of TF-IDF are easy to calculate, inexpensive to compute, and a good starting point for similarity computations. It's worth noting that TF-IDF can't help carry semantic meaning. It assesses the importance of the words based on how it weights them, but it is unable to deduce the contexts of the phrases and so grasp importance in this manner.

C. Word2Vec

Word2Vec is one of the most effective methods for representing words. It's a word embedding training algorithm that Google came up with. It is based on the distributional hypothesis, which asserts that the words that are frequently adjacent to each other are semantically similar. This helps in the mapping of semantically similar words to embedding vectors that are geometrically close. The words are plotted in a multi-dimensional vector space, with similar words clustered together. A word's context is determined by the words surrounding it.

This technique involves training a basic neural network with a one hidden layer to complete a fictitious task. This network isn't dependent on the outcomes of the task on which it was trained. Instead, the major goal is to learn a representation of

the input data, which are actually the 'word vectors' derived from the hidden layer's learnt weights.

The model's performance in predicting words is determined by how many times it sees these words in the same context across the dataset. During the training phase, additional words and context co-occurrences are added to the hidden representation, allowing the model to make more future successful predictions and resulting in a stronger representation of word and context in the vector space. Skip gram is slower than CBOW, but it is more accurate when dealing with rare words.

Word2Vec can generate a distributed representation of words using one of two model architectures: continuous bag-of-words (CBOW) or continuous skip-gram.

- CBOW: A window of surrounding context words is used to forecast the current word. The order of the words in the context has no bearing on the outcome (bag-of-words assumption). If you give the trained network the input words "Soccer," "NBA," "Game," and "Tennis," the output probabilities for words like "Player" and "Tennis" will be substantially higher than for unrelated words like "cheese" and "elections."
- Continuous skip-gram: This model predicts the surrounding window of context words based on the current word. Nearby context words are given more weight in the skip-gram architecture than more distant context words. The output probabilities will reflect the likelihood of each vocabulary term being found near our input word. If you give the trained network the input term "Europe," the output probabilities for words like "Belgium" and "Continent" will be substantially greater than for unrelated words like "fruits" and "cats."

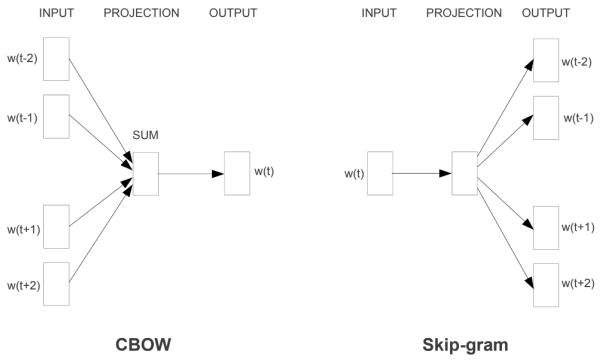

Fig. 3. **Word2Vec Models**

The size of the context window in both models restricts the number of words that can be included as context words for a particular word before and after it. For each observed word in the phrase, a window size of three will contain the three words to the left and three words to the right as context. As more word-context combinations need to be trained, increasing the window size lengthens the training time. It's also possible that it'll record context words that aren't related to the current term. When the number of context words is reduced, it is easier to record relationships between words and stop words, which is undesirable.

This word embedding method can capture a variety of degrees of similarity between words. Patterns like "Man is to Woman as Brother is to Sister" can be created by performing algebraic operations on the vector representations of these words, with the vector representation of "Brother" - "Man" + "Woman" producing the closest result to the vector representation of "Sister" in the model. Such links can be produced for both semantic (such as Country–Capital) and syntactic (e.g. present tense–past tense) relations.

