2019

Effective and Secure Healthcare Machine Learning System with Explanations Based on High Quality Crowdsourcing Data

Qinghan Xue
Lehigh University, xqhgeorge@gmail.com

Effective and Secure Healthcare Machine Learning System with Explanations Based on High Quality Crowdsourcing Data

by

Qinghan Xue

Presented to the Graduate and Research Committee

of Lehigh University

in Candidacy for the Degree of

Doctor of Philosophy

in

Computer Science

Lehigh University

(May 2019)

Approved and recommended for acceptance as a dissertation in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

_____
Date

_____
Dissertation Advisor

Committee Members:

_____
Prof. Mooi Choo Chuah, Committee Chair

_____
Prof. Ting Wang

_____
Prof. Brian Davison

_____
Prof. Wei-Min Huang

Acknowledgements

First of all, I would like to express my sincere gratitude to my research advisor, Professor Mooi Choo Chuah, for her generous support, guidance, patience, and encouragement throughout my Ph.D. years at Lehigh. Professor Chuah taught me a lot from critical thinking, motivating new problem, scientific evaluation, presentation organization to seeing the big picture. I am very grateful that Professor Chuah provides me the freedom to explore many interesting research problems and she is always helpful both on stimulating research ideas and on providing favorable research facilities. Professor Chuah is not noly an advisor for research, but also a mentor for many aspects in my life. Without her advice, I could not have finished this dissertation.

I would like to thank those who served as members of my Ph.D. committee, Professor Ting Wang, Professor Brian Davison, Professor Wei-Min Huang and Professor Yinzhi Cao for taking time to attend my Ph.D. defense and providing excellent suggestions for my dissertation.

I would also like to thank my collaborator Professor Yingying Chen at Rutgers for her great help during the early stage of my Ph.D. study. I greatly appreciate Dr. Lei Liu, Dr. Wei-peng Chen from Fujitsu Research Lab, Dr. Wei Wang and Dr. Zhijun Liu from Enviveus, and Dr. Xiaoran Wang, Dr. Jilong Kuang, Dr. Jun Gao from Samsung Research America for offering me internship opportunities. These internship experiences provide me the opportunity to learn how my research can impact the world from the perspective of the industry. I also made many good friends, have learned a lot from them and benefited greatly.

Many thanks to my colleagues and friends in Lehigh University who offered help to me in

various ways. I thank my fellow labmates in Lehigh WiNS Lab: Dawei Li, Wenbo Li, Xin Li, Xiaowen Ying, Xiao Zang, Li Tian for discussion and help on a wide area of topics. I thank my friends in Lehigh University: Tao Xu, Tingzhe Zhou, Jieli Tian, Jiajun Duan, Zhiyu Chen, Zhiyuan Cui, Jundong Yao, Yujie Ji for their help in my Ph.D. life.

Finally and most importantly, I am deeply indebted to my dear mother and father for their love and string support during my Ph.D. study. They have both been with me through the whole process with its ups and downs.

I gratefully acknowledge the support from the research sponsors, including National Science Foundation through CSR grant 1016296, CNS grant 1217379 and CNS grant 1217387.

Contents

List of Tables

List of Figures

Abstract

Affordable cloud computing technologies allow users to efficiently outsource, store, and manage their Personal Health Records (PHRs) and share with their caregivers or physicians. With this exponential growth of the stored large scale clinical data and the growing need for personalized care, researchers are keen on developing data mining methodologies to learn efficient hidden patterns in such data. While studies have shown that those progresses can significantly improve the performance of various healthcare applications for clinical decision making and personalized medicine, the collected medical datasets are highly ambiguous and noisy. Thus, it is essential to develop a better tool for disease progression and survival rate predictions, where dataset needs to be cleaned before it is used for predictions and useful feature selection techniques need to be employed before prediction models can be constructed. In addition, having predictions without explanations prevent medical personnel and patients from adopting such healthcare deep learning models. Thus, any prediction models must come with some explanations. Finally, despite the efficiency of machine learning systems and their outstanding prediction performance, it is still a risk to reuse pre-trained models since most machine learning modules that are contributed and maintained by third parties lack proper checking to ensure that they are robust to various adversarial attacks. We need to design mechanisms for detection such attacks. In this thesis, we focus on addressing all the above issues: (i) Privacy Preserving Disease Treatment & Complication Prediction System (PDTCPS): A privacy-preserving disease treatment, complication prediction scheme (PDTCPS) is proposed, which allows authorized users to conduct searches for disease diagnosis, personalized treatments, and prediction of potential complications. (ii) Incentivizing High Quality Crowdsourcing Data For Disease Prediction: A new incen-

tive model with individual rationality and platform profitability features is developed to encourage different hospitals to share high quality data so that better prediction models can be constructed. We also explore how data cleaning and feature selection techniques affect the performance of the prediction models. (iii) Explainable Deep Learning Based Medical Diagnostic System: A deep learning based medical diagnosis system (DL-MDS) is present which integrates heterogeneous medical data sources to produce better disease diagnosis with explanations for authorized users who submit their personalized health related queries. (iv) Attacks on RNN based Healthcare Learning Systems and Their Detection & Defense Mechanisms: Potential attacks on Recurrent Neural Network (RNN) based ML systems are identified and low-cost detection & defense schemes are designed to prevent such adversarial attacks. Finally, we conduct extensive experiments using both synthetic and real-world datasets to validate the feasibility and practicality of our proposed systems.

Chapter 1

Introduction

1.1 Motivation

In the past few years, cloud computing has emerged to be a useful enterprise IT solution, where scalable and elastic storage and computational resources are provisioned for different cloud services through the Internet. With such affordable services, more and more organizations, businesses and individuals have outsourced both their computational tasks and large-scale data to the cloud. Unfortunately, such data may contain sensitive information such as Personal Health Records (RHRs), business emails, etc, to cloud services [22]. In order to enjoy monetary savings and simplify their local IT managements, they also outsource computational tasks on their data to third-party service providers. Amazon Cloud [9], Dropbox [10], Microsoft Azure Cloud, Google Cloud are some popular cloud platforms utilized by many enterprises.

In addition, many healthcare providers and pharmaceutical companies also have increased cloud-based eHealth solutions to manage health-related information and to automate administrative and clinical functions. For example, the Personal Health Record (PHR) services [109, 102] allow a patient to create, manage, and control his/her personal health data in a centralized place through the web, which has made the storage, retrieval, and sharing of the medical information more efficient. Two major cloud platform providers, Google and Microsoft are both providing PHR services, Google Health [11] and Microsoft HealthVault [14], respectively. The global healthcare systems are rapidly adopting Electronic

Health Records (EHR), which are systematic collections of longitudinal patient health information (e.g., diagnosis, medication, lab tests, procedures, etc.). This will dramatically increase the quantity of clinical data that are available electronically.

With the huge growth of the large scale clinical data collected by the healthcare service providers and the growing need for personalized care, researchers are keen on developing data mining methodologies to mine hidden patterns such that they can identify critical factors which affect disease progressions, and use such information to aid the healthcare professionals in making better treatment decisions. While studies have shown that those progresses can significantly improve the performance of various healthcare applications, learning efficient medical patterns of healthcare concepts, however, is still an open challenge. Recently, deep learning techniques have been adopted in medical patterns and patient representation learning. In [113], the authors develop a deep neural network composed of a stack of denoising autoencoders to process Electronic Health Records (EHR) that captured stable structures and important patterns in the data records.

Moreover, based on those clinical data mining techniques, cloud service providers may also wish to offer genetic analysis or predictive services over the internet so that users can get relevant medical answers based their submitted healthcare related questions. For example, a patient may want to use a web service that stores and maintains all his/her medical records. In addition, he/she may want to obtain prediction of whether or not he/she will suffer from a specific disease. Therefore, data driven healthcare, defined as the usage of those available big medical data to provide the best and most personalized care, is becoming to be one of the major research trends that aim to revolutionize the healthcare industry [111].

1.2 Problems and Challenges

1.2.1 Privacy Leakage of Cloud Services

While cloud-assisted mHealth monitoring could offer a great opportunity to improve the quality of healthcare services and potentially reduce healthcare costs, there are many security and privacy concerns such as data confidentiality and privacy exposure which could

impede making this technology a reality.

The main concern is about the privacy of patients' sensitive Personal Health Information (PHI) and who could gain access to the PHI when they are stored on a third-party server. Patients lose physical control to their own personal health data when such data is placed under the control of cloud service providers, which people may not fully trust.