D. Doc2Vec

Another extensively used technique is Doc2Vec, which makes an embedding of a document regardless of its length. In contrast to the Word2Vec model, the Doc2Vec model is used to construct a vectorised representation of a set of words regarded as a single unit. [4] It calculates more than just the average of the words in the sentence. For each document in the corpus, it generates a feature vector. It is based on Word2Vec, with the exception of adding another vector to the input, namely the paragraph ID. This embedding technique can use one of two model architectures: PV-DM (Paragraph Vector in Distributed Memory) or PV-DBW (Paragraph Vector in Distributed Bag of Words) (PV-DBOW).

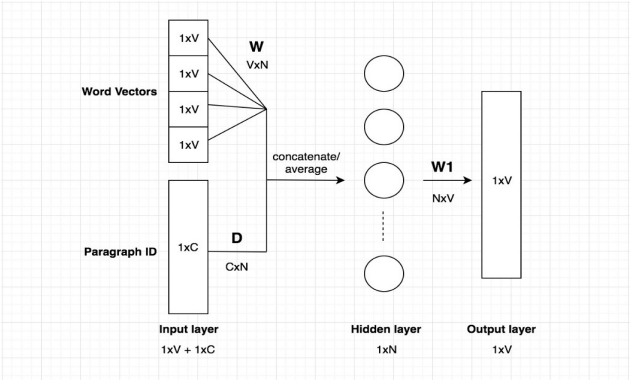

Fig. 4. **Doc2Vec Model**

Instead of merely using surrounding words to forecast the term, we also incorporated a document-unique feature vector. When the word vectors W are trained, the document vector D is also trained, and it holds a numeric representation of the document at the end of the training. Word vectors and document Id vectors are used as inputs. A one-hot vector of the dimension 1xV is referred to as a vector. The document Id vector is 1xC in size, where C is the total number of documents. The buried layer's weight matrix W has a size of NxV. The weight matrix D of the hidden layer has the dimension CxN.

III. METHODS

A. Dataset

For our data set, we have considered various test reports. These test reports have been collected from pubic hospitals

and clinics which were stored in Electronic Health Records (EHR)1. EHR is an electronic medical records that contains the results of clinical administrative encounters between physician and a patient. It is an electronic version of patients test reports and medical history that is maintained over time, and may include clinical data relevant to that persons care under a particular hospital/clinic, including demographics, progress reports, symptoms, medications, vital signs, past medical history, immunizations, laboratory data and radiology reports. The EHR can automate the access to information and it has the potential to streamline the clinician's workflow.

The EHR also has the ability to support activities directly or indirectly through various interfaces, including evidence-based decision support, quality management, and outcomes reporting. These reports mainly contain information about the patient, diagnosis, complaints, laboratory and instrumental test results, and treatments. To identify the most frequent complaints and diagnosis from patients reports, diagnosis and complaints sections of the records were analyzed.

As we are building a prototype of our model, we have considered a small dataset. So here we have used twenty three labels and are given input to the model. These labels are a collection of commonly occurring disorders in the patients' reports. Labels are considered as different classes which are used to categorize the patients; disorders when an unlearned data is given as input. Twenty three entities were comprised into a list for further extraction. The data set was divided into semi-automatic and manual mode to simplify the use and to organize the data set in a simple manner which is easy to read and understand. In semi-automatic mode, first the entities are searched by using the specific keywords and then the user will accept or reject the identified example.Entities were searched as keywords in semi automatic mode and then the users were asked to accept r reject the identified keyword. Manual mode was used to avoid missing of some complicated entities. This mode was implemented using manual labelling. User can select the desired keyword and mark it as corresponding entity and this marked keyword was put into data set by using semi-automatic and manual modes.