Although the exist privacy laws such as HIPAA (Health Insurance Portability and Accountability Act) [53] provide baseline protection for PHI, they are generally considered not applicable or transferable to cloud computing environments [60] when the cloud providers are not fully trusted worthy [61]. As a famous incident, a department of veterans affairs database containing sensitive PHI of 26.5 million military veterans, including their social security numbers and health problems was stolen by an employee who took the data home without authorization [150]. On the other hand, due to the high value of the sensitive PHI, the third-party storage servers are often the targets of various malicious behaviors which may lead to exposure of the PHI. Thus, to ensure patient-centric privacy control over their own PHI, it is essential to have privacy protection mechanisms that work with semi-trusted servers. Traditional protection mechanisms of merely removing patients' personal identity information (such as names or SSN) or using anonymization technique are insufficient to prevent information leakage.

In order to minimize the risk of data leakage to the cloud service providers, data owners opt to encrypt their health records before outsourcing to the cloud. Basically, the PHR owners themselves should decide how to encrypt so that only the legitimate data owners can access the data by decrypting it using their private decryption keys. While encryption is a commonly used method to preserve data confidentiality by storing ciphertext in the cloud, it may prevent clients to utilize and compute their cloud data efficiently. For example, a client using Wuala [19] has no way to operate meaningful search on its encrypted cloud data, unless it first retrieves all the data from the cloud side and decrypts those ciphertexts. This process is resource-consuming to a client, especially for medical data with large-scale data size. Thus, this poses the question of how to facilitate effective search while minimally revealing which contents the health service providers possess.

In order to solve this problem, various searchable encryption (SE) solutions (e.g., [44,

57, 139, 32]) have been proposed recently to allow users to securely search over encrypted data. However, most of the earlier proposed works [32, 41, 42, 86, 141, 173, 152] focus on supporting simple search functions, such as single-keyword queries are too restrictive for practical use. For example, they are not suitable for handling complex search operations, such as multi-dimensional range queries. Thus, to enrich search functionality, several multi-keyword search schemes that enable conjunctive or disjunctive search terms were proposed [32, 36]. Unfortunately, most of them only allow exact keyword search instead of considering typos in the users' inputs. while some schemes that can handle typos have recently been proposed [100, 107], they are either inefficient or do not handle query unlinkability.

In addition, it is hard to learn useful information or patterns from the encrypted data (i.e. medical records) due to the cryptographic techniques and the search complexity increases linearly with the number of stored data. Therefore, it is important to explore how one can design a system that can support encrypted data analysis and efficient data retrieval without compromising privacy and security.

1.2.2 Challenges of Collecting Large Medical Dataset and Mining Hidden Patterns from such dataset

The ability to learn an accurate model for predicting patient outcomes typically hinges on the amount of training data available. It is difficult to generate learning models that can accurately predict rare or infrequent events. A single hospital may not have sufficient data about rare health events and hence need other hospitals to share such data so that more accurate prediction model can be created. In addition, unlike other data sources, medical data is inherently noisy, irregularly sampled, and heterogeneous (i.e. data comes from different sources such as lab tests, doctor's notes, etc). Therefore, two important questions naturally arises:

- how to gather a large collection of useful electronic clinical records, which can provide an opportunity to study medical cases, evidences and knowledge.
- how can one employ learning methods, which can automatically extract useful patterns to facilitate the advancement of clinical research informatics.

Recently, with the fast development of hospital information systems, a large collection of electronic clinical records become available, where research experts investigate approaches to building predictive models that involves augmenting data from individual hospitals with data from other hospitals. Clinical Data Mining (CDM) is the application of data mining techniques using clinical data [80], which involves the conceptualization, extraction, analysis, and interpretation of available clinical data for practical knowledge-building and clinical decision-making [64]. The main objective of clinical data mining is to develop quantitative models for patients that can be used to predict health status, as well as to assist the healthcare professionals in decision making. While, applying data from multiple hospitals presents an opportunity for making predictions in a target task, the data collected from each hospital lack of quality verifications and may contain hospital-specific distinctions, which in turn make it hard to extract important observations or features. For example, a hospital may desire a highly accurate disease prediction model but unwilling to disclose the details of all patient records and hence only provides data with missing or noisy values. Thus, hospitals need to be encouraged to share high quality data such that healthcare data mining researchers can access to huge amount of high quality patients' data and produce better models. Incentive mechanisms need to be designed to encourage hospitals to share high quality data so that disease prediction models with higher accuracy can be produced.

In addition, instead of collecting data from multiple hospitals, previous works also tried to link multiple data sources (e.g., medical website, medical blogs) to build joint knowledge based dataset that could be used for predictive analysis and discovery [163, 48, 151]. While the large availability of biomedical data offers great promise for accelerating clinical research, there is a growing need in the healthcare scenario to develop data mining methodologies to mine hidden patterns and discover new knowledge to provide more personalized disease diagnosis or prediction models. For example, precision medicine attempts to ensure that the right treatment is delivered to the right patient at the right time' by taking into account several aspects of patient's data, including variability in molecular traits, environment, Electronic Health Records (EHRs) and lifestyle [16, 110, 54].

While machine learning methods demonstrate great promises (e.g., [149, 115, 101, 103]) in clinical data analysis, sometimes, we need to use expert medical knowledge which un-

fortunately is spread out across different medical websites. Using such distributed medical information is challenging since such information is highly complex and heterogeneous in natural. For example the same clinical phenotype can be expressed using different codes and terminologies. A patient diagnosed with "type 2 diabetes" can be identified by laboratory values of hemoglobin A1C greater than 7.0, presence of 250.00 ICD-9 code, "type 2 diabetes mellitus" mentioned in the free-text clinical notes, etc. Therefore, exploring the associations among all the different pieces of information from those sizeable clinical data [116, 132] is a fundamental problem that needs to be solved before one can develop reliable medical tools based on data-driven approaches and machine learning techniques.

1.2.3 Challenges of Understanding Patients' Queries

Based on those clinical data mining techniques and generated learning models, cloud service providers may allow authorized patients to conduct searches for disease diagnosis, medicine recommendations, etc. Thus, enabling computers to understand users' queries and answer questions about their contents has recently attracted intensive interest [131, 73, 74, 129, 119, 31].

In the healthcare domain, patient data collected by hospitals consists of multiple concepts including discharge summaries, progress notes, etc, which contain rich latent relationships that are difficult to be learned and represented. In order to overcome this limitation, it is common in healthcare applications, to rely on carefully designed information extraction and feature representations [142, 65, 158]. The most common approach to extract the semantic similarity of the data is to encode them with many lexical, syntactic and semantic features and then compute various similarity measures between the obtained representations. Recently, it has been shown that the problem of semantic text matching can be tackled using distributional word matching, where the questions could be matched with candidate answers [167].

However, it is still a challenge to understand latent relationships and efficient representations of medical concepts in the health-oriented questions. In order to solve this problem, many researchers and institutions have utilized deep learning methods to understand the

representations of the complicated data, which can achieve good performance on multiple prediction tasks. For example, Recurrent (RNN) or Convolutional Neural Network (CNN) models have recently achieved significant performance gains and increasingly been used for various NLP related tasks such as language modeling [30], sentiment analysis [138], syntactic parsing [55], and machine translation [99]. In addition, these deep neural networks are also able to effectively capture the compositional process of mapping the meaning of individual words in a sentence to a continuous representation of the sentence. For example, it has been recently shown that Convolutional Neural Networks (CNNs) are able to efficiently learn to embed input sentences into low-dimensional vector space, preserving important syntactic and semantic aspects of the input sentences, which leads to state-of-the-art results in many NLP tasks [83, 88, 169]. While existing systems can discover efficient representations of medical concepts, they have difficulty in dealing with patients' questions especially for the synonym scenarios since patients may use different medical terms in their questions even when they have suffered from the same diseases. In addition, most systems lack explanations for the prediction results which prevent medical personnel and patients from adopting such learning models. Thus, a deep learning approach which can better understand users' healthcare related queries needs to be designed to provide higher accuracy and more useful explanations for the prediction results to these users.

1.2.4 Attacks & Defenses of Machine Learning Models

Despite the efficiency of machine learning systems and their outstanding prediction performance, most machine learning modules that are contributed and maintained by third parties, lack proper checking to ensure that they are robust to the adversarial attacks.

Recent studies have shown that many machine learning models are not robust at all when someone wants to crack them on purpose [67, 146, 124, 123, 40, 170]. For example, intelligent attackers can force many deep learning models to misclassify examples by adding small and hardly visible modifications on a regular sample (i.e. make imperceptible modifications to pixel values). This leads to security concerns, especially when applying deep neural networks to safety related systems such as medical diagnosis, self-driving cars, etc.

In addition, the attackers may also add some small perturbations to the weights in the original models, which will make classifiers unable to classify correctly. For example, the attackers can add random noise on the learning weights in a RNN system such that there is a high chance that some important features will be treated as unnecessary in the feature selection step, which ruins the robustness of learning-based classifiers. Such attacks damage the capability of a RNN-based model to learn properties of the important long-term dependent features for the classification task. Thus, in order to make deep neural networks more robust to adversarial attacks, several defensive algorithms have been proposed recently [78, 124, 171, 91, 164, 172]. Those defenses can be clustered into three different approaches: (i) training the target classifier with adversarial examples, called adversarial training [146, 68]; (ii) modifying the training procedure of the classifier, e.g., defensive distillation [126]; and (iii) quantizing neural network weights and activation functions into low bit-width [165, 161, 130]. However, recent studies showed that those defensive algorithms can only marginally improve the accuracy under the adversarial attacks [39, 40]. Meanwhile, some of the those defenses require adversarial examples to train the model and are devised with specific attack models in mind, which are not effective against new attacks. Thus, more research needs to be conducted to ensure the robustness of any deep learning models created for healthcare purposes and it is also necessary to find better methods to defend against the potential adversarial attacks.

1.3 Dissertation Statement and Contributions

As discussed previously, with the increasing availability of affordable cloud computing resources and the availability of large healthcare related datasets, recent research has focused on designing a robust healthcare system that can provide personalized care such as disease diagnosis, survival rate prediction, medication recommendation, etc. Several important issues such as security & privacy of healthcare data, availability of large heterogeneous medical data, developing highly accurate and secure deep learning based disease diagnosis and progress prediction models need to be explored before this vision can be realized. In this thesis, we have made the following contributions to the above important issues that

help to provide a more accurate personalized cloud-based healthcare system:

- In order to protect the information leakage, we design a privacy-preserving disease treatment, complication prediction scheme (PDTCPS), which allows authorized users to conduct searches for disease diagnosis, personalized treatments, and prediction of potential complications of their illnesses. In particular, we design an encrypted index tree which supports fuzzy keyword queries. Such tree-based structure is used to provide search efficiency, which consists of three levels:
  - Each top level node contains an encrypted category keyword that represents a specific body part, e.g., bone & joints, kidneys, etc, and a Bloom filter which contains all the disease keywords classified under this top level node, and their associate fuzzy keyword sets.
  - The 2^{nd} level nodes store the relevant information, e.g., associated diseases classified under each top-level node. For example the related diseases "acute tubular necrosis", "kidney stones", etc. will be stored in the 2nd level node under the top level node that represents "kidney".
  - Each 2^{nd} level node has three child nodes, one for diagnosis, one for complication prediction and one for treatment options. These three child nodes are leaf nodes, where each stores relevant information including a training model, a encrypted feature set and the corresponding bloom filter containing fuzzy keyword set of each disease.

  In addition, we also include random components in our design to provide query unlinkability, hide search and access patterns. Such features strengthen further the security & privacy capability of our design. Moreover, our scheme accommodates typos in users' submitted requests, which could not be handled by the existing schemes. Finally, we present security analysis, and conduct experiments to evaluate the effectiveness and efficiency of our proposed scheme.

- While encryption algorithms are useful in preventing the information leakage, some organizations or hospitals are not interested in sharing their data unless they can

obtain certain rewards. Thus, in order to create high quality healthcare datasets to accelerate clinical research, we have developed two tools:

  - We propose an incentive mechanism to encourage hospitals to share truthfully high quality data which can then be aggregated to generate prediction models with higher accuracy rates. Our incentive mechanism satisfies two desirable properties: individual rationality (the rewards that the selected hospitals receive will be greater than their resource consumptions) and platform profitability (the benefits brought by the participating hospitals should be larger than the total rewards paid to those hospitals).
  - In order to generate more comprehensive and accurate prediction models, we propose a data cleaning & feature selection method which allows healthcare data mining researchers to clean up the available large scale dataset and identify relevant features which are critical in predicting the progression and survival rate of any given disease. We also explore how data cleaning and feature selection techniques affect the performance of the prediction models.

  Finally, we demonstrate the effectiveness of our approaches using three large datasets from three population-based studies, namly ALS [1], RHC [2] and STAR*D [121]. Our experimental results show that our prediction models yield good performance in ALS slope, ALS & RHC survival, and STAR*D relapse predictions.

- Instead of collecting data only from multiple hospitals, we have designed a knowledge extraction framework, which can integrate the expert knowledge from multiple sources to create a more useful knowledge base for disease diagnosis and prediction. For example, online medical websites can be mined to extract reliable contents that can be included in the learning models to improve the overall performances of the system.

  Meanwhile, we also propose a deep learning based medical diagnosis system (DL-MDS), which can allow authorized users to conduct searches for disease diagnosis based on their own personalized queries and also provide explanations for the prediction results. Our DL-MDS consists of three components:

- A Topic Model Module: in order to explore key information in the questions, we extract informative keywords to generate a topic module, which can (i) capture the relational connections among questions and diseases; (ii) provide model interpretation and improve the prediction performance.
- A Query Processing Module: in order to further analyze those questions, we also train a query processing module, where we first extract medical questions from healthcare web blogs. Then, we apply a CNN-based word embedding method, at a sentence level to capture important information. Next, we combine those key information with the information extracted through topic module to train a deep learning module.
- Medical Diagnosis Modules: as for the medical diagnosis, we first collect disease information from professional medical websites. Then, a similarity matching algorithm will be applied to generate an aggregated dataset where the information of same diseases from multiple sources will be linked together. Next, we separate diseases into different clusters based on their ICD-10 codes and utilize machine learning techniques to generate a medical diagnosis module for every disease cluster.

Finally, we evaluate our proposed methods using real-world data. Our experimental results show that our proposed system yields good performance on patients' queries processing and medical diagnosis.

- While existing machine learning (ML) models have been used to make remarkable progress in multiple prediction tasks, they are still vulnerable to the adversarial attacks, which leads to security concerns, especially when applying deep neural networks to safety related systems such as medical diagnosis, treatment recommendations, etc. Thus, in our work, we present two types of harmful threats to RNN-based machine learning systems which can trick such ML systems to output wrong prediction results.
  - For the first attack, we present an efficient and effective framework, which is recently designed by a MSU research team, to generate adversarial samples with

minimum perturbations on input medical sequences.

- For the second attack, instead of using adversarial samples, we propose a new attack strategy, which is to randomly add gradient noise to adjust recurrent weights, where the gradients noise are propagated through a set of important weight vectors until those features associated with the modified weights become insignificant during training and hence produce a ML model which generates wrong classification labels for targeted classes.

In addition, we also propose a detection scheme which can be used to infer these two types of adversarial attacks.

- Detection for Adversarial Samples Attack: Recent studies have shown that adversarial samples are much more sensitive to perturbations than normal samples. If we impose random perturbations on a normal and an adversarial samples respectively, there is a significant difference between the ratio of label change due to the perturbations. Thus, we present a recent work on detecting adversarial samples, where the authors determine if an input is a normal sample or an adversarial one through mutation testing.
- Detection for Weights Adjustment Attack: Based on our observations, we notice that important features identified from a genuine RNN-based model are significantly different from those identified from a maliciously modified model created from the original model. Thus, we design a detection scheme, which compares the identified important features of a similar ML system (e.g., a traditional random forest model) with those obtained using the downloaded RNN-based ML system which may be maliciously modified.

Moreover, we propose a Static/Dynamic Quantization-based defense solution (SDQDS), which uses quantized weights and does not require any modification to the training procedure and yet relatively effective in defending against these two types of adversarial attacks. Finally, we conduct various experiments using both synthetic and real healthcare datasets which contain patient records with many attributes to demon-

strate that these two types of attacks are effective against RNN-based healthcare machine learning models. The evaluation results for our detect and defense solutions also show that they are feasible and practical.

1.4 Organization

The dissertation is organized as follows.

In Chapter 2, we propose a privacy-preserving disease treatment and complication prediction scheme (PDTCPS), which allows authorized users to conduct searches for disease diagnosis, personalized treatments, and prediction of potential complications. PDTCPS uses a tree-based structure to improve search accuracy and storage efficiency. In addition, our design also allows healthcare providers and the public cloud to collectively generate aggregated training models for disease diagnosis, personalized treatments and complications prediction. Moreover, our design provides query unlinkability and hides both search & access patterns. Finally, we conduct experiments via using two realistic datasets and the evaluation results show that our scheme is more efficient and accurate than the other two existing schemes.

In Chapter 3, we propose an incentive model which provides individual rationality and plat- form profitability features to encourage hospitals to share high quality data for such predictions. We also demonstrate a data cleaning & feature selection method, which allows healthcare data mining researchers to clean up the available large scale dataset and identify critical features in predicting the progression and survival rate of any given disease. In addition, we provide extensive experiments using three large datasets to prove that our learning models perform well in disease predictions and our incentive model can achieve individual rationality and platform profitability in practice.