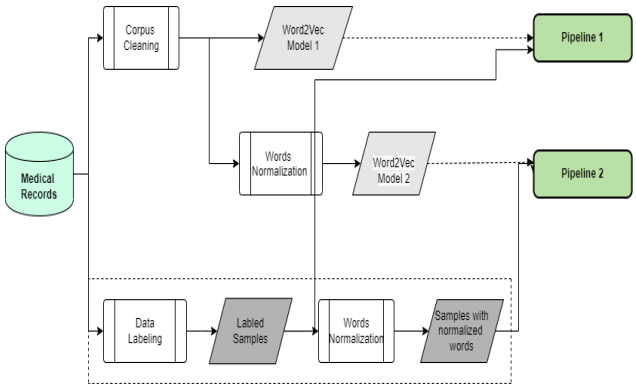

Fig. 5. **Word embedding and preprocessing of samples**

B. Word Embedding Models and Pipelines

Here we have used two different word embedding models. Word embedding is representing a learned text where the words with similar meanings or synonymous words are represented similarly.For a predefined vocabulary which is of fixed size from a corpus of text, the word embedding model learns the areal valued vector. Such two models as shown in the figure 3 were trained on our text corpora. First we gathered all medical records motioned above and these records were given as input to our model. [2] Then these records were preprocessed that is, all the records were cleaned by removing all punctuation, then all non alphabetic tokes were filtered out and these tokens were converted into lowercase.Using this cleaned corpus, Word2Vec model 1 was trained. Word2Vec model is an natural language processing technique, its algorithm uses neural network model to learn word associations from large corpus of text.

After training the model, it is able to identify words with similar meaning or it is also able to suggest some relevant and additional words for partial sentences. As the name says, it represents every word with a particular set of numbers which are known as vectors. The data which was given input to Word2Vec Model 1 was not normalized and it was in its original form. The combination of pre-processing techniques and Word2Vec Model 1 is called Pipeline 1. Here pipeline defines how samples and embedding model from data set were represented to be fed to classifier.

On the other hand for Word2Vec Model 2, after the data is pre processed, word normalization technique is used. Word normalization is used to reduce the randomness and to bring the text closer to a predefined standard. It is a process that converts list of words to a more uniform sequence. By converting the words to a standard format, other operations are able to work with the data and will not deal with issues that might compromise the process. For example, converting all words to lowercase will make the word search process much easier. This helps to reduce the amount of words by reducing the repetitive words and therefore it improves the efficiency. Pipeline 2 in contrast with pipeline 1 includes normalization for embedding model and for samples in dataset. Data from every pipeline was fed to different types of neural networks which were mainly considered of multi-layer perceptron and convolutional neural network [3] [1].

IV. Experiments

A. Experimental Analysis

Natural language text processing can be divided into two categories: classical methods based on language norms and domain ontologies, and machine learning methods. In natural language processing tasks such as data extractions and named entity recognition, machine learning methods produce excellent results. A multi-layer perceptron (MLP) is a type of artificial neural network that consists of an input layer for receiving data, an output layer for making a judgment or prediction based on the input, and hidden layers. This type

of network is distinguished by the fact that each neuron in one layer is connected to every other neuron in the next layer. These layers are also known as dense or fully connected layers. Training, like any other supervised learning technique, entails modifying the model's parameters, particularly the weights and biases, to reduce error.

Convolutional neural networks (CNN) for text classification is one of the successful ways. In image recognition, CNNs have proved to be superior, and this neural network architecture can be used to vector-represented text. Convolutional neural networks (CNNs) have a more sophisticated structure than fully-connected networks. In image classification, speech recognition, and phrase classification, CNNs have attained state-of-the-art performance. In contrast to fully-connected layers, CNN convolution layers are only connected to a tiny portion of the previous layer and have a substantially reduced number of parameters. Technically, this allows us to employ deeper models that use the same amount of memory but perform better. The convolutional layer, pooling layer, and fully-connected layer are the three types of layers necessary to build a CNN. Convolutional layer parameters are a collection of filters that can be learned. Each filter computes a convolutional function by sliding across the input data, which is represented as a matrix. As a result, we have an activation map that shows the filter responses at every spatial position. The filters usually change to activate themselves when they receive an input during the training phase, which can be a property of desirable output. To minimize the size of the spatial representation, reduce computation, and avoid overfitting, pooling layers are placed between convolutional layers. Pooling merges the outputs of various neuron clusters into a single neuron that represents the cluster's greatest value. The same layers that were discussed in the NLP section are used in CNN. It returns the projected class after making the final prediction.