In Chapter 4, we design a medical knowledge extraction framework to merge medical knowledge extracted from heterogeneous medical websites to create a more useful knowledge base for disease diagnosis and prediction. In order to allow authorized users to conduct searches for dis- ease diagnosis more accurately, we also propose a deep learning based medical diagnosis system (DL-MDS), which consists of three components: (i) medical diagnosis

modules; (ii) a topic model module; (iii) a query processing module. We evaluate our proposed methods using real-world data, where the experimental results show that our proposed system yields good performance on patients' queries processing and medical diagnosis.

In Chapter 5, we present a particular threat to Recurrent Neural Network (RNN) based healthcare ML systems by introducing two types of attacks: (i) adversarial samples attack, (ii) the malicious modification attack. We also propose low-cost detection and defense mechanisms to prevent such adversarial attacks. Finally, we conduct experiments using both synthetic and real-world datasets to validate the feasibility of our proposed methods.

In Chapter 6, we will conclude the dissertation and discuss future work directions.

Chapter 2

Privacy Preserving Disease Treatment & Complication Prediction System (PDTCPS)

2.1 Background

In recent years, cloud computing has emerged to be a popular technology that provides scalable and elastic storage and computation resources for enterprises and individuals. More and more organizations and individuals begin to embrace these benefits by outsourcing their data into the cloud [23]. For example: online personal health record (PHR) systems such as Microsoft HealthVaults allow patients to store and manage their own medical records in the public cloud. Such systems allow users easy access and sharing of their personal health data.

Although the cloud-assisted healthcare systems offer a great opportunity to improve the quality of healthcare services and potentially reduce healthcare costs, there are many security and privacy concerns. For example, people have started to realize that they would completely lose control over their personal information once it enters the cyberspace. In order to minimize the risk of data leakage to the cloud service providers, sensitive data must be encrypted before being outsourced into the cloud. By doing so, the cloud service providers

can only see data in encrypted form and never learn any information about the encrypted data values. However, this in turn makes data utilization challenging. For instance, it is difficult to apply machine learning techniques to learn from aggregated privately encrypted data for accurate predictions. In order to solve the problem, a set of techniques has been developed (e.g., [25, 62, 69, 95]). While some approaches [62, 69] demonstrated that basic machine learning algorithms such as simple linear classifiers can be performed efficiently to build models over a small scale encrypted dataset, their efficiency degraded rapidly as its size grows. Though other techniques [25, 95] had utilized more sophisticated classifiers (e.g., support vector machine) to address the problem, either they lack security & privacy features or require large computational cost. A recent work [105] designed a scheme which provides machine learning models over encrypted dataset but their encryption scheme has high computational and communication cost. Another recent work [33] also designed a scheme which allows data mining over encrypted data but their scheme does not construct encrypted index tree for efficient search. Neither schemes provide features to hide search and access patterns.

To overcome the above limitations, in this chapter, we design a privacy-preserving disease treatment, complication prediction scheme ($PDTCPS$), which allows authorized users to conduct searches for disease diagnosis, personalized treatments, and prediction of potential complications of their illnesses. In particular, we design an encrypted index tree which supports fuzzy keyword queries. The tree-based structure is used to provide search efficiency. Each top level node in our encrypted index tree contains an encrypted category keyword that represents a specific body part, e.g., bone & joints, kidneys, etc, and a Bloom filter which contains all the disease keywords classified under this top level node, and their associate fuzzy keyword sets. All relevant information, e.g., associated diseases classified under each top-level node will be stored in the 2^{nd} level nodes. Each 2^{nd} level node (representing k diseases) has three child nodes, one for diagnosis, one for complication prediction and one for treatment options. These three child nodes are leaf nodes. Each leaf node stores relevant information about k diseases, including a training model, its encrypted feature sets and the corresponding Bloom filter containing fuzzy keyword set of each disease. In addition, we include random components in our design to provide query

unlinkability, hide search and access patterns. Such features strengthen further the security & privacy capability of our design. Moreover, we present security analysis, and evaluate the effectiveness and efficiency of our proposed scheme using two datasets from the UCI machine learning repository [18]. Our experimental results show that compared to two existing schemes described in [105, 33], our scheme is more efficient (in terms of communication cost) and has higher accuracy than both of these existing schemes. Additionally, our scheme accommodates typos in users' submitted requests, which could not be handled by the existing schemes. In summary, our contributions can be summarized as follows:

- We propose a Privacy-Preserving Disease Treatment, Complication Prediction Scheme ($PDTCPS$), which allows users to conduct privacy-aware searches with high search efficiency and accuracy.
- $PDTCPS$ is designed to handle typos in queries and provide query unlinkability with minimal information leakage.
- Our design allows healthcare providers and the public cloud to collectively generate aggregated training models for disease diagnosis, personalized treatments and prediction of potential illness complications.
- We provide a formal security analysis to justify the privacy-preserving guarantee of our proposed scheme.
- We present simulation results of our proposed scheme using two UCI datasets, namely the PIMA Indians Diabetes and the Breast Cancer Wisconsin Datasets.

2.2 Problem Formulation

In this subsection, we first describe our system and threat models. Then, we describe the design goals of our proposed privacy-preserving disease treatments and complications prediction system ($PDTCPS$). Next, we provide descriptions of some important building blocks used in our solution.

2.2.1 System Model for PDTCPS

PDTCPS consists of four parties: the hospitals, the public cloud server, a fully-trusted authority (TA), and individual clients, as shown in Fig 2.1.

- Hospitals: Hospitals first collect patients' medical records, encrypt them and store them in the private clouds they owned. The private cloud servers may perform data mining operations over the stored data to generate locally trained models. Based on these models, the hospitals can later diagnose diseases, provide personalized treatments, and predict disease complications for their patients. However, since each hospital may only have limited number of patients associated with a particular disease, the prediction models may not always be comprehensive and accurate. Thus, in order to obtain more accurate prediction, the hospital servers may send relevant information extracted from their trained model securely to the public cloud so that the public cloud can perform data mining operations on the aggregated data received to generate a more accurate prediction model for all participating hospitals to use.

- Semi-trusted public cloud: The public cloud stores the relevant encrypted information sent from each participating hospital, and performs data mining operations to generate predictive models. It also constructs a keyword based encrypted index tree which allows authorized clients to conduct searches based on their individual profiles, lab tests for potential disease diagnosis, treatment options, and risk analysis of potential complications related to their current illnesses.

- Fully-trusted authority (TA): TA is responsible for generating and distributing the symmetric encryption keys to authorized clients and all participating hospitals. It is also responsible for sending relevant disease categorization information, e.g., which disease belongs to which top-level category nodes, to the hospitals and the public cloud.

- Clients: Clients refer to those who wish to conduct searches for disease diagnosis, personalized treatments, and assessing their risks of disease complications caused by their current illnesses.

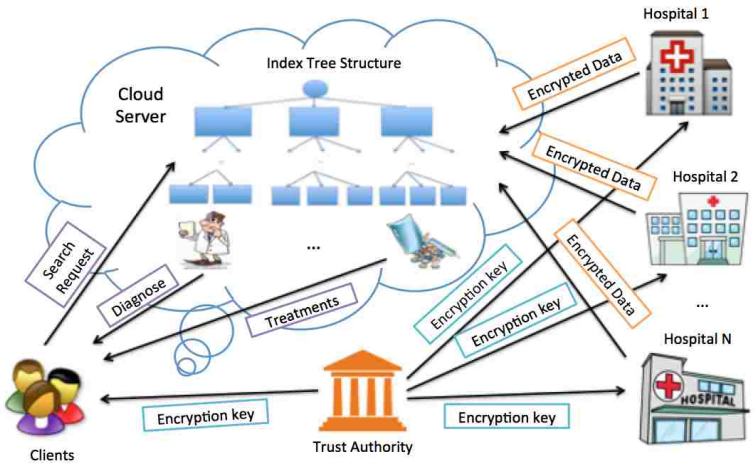


Figure 2.1: System Model for PDTCPS

At the initial phase, TA generates the encryption keys and sends them to all participating hospitals and their authorized clients. Upon receiving the keys, each hospital server first encrypts its data and performs data mining operations to generate locally trained models. Each hospital server then sends the relevant information from the locally trained models securely to the public cloud. The public cloud will then generate aggregated trained models for disease diagnosis, possible treatment models for different groups of patients based on their profiles and medical histories, and prediction models of any potential disease complications. In addition, the public cloud will generate an encrypted index tree which allows clients to search for information more efficiently. Details of what the encrypted index tree contains will be described in Section 2.3.

When a client wishes to query the public cloud for health related predictions, the client first uses the received keys from the TA to generate a search request and then sends it to the public cloud. After receiving the encrypted query, the public cloud server will perform the search over the encrypted index tree and send back all the relevant answers to the authorized client.

2.2.2 Adversarial Model

We assume that the trusted authority can be trusted fully and it will not be compromised. As for all participating hospitals, we assume that they are semi-trusted, i.e., they will honestly follow the designated protocols but always curious to gain additional insights from

the information sent by other hospitals. They may also collude with the cloud server to find such information.

Similarly, we also adopt a "honest-but-curious" model for the public cloud server as in [153, 154]. Like the hospitals, it will execute the designated protocols honestly but will be curious to infer any extra information it can derive from the information sent by all participating hospitals and from the queries/responses issued/received by the authorized clients.

Depending on the available information to the cloud server, the following two threat models are considered in this work:

- Known Ciphertext Model: The encrypted data, the secure index, encrypted queries and responses are all available to the cloud server.
- Known Background Model: In addition to the available information assumed in the former model, the cloud server can also use statistical information to deduce specific contents in a query. It can even collude with other attackers to derive additional information from the encrypted data.

In addition, we assume users are trusted entities. They obtain authorized keys from the TA.

2.2.3 Design Goals

To address the security and threat models we have presented earlier, we design a $PDTCPS$ scheme, which allows authorized users to conduct privacy-aware searches for disease diagnosis, personalized treatment and prediction of potential complications based on their individual profiles, laboratory test results, and potential medical histories. Our system is designed with the following goals in mind:

- Fuzzy Keyword Search: During query generation, an authorized client may make typos while inputting query contents. For example, a client may type "dibetes" instead of "diabetes" in the following query: "(disease="dibetes")". Our scheme should support such fuzzy query and still return relevant information.

- Search Efficiency and Accuracy: The scheme should achieve high search accuracy, i.e., it should return mostly correct answers to search queries. It should also achieve high search efficiency, i.e., the average search time per query is small.

- Privacy Guarantee: Our system should provide privacy guarantees by not leaking sensitive information about stored data or encrypted indices. Our system should provide query privacy and unlinkability. The cloud server should not be able to deduce sensitive contents that have been used for search. Submitted queries should look different each time even if the same keyword and lab results are submitted. Furthermore, the search and access patterns should be hidden from the public cloud server. In other words, the encrypted index structure should be designed such that the server traverses different nodes on the index tree even for the same search request.

- Extensibility: Our system should be designed such that the encrypted index tree as well as trained data models can be updated easily without complete redesign.

2.2.4 Important Building Blocks

Before we present the detailed description of our newly designed scheme, we first discuss some of the security tools we use in this work, and define a few terminologies.

1. Organization of Information Regarding Various Diseases: Patients may suffer from different types of diseases. To make it easier for $PDTCPS$ we design to answer users' questions regarding diagnosis, treatment options, or potential disease complications, we decide to categorize patients' illnesses similar to how a popular healthcare forum website that organizes different types of diseases. Diseases are categorized based on how they affect human body parts (refer to Fig 2.2), e.g., Endocrine includes all diseases which affect the endocrine system such as diabetes, hypothyroidism, hyperthyroidism, etc.

For each disease, our system keeps several pieces of important information, namely (i) a trained model for disease diagnosis based on results of laboratory tests, symptoms, (ii)various treatment options based on patients' profiles, and (iii) a trained model for complication prediction based on patients' profiles, laboratory tests, and medical histories, e.g., other diseases a patient may have.

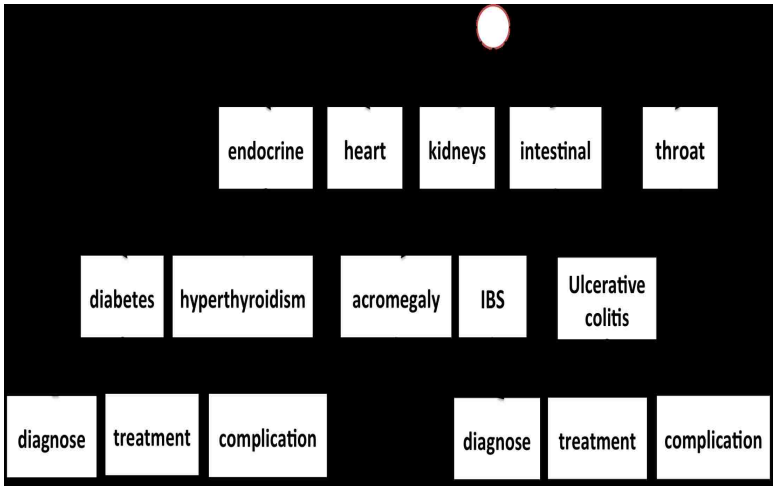


Figure 2.2: Healthcare Searchable Tree

2. Order-preserving Encryption: Order-preserving symmetric encryption (OPE) is a deterministic encryption scheme which preserves numerical ordering of the plaintexts. It allows order relations between data items to be established based on their encrypted values, without revealing the data itself. For example if $x \leq y$, then $OPE_K(x) \leq OPE_K(y)$, for any secret key K. Thus, with the help of OPE encryption, the server can perform data mining operations over the encrypted data.

3. Parallel SVM method: Support Vector Machines (SVMs) are powerful classification and regression tools, but their computational costs increase rapidly with the size of training instances. Efficient parallel algorithms for constructing SVM models are critical to ensure that SVM can be used for large scale data mining analysis.

The parallel SVM method [162] we use is based on the cascade SVM model where a partial SVM model is constructed for each partition of a large dataset. Then, the partial SVMs are aggregated iteratively as shown in Fig 3. The sets of support vectors from two SVMs are merged into one set and used to create a new SVM. Such a process is repeated until only one set of support vectors remain. This parallel SVM approach allows large scale optimization problems to be divided into smaller independent optimizations.

4. Parallel Decision Tree method: Decision trees are simple yet effective classification algorithms, but one needs to sort all numerical attributes in order to decide where to split a node within a decision tree, which costs much computation time when a large data set is involved. Thus, it is important to develop parallel version of decision tree algorithms which

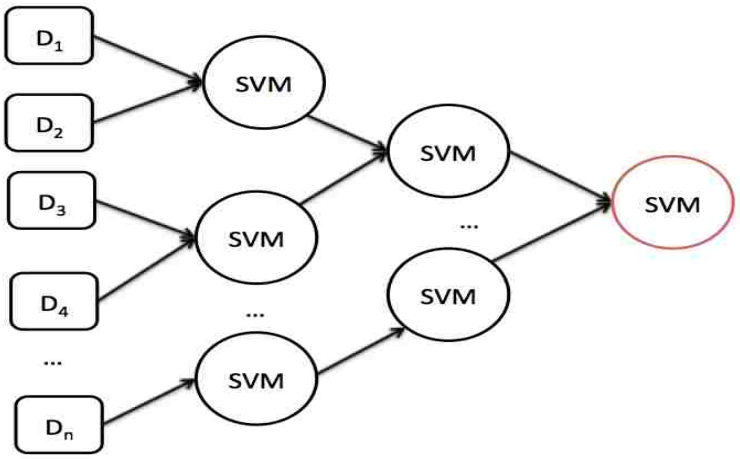


Figure 2.3: Training process of parallel SVM

can be efficient and scalable.

The decision tree method we use is a parallel histogram-based decision tree algorithm for classification [29] where the master node builds the regression trees layer by layer as shown in Fig 2.4. At each iteration, a new layer is constructed as follows: each node compresses its share of the data using histograms and sends them to the master node. The master node merges the histograms and uses them to approximate the best splits for each leaf node, thereby constructing a new layer. Then, the master node sends this new layer to each participating node, and those nodes construct histograms for this new layer. Therefore, every iteration consists of an updating phase performed simultaneously by all the participating nodes and a merging phase performed by the master node. The communication cost for this method consists of all the histograms sent by the participating nodes to the master and the master sending information of a new layer of the tree to those nodes.

2.3 PDTCPS SCHEME

As discussed earlier, PDTCPS provides a secure way for clients to diagnose their diseases, predict complications and search for possible treatment options for their illnesses. One important component of our PDTCPS system is the encrypted index tree that the public cloud constructs based on instructions given by the TA. Before we describe our scheme, we

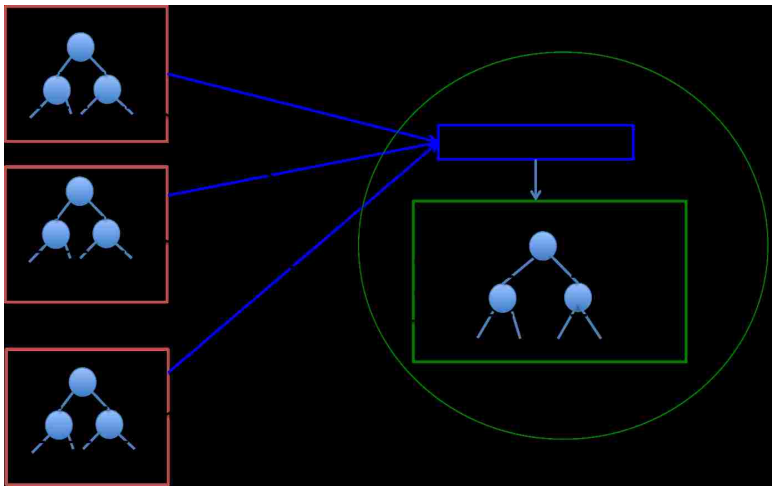


Figure 2.4: Training process of parallel Decision Tree

first give the definitions of various notations we use.