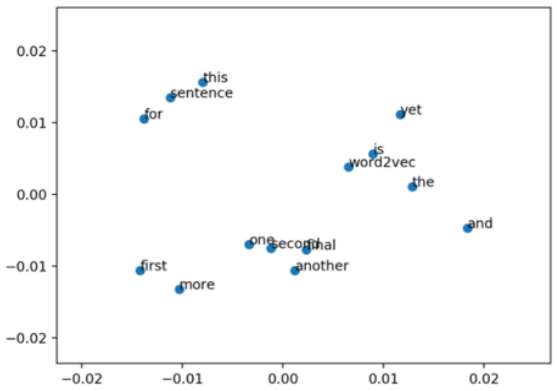

Fig. 6. **Vector Representation of Words**

As a part of the word to vector model, we have first tried to train an inbuilt Word2Vec model from genism module with a few sample electronic health records which were gathered from different hospitals which mainly constituted of the patients medical records, treatment, symptoms and the prescription which was advised to them by the doctor. Once the model was trained then we have used this model as a pre train detector to the new model which has now been created to perform the word embedding algorithm and determine key words i.e. the words which were served to the model as labeled data and which were present in the test data so as to identify the disorder present in the patient. The word embedding algorithm is mainly used here, to identify the words present in the patients report and segregate his report accordingly in the system.In order to segregate the different diseases/disorders present in the patients reports we have used 22 different disorders as the labeled data. Please find a snippet of the labelled data in Fig. 7.

```
0          Asthma with status asthmaticus (disorder)
1     Varicose veins of lower extremity (disorder)
2                              Gastritis (disorder)
3                            Hemorrhoids (disorder)
4                                Headache (finding)
5                          Lightheadedness (finding)
6          Benign essential hypertension (disorder)
7                                Fatigue (finding)
8                       Disorder of liver (disorder)
9                           Hiatal hernia (disorder)
10                       Cholecystitis (disorder)
11                      Pyelonephritis (disorder)
12             Diabetes mellitus type 2 (disorder)
13       Chronic ischemic heart disease (disorder)
14         Body mass index 30+ - obesity (finding)
15        Calculus of kidney and ureter (disorder)
16          Scoliosis of thoracic spine (disorder)
17                Chronic pancreatitis (disorder)
18              Duodenal ulcer disease (disorder)
19                      Chronic anemia (disorder)
20             Edema of lower extremity (finding)
21                Chronic kidney disease (disorder)
```

Fig. 7. **Labeled Data**

Once the labeled data is stored in the system, we are then encoding and categorizing the words present in the test document using label encoder where the words are first categorized and then the corresponding values are then encoded with binary values. In order to encode the words, they are first indexed i.e. each word is being segregated and a separate index value is being assigned to it. In the process, all the stop words are removed from the file. Also, the words are being normalized and only the new words which have not been appeared before and being promoted to the next step in the process. The encoded values are then being passed to the convolution neural network so formed. Once the values are being processed and predicted by the model then they are again decoded and present in the form of the normal words which are readable by humans i.e. in natural language.The indexes assigned the words appear as in fig 8.

Once the words are indexed and their corresponding values are encoded. Then they are served to the model as training, validation and test data and once the data is trained, then we are trying to identify the accuracy of the prediction of the model. The accuracy depends on how well the model is being trained i.e. how many words have been served to the model as the training data and how many times the model is trained with those. So depending on how well the model has learned the words, it is able to predict the new ones and classify them.

```
inx 0
the
inx 1
of
inx 2
in
inx 3
is
inx 4
a
inx 5
/
inx 6
not
inx 7
-
inx 8
and
```