Notations:

- K_O - the symmetric key for OPE encryption.
- K_B - the Bloom filter generation key.
- K_A - the key used to generate key hash values for keywords, i.e. $Enc(w) = KeyHash(K_A, w)$.
- CW - the category keywords set, denoted as CW=$\{cw_1, cw_2, \cdots, cw_{|CW|}\}$.
- W - the disease keywords set, denoted as W=$\{w_1, w_2, \cdots, w_{|W|}\}$.
- $\widetilde{W_i}$ - a subset of W, indicating the disease keywords in the i^{th} category, denoted as $\widetilde{W_i}$=$\{a_{i1}, a_{i2}, \cdots, a_{i|\widetilde{W_i}|}\}$, where $a_{ij} \in W$.
- $\widetilde{S}$ - a set, indicating the number of children that under each category node, denoted as $\widetilde{S} = \{|C_1|, |C_2|, \cdots |C_{|\widetilde{S}|}|\}$.
- k - the number of diseases stored in every 2^{nd} level node.
- $bf(w_i)$ - a Bloom filter, containing the keyword w_i and its associated fuzzy keywords.
- $\widetilde{sv}$ - a set, indicating the training features of a disease.
- cw_q - the category keyword for the query.
- F - the lab test results set, denoted as $F = \{F_1, F_2, \cdots\}$.
- h - the number of hash functions used in generating the Bloom filter.

2.3.1 Overview

Fig 2.5 is an overview of PDTCPS which shows the information provided by the TA, hospitals, and queries submitted by authorized clients.

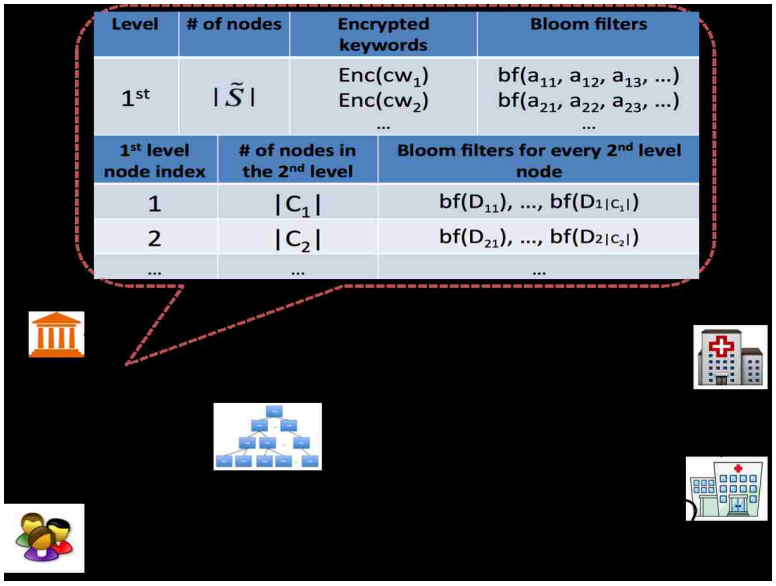


Figure 2.5: Overview of PDTCPS

During the encrypted index tree construction phase, the TA sends the public cloud some information to help the public cloud build the encrypted index tree. Specifically, TA sends the public cloud a set of encrypted category keywords, which will form the 1^{st} level nodes. In addition, the TA sends a Bloom filter for each 1^{st} level node which contains keywords of all the illnesses listed under this 1^{st} level node as well as their associated fuzzy keywords. Fuzzy keywords are generated to deal with typos. For example, for a disease or category keyword, w_i = "hypoglycemia", the following wild-card keywords having an edit distance of 1 from the keyword "hypoglycemia": $\{*hypoglycemia, h*ypoglycemia, hy*poglycemia, hyp*oglycemia, hypo*glycemia, hypog*lycemia, hypogl*ycemia, hypogly*cemia, hypoglyc*emia, hypoglyce*mia, hypoglycem*ia, hypoglycemi*a, \cdots\}$ are inserted into the Bloom filter. Note that, it is easy to extend our system to support multiple edit distances. (e.g., generate one Bloom filter per edit distance).

The TA also sends information regarding the number of children each 1^{st} level node will have, e.g., top-level node i will have $|C_i|$ 2^{nd} level nodes. Each 2^{nd} level node has a Bloom filter containing k encrypted disease keywords and their associated fuzzy keyword sets (to

address typos). All Bloom filters associated with 2^{nd} level nodes are also sent to the public cloud. The public cloud then stores those Bloom filters in the appropriate child nodes of first level nodes.

Each 2^{nd} level node represents k diseases and has three child nodes, namely (i) diagnosis, (ii) complication prediction, and (iii) treatment options. These child nodes are leaf nodes. The diagnosis node will contain the training models for disease diagnosis of the k diseases that this 2^{nd} level node represents. Each training model has an associated disease token (which is a secure Bloom filter that contains hash values of a disease keyword with its typos) for the public cloud to determine which training model to use when it processes a query. Similarly, the complication prediction node contains k training models, each for predicting potential complications which may arise of a particular disease. Finally, the treatment node contains training models for k diseases, one for each illness. Each training model represents an aggregated model constructed by the public cloud using encrypted information sent by each hospital. The training model is built using patients' profiles, disease treatments, laboratory tests, etc, and is used to assess the best treatment option for a particular patient based on his personal profile, and/or laboratory test results.

As for the clients, they may use their personal profiles and lab tests results to generate search requests. After receiving an encrypted search request, the public cloud server first finds a matched category in the 1^{st} level category nodes. Then, the server will only search for matching results among the child nodes of that best matched 1^{st} level category node. This can significantly reduce the search time because the server merely searches information within this relevant sub-tree structure, only a subset of the whole information collection. The server goes through the child nodes of this selected category node to find $k = 2$ best matched 2^{nd} level nodes. Next, based on the query identifier, the server randomly selects one of the k matched level 2 nodes, and traverses into its sub-tree structure based on the query type, e.g., diagnosis, complication or treatment. After finding the matched leaf node, the cloud server will return the answers using the appropriate training model for that query. For example, based on the disease token and query type, the cloud server selects the appropriate training model to see if a client has suffered this disease or predict potential complications that may arise or the treatment options for this particular disease based on

that client's unique profiles and laboratory test results.

2.3.2 Detail Design of PDTCPS

We present more detailed descriptions of the proposed scheme in this subsection.

(1) Index Tree Construction

The public cloud constructs a keyword based encrypted index tree which allows authorized clients to conduct searches for health related questions based on their individual profiles and lab tests results.

Here, we describe how the encrypted index tree is constructed. In our design, we use $SHA-256$ as our keyed hash function, and use L-bit Bloom filters to handle typos.

1. Operations performed by the TA:

(i) In the initialization phase, a secret key $SK = \{K_O, K_B, K_A\}$ is produced by the trust authority where (a) K_O is a symmetric key for OPE operation; (b)K_B is the generation key for the Bloom filter generation; (c) K_A is the key used for computing key hash values of category keywords.

(ii) Then, TA generates a set of key hash values of category keywords, $Enc(CW) = \{Enc(cw_1), Enc(cw_2), \cdots\}$ which will form the 1^{st} level nodes.

(iii) For every category i, TA also produces a set: $\widetilde{W_i} = \{a_{i1}, a_{i2}, \cdots, a_{i|\widetilde{W_i}|}\}$, where a_{ij} is a disease keyword belonging to category i. Next, for each keyword a_{ij}, the TA generates a fuzzy keyword set: $\{a_{ij_1}, a_{ij_2}, \cdots\}$, where a_{ij_z} is a single-typo keyword of a_{ij}. The TA inserts the keyed hash values of all relevant disease keywords and their associated fuzzy keyword sets into a L-bit Bloom filter, $bf(\widetilde{W_i})$, using the secret key K_B.

(iv) In addition, TA determines the number of children nodes, $|C_i|$ for each category node i and forms the set $\widetilde{S} = \{|C_1|, |C_2|, \cdots, |C_{|\widetilde{S}|}|\}$. Then, for each child node (e.g., the j^{th} child node of the i^{th} category), it stores a keyword set D_{ij}, which contains k disease keywords. Next, TA generates a Bloom filter $bf(D_{ij})$, which contains those k keywords as well as their associated fuzzy keywords. Our solution inserts the same disease into k different 2^{nd} level nodes so that the cloud server can go through k different nodes (based on query identifiers) to find a matched leaf node even with the same keyword search request. Thus, both the search and path patterns can be hidden from the cloud server.

(v) Finally, TA delivers all the generated ciphertexts including $\{Enc(CW), \widetilde{S}, \{BFD(\widetilde{W_1}), \cdots, BFD(\widetilde{W_{|S|}})\}\}$, where $BFD(\widetilde{W_i}) = \{I_{cw_i}, bf(\widetilde{W_i}), \{bf(D_{i1}), bf(D_{i2}), \cdots, bf(D_{i|C_i|})\}\}$ and I_{cw_i} is a category index, to the cloud server.

(vi) It also sends both encrypted category keywords, $Enc(CW)$ and the secret key SK to every hospital. The secret key SK is also sent to all authorized clients.

2. Operations performed by hospitals:

(i) Every hospital H_m contains a category set $\widetilde{CW_m} = \{kw_1, kw_2, \cdots, kw_{|\widetilde{CW_m}|}\}$ and a disease keyword set $\{G_m(kw_1), G_m(kw_2), \cdots, G_m(kw_{|\widetilde{CW_m}|})\}$, where $G_m(kw_i) = \{b_{i1}, b_{i2}, \cdots, b_{i|G_m(kw_i)|}\}$ and b_{ij} is a disease keyword.

(ii) Then, hospital H_m generates $Enc(CW_m)$ which consists of all key hash values of category keywords in CW_m.

(iii) For each illness b_{ij} which H_m has relevant patients' information, it also generates $bf(b_{ij})$ using the secret key K_B.

(iv) Next, H_m uses a classification method to extract the training feature set for that illness b_{ij}, denoted as $\widetilde{sv}(b_{ij})$. Later, it encrypts this training feature set using OPE and the K_O key to produce $\widetilde{SV}(b_{ij}) = OPE_{K_O}(\widetilde{sv}(b_{ij})) + r_{ij}$, where r_{ij} are some random value sets. The random values are added to ensure the participating hospitals cannot uncover the true values of these feature vectors each hospital sends even if some hospitals collude with the cloud server.

(v) Finally, hospital H_m sends $\{Enc(\widetilde{CW_m}), \{BSV(G_m(kw_1)), \cdots, BSV(G_m(kw_{|\widetilde{CW_m}|}))\}\}$, where $BSV(G_m(kw_i)) = \{I_{kw_i}, \{bf(b_{i1}), \widetilde{SV}(b_{i1})\}, \cdots, \{bf(b_{i|G_m(kw_i)|}), \widetilde{SV}(b_{i|G_m(kw_i)|})\}\}$ and I_{kw_i} is the category index for category keyword kw_i, to the cloud server.

3. GenIndex:

(i) The cloud server first builds the 1^{st} level nodes, where the i^{th} category node stores the encrypted keyword $Enc(cw_i)$ and the corresponding Bloom filter $bf(\widetilde{W_i})$.

(ii) Then, the public cloud uses the received information to build the 2^{nd} level, where the j^{th} second level node of the i^{th} first level node stores the corresponding Bloom filter $bf(D_{ij})$.

(iii) For each 2^{nd} level node, the cloud server constructs three children nodes, one for each query type, i.e., "diagnose", "complication" and "treatment". An integer value can be

used to represent each query type.

(iv) For each received set of information from a hospital, the cloud server first uses the received $Enc(kw_i)$ to find the matched 1^{st} level category node. Then, it computes the inner product values between each of the received $bf(b_{ij})$ and the Bloom filters stored in the $|C_i|$ child nodes under the i^{th} category node to find the k matched 2^{nd} level nodes. Next, the cloud server traverses into their leaf nodes to find the right training model.

(v) If no training model exists, the newly received training model will be stored. If a training model already exists, the public cloud generates a new model by combining previously stored feature vectors for that disease with the most recently received feature vectors to generate a new training model.

With the procedures outline above, the cloud server finally constructs the encrypted index tree, which is shown in Fig 2.6.

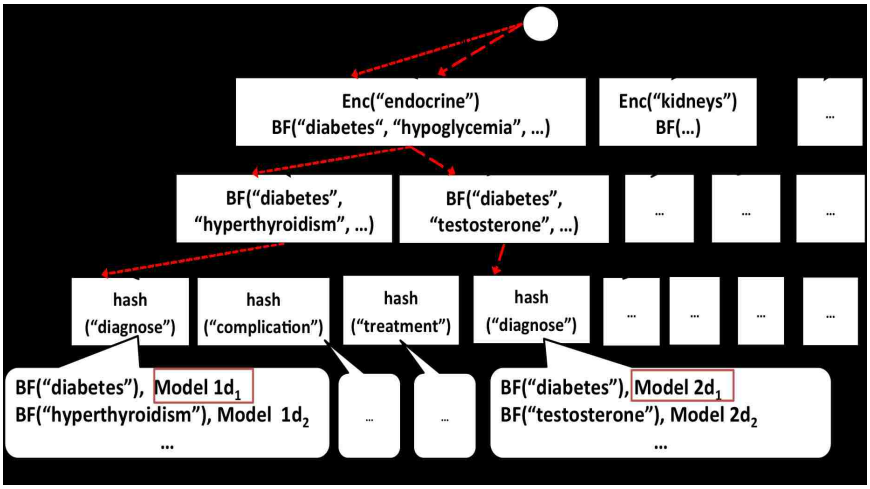


Figure 2.6: PDTCPS Index Tree Structure

(2) Query Generation

To provide query unlinkability, we need to generate a different search request even for the same keyword query.

(i) Given a query Q=$\{cw_q, (F_1, \cdots, F_i, \cdots), x_q, t_q\}$, where cw_q is the category keyword, F_i is either a personal attribute of a client or his lab test result i, x_q is the disease keyword and t_q is the query type.

(ii) An authorized client first generates a random query id ID_q and a keyed hash value

of the category keyword as $Enc(cw_q)$.

(iii) Each F_i will be encrypted as $OPE_{K_O}(F_i)+R_i$ using the received encryption key K_O and a random value R_i. This step ensures that the same F_i results in different encrypted value and hence provides query unlinkability.

(iv) The client also generates a fuzzy keyword set: $\{x_{q_1}, \cdots, x_{q_i}, \cdots\}$, where x_{q_i} is a single-typo keyword of x_q. Then, the client generates $bf(x_q)$ using the secret key K_B.

(v) Finally, the encrypted search request $Enc_{SK}(Q)$=$\{ID_q, (OPE_{K_O}(F_1) + R_1, \cdots, OPE_{K_O}(F_i) + R_i, \cdots), Enc(cw_q), bf(x_q), hash(t_q)\}$, is submitted to the cloud server.

(3) Search Process

(i) Upon receiving the search request $Enc_{SK}(Q)$, the server first checks if $Enc(cw_q)$ can be matched with the stored encrypted keywords $Enc(CW)$ in the 1^{st} level nodes.

(ii) If it is not found, then the cloud server computes the inner products of Bloom filter $bf(x_q)$ in the query with $|\widetilde{S}|$ Bloom filters stored in the 1^{st} level nodes. The one with the best match will be the selected 1^{st} level node. (Fig 2.7)

(iii) Next, the cloud server searches through the child nodes of this selected 1^{st} level category node by performing the following operations:

• Compute the inner product values between the $bf(x_q)$ and the stored $|C_i|$ Bloom filters in the 2^{nd} level nodes.

• Find the top k nodes among those $|C_i|$ nodes in the second level.

• Next, select one of k matched 2^{nd} level nodes, node j, using $j = ID_q \ mod \ (k)$ and travels into its sub-tree nodes based on the query type.

(iv) After finding the matched leaf node, the cloud server can find the appropriate training model to diagnose disease, predict potential complications or determine the best treatment options for a client based on his query type, t_q.