Fig. 8. **Indexed Values**

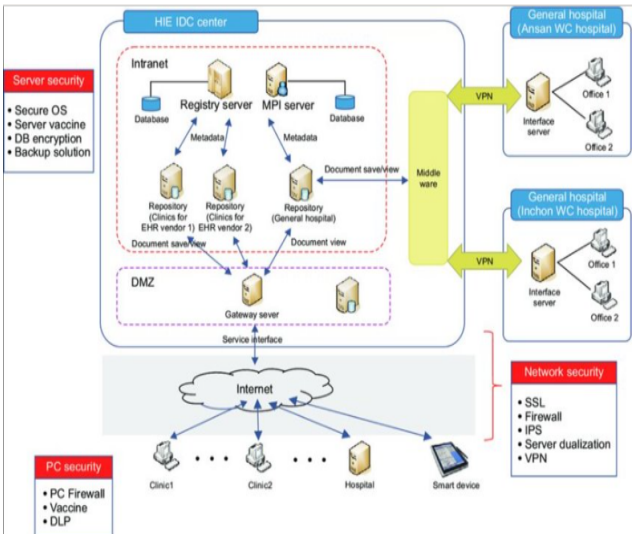

Fig. 9. **Working of Distributed model**

B. Distributed Approach

Let's discuss about the distributed implementation of this approach, here, in this model we have tried to gain the data from multiple sources i.e. we have tried to gather the data from various hospitals, clinics etc.. Nowadays, the majority of data is generated by diverse and geographically scattered infrastructures, with highly dynamic information sources. The rising use of digital infrastructures and data, such as electronic health records, have contributed to generation of an unprecedented volume of recorded data in the health domain. Several management challenges have evolved as a result of the information overload, with possible detrimental repercussions on clinical performance, such as omission errors, delays, and overall patient safety. In fact, managing health data generated in a huge amount on a daily basis, and appropriate handling of medical acts and processes, are both necessary. Building the corresponding A system to efficiently support all health requirements Today's environment is one of the most important challenges. mechanism To find useful patient EPAs in a shared healthcare system Proposed, but most are center-based The approach works fine on a network of limited size. In fact, in large, dynamic systems, centralized mechanisms are limited and desirable. Design an innovative and decentralized approach.

The aim of this algorithm is to build a information system through a logically organized overlay of clinical servers To improve the operation of finding useful information. Each clinical server uses a self-organizing approach Do some simple things in fully autonomous mode Operation based on local information and global scale, A sorted overlay is created. The server contains several seeds Of clinical data such as EHR Vector obtained using Doc2Vec library. That's it You can achieve a logical sort between servers by defining metrics based on server vectors.

V. Conclusion and Further Reference

This paper introduced a decentralized, self-organizing NPL. Base algorithm for building a data management system in A highly dynamic environment such as healthcare. Any health Data such as electronic health records (EHR) contained in the clinical server was indexed by a stored semantic vector According to the artificial neural network model Doc2Vec. The server performs operations locally through a sorted overlay network of clinical servers. So the user Requests can efficiently reach the server containing the destination Content or services in an informed way. experimental The results show the effectiveness of the following approaches: The detection operation becomes faster, The square root of the network size.

A few improvements in this approach might be to increase the size of the training data and also the labeled data as for now, we are using very few reports for training the model and also only 22 classes of objects as labeled data in order to categorize the data being processed. For now, the application of the model is limited to a few samples and hospital and thus it requires more training to be applicable for the real world application where it would be really helpful to be used in the real time health care domain.

References

[1] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167, 2008.

[2] Aleksei Dudchenko and Georgy Kopanitsa. Comparison of word embeddings for extraction from medical records. *International journal of environmental research and public health*, 16(22):4360, 2019.

[3] Agostino Forestiero and Giuseppe Papuzzo. Natural language processing approach for distributed health data management. In *2020 28th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pages 360–363. IEEE, 2020.

[4] Mikolov T. Le, Q. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning*, page Volume 32. pp. II–1188–II–1196., 2014.

[5] Balaji Polepalli Ramesh, Thomas Houston, Cynthia Brandt, Hua Fang, and Hong Yu. Improving patients' electronic health record comprehension with noteaid. In *MEDINFO 2013*, pages 714–718. IOS Press, 2013.