2.4 Security Analysis

In this subsection, we analyze the privacy characteristic of $PDTCPS$ against possible attacks by various entities involved in our system. Adversaries in our system could be participating hospitals, network eavesdroppers, or even the cloud server. For instance, hospitals

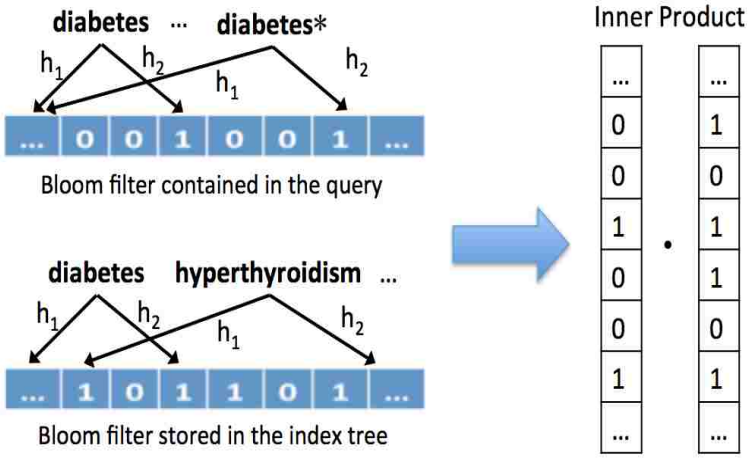


Figure 2.7: Inner Product Computation

and the cloud server in our system are assumed to be semi-trusted, implying that they follow the protocol execution, but may attempt to learn additional information. A network eavesdropper could have the resources to monitor all messages in the network or the messages sent by a particular hospital or a client.

2.4.1 Network Eavesdroppers

For network eavesdroppers, *PDTCPS* achieves privacy preserving mainly via encrypted communication. Our designed scheme guarantees that attackers cannot uncover any knowledge of any content within the ciphertext as long as eavesdroppers cannot obtain the cryptographic keys. In addition, by adding randomness in each encrypted query, the adversaries cannot conduct frequency analysis to gain additional information about submitted queries and hence no sensitive information is revealed.

2.4.2 Semi-honest Hospitals

In the presence of semi-honest hospitals, our scheme achieves information-theoretic security. Specifically, our design adds some randomness to the encrypted training features generated by each hospital before they are being sent to the cloud server. Thus, participating hospitals cannot gain additional information on the ciphertext sent by other hospitals even if they know the secret key.

2.4.3 Semi-honest Cloud Server

In this part, we will show how $PDTCPS$ satisfies several search privacy requirements:

• Index and Query confidentiality under both the known ciphertext model and the known background model. More details are provided in subsequent subsections.

• Query unlinkability: Our $PDTCPS$ generates different search requests even with the same query keyword and hence provides query unlinkability to a certain extent.

• Hiding access pattern: Our design ensures that the cloud server traverses different nodes to find a match even with the same keyword search request and hence the access patterns are hidden from the server.

Security Analysis of PDTCPS Under the Known Ciphertext Model

Here, we adapt the simulation-based security model in [144] to prove that our scheme can be secure under the known ciphertext attack. Before proving, we first introduce some notations that will be used in the proving process.

• History: It is an index set I and a query set $Q = \{Q_1, Q_2, \cdots\}$, denoted as $H = (I, Q)$.

• View: The cloud server can only see the encrypted form of a H, denoted as $VI(H)$, including the secure indexes $Enc(I)$ and the encrypted search requests $Enc(Q) = \{Enc(Q_1), Enc(Q_2), \cdots\}$.

• Trace: A trace is a set of queries, denoted as $Tr(H) = \{Tr(Q_1), Tr(Q_2), ...\}$. $Tr(Q_i)$ captures the information for each query Q_i including the search pattern PA_{Q_i}, and the outcome of the search RE_{Q_i} which is available to the cloud server to gain additional information.

As in [144], our proof is based on the following argument: given two histories that produce the same trace, if the cloud server cannot distinguish which history is produced by the simulator, then the cloud server cannot learn additional knowledge beyond the information that the system is willing to leak.

We adopt a simulator that can simulate a view $VI(H)'$ indistinguishable from cloud server's view $VI(H)$. The simulator works as follows:

1. For the encrypted query $Enc(Q_1)$, the simulator generates $Enc(Q_1)'$ as follows:

(i) The simulator first selects random strings $\{s_1, s_2, s_3\}$, where $s_i \in \{0,1\}^U$, and then sets $ID'_{Q_1} = s_1$, $hash(cw'_{Q_1}) = s_2$ and $hash(t'_{Q_1}) = s_3$ separately. U is the length of hash-value, ID'_{Q_1} is the query identifier, and cw'_{Q_1} is the category keyword in that query and t'_{Q_1} is its query type.

(ii) The simulator also generates a L'-bit vector $v \in \{0,1\}^{L'}$ and sets $bf(x'_{Q_1}) = v$ where x'_{Q_1} mimics the disease keyword in the query.

(iii) Next, the simulator builds a vector which represents encrypted attributes used in the query, $Enc(F'_{Q_1}) = \{G_1, G_2, \cdots\}$, where G_i is a random string chosen from $\{0,1\}^U$.

(iv) After the above steps, the following encrypted query, $Enc(Q_1)'$=$\{ID'_{Q_1}$, $Enc(F'_{Q_1})$, $hash(cw'_{Q_1})$, $bf(x'_{Q_1})$,$hash(t'_{Q_1})\}$, is simulated.

2. Based on the search pattern PA_{Q_1}, the simulator can generate the $Enc(I)'$ as follows:

(i) Let us assume PA_{Q_1} goes through category node $ca(1)$, intermediate node $im(1)$ and leaf node $ln(1)$ of the index tree.

(ii) The simulator first sets the $Enc(ca(1))' = hash(cw'_{Q_1})$.

(iii) Then, the simulator adds $bf(x'_{Q_1})$ to $bf(ca(1))'$.

(iv) The simulator also sets $bf(\widetilde{D'}_{Q_1})' = bf(\widetilde{D'}_{Q_1})' + bf(x'_{Q_1})$, where $|\widetilde{D'}_{Q_1}| = k$ and $\widetilde{D'}_{Q_1}[(ID'_{Q_1})mode(\mathrm{k})] = im(1)$.

3. For subsequent queries such as Q_j with search pattern PA_{Q_j} which goes through category node $ca(j)$, intermediate node $im(j)$ and leaf node $ln(j)$ of the index tree where 2 $\leq$ i $\leq j \leq |Q|$, the simulator does the following:

(i) If $ca(j)$, $im(j)$, $ln(j)$ are not same as $ca(i)$, $im(i)$, $ln(i)$, then the simulator repeats the same process as simulating $Enc(Q_1)'$ and $Enc(I)'$.

(ii) If $ca(j)$ is the same as $ca(i)$ but $im(j) \neq im(i)$, then the simulator repeats the same process as simulating $Enc(Q_1)'$ and $Enc(I)'$ with the condition that $hash(cw'_{Q_j})$=$hash(cw'_{Q_i})$.

(iii)If $im(j)$ is the same as $im(i)$ but $ln(j) \neq ln(i)$, then the simulator sets $hash(t'_{Q_j}) \neq hash(t'_{Q_i})$ and also generates all the other necessary information.

(iv) If the search pattern PA_{Q_j} ends at the same leaf node $ln(i)$ as the previous query Q_i, then the simulator sets $hash(t'_{Q_j}) = hash(t'_{Q_i})$ and does the following:

• If the search result RE_{Q_j} for the query Q_j, is not the same as RE_{Q_i}, then the simulator repeats the same process as simulating $Enc(Q_1)'$ and $Enc(I)'$ with the condition that the

$Enc(F'_{Q_j})$ is different from the $Enc(F'_{Q_i})$.

• If the search result is the same, then the simulator sets $bf(x'_{Q_j}) = bf(x'_{Q_i})$ and generates the $Enc(F'_{Q_j})$, which is similar to the $Enc(F'_{Q_i})$.

4. After all the queries have been simulated, the simulator does the following:

• It converts each $bf(ca(i))'$ and $bf(im(i))'$ into L'-bit $\{0,1\}^{L'}$ vectors by replacing the elements bigger than 1 with 1.

• It adds $Enc(F'_{Q_i})$ to the training feature set $\widetilde{sv}(x'_{Q_i})$ to make sure that the query result is the same as RE_{Q_i}. Note that the training feature set $\widetilde{sv}(x'_{Q_i})$ is attached to the appropriate simulated leaf node.

5. The simulator outputs the view $VI(H)'$=($Enc(I)'$, $Enc(Q)'$).

In summary, the $Enc(I)'$ and $Enc(Q)'$ can generate the same trace as the one that the cloud server has. Thus, we claim that no probabilistic polynomial-time (P.P.T) adversary can distinguish between the view $VI(H)'$ and $VI(H)$.

Security Analysis of PDTCPS Under the Known Background Model

In this part, we analyze the security of $PDTCPS$ under the known background attack model. For each query Q_i we generate the encrypted search request as follows: $Enc(Q_i)$=$\{ID_{Q_i}$,$Enc(F_{Q_i})$,$hash(cw_{Q_i})$,$bf(x_{Q_i})$,$hash(t_{Q_i})\}$. Since a random value set is introduced during the query generation, $PDTCPS$ produces different search requests even for the same query. Thus, our scheme can achieve query unlinkability such that it makes it hard for the cloud server to link one transformed request to another even if both contain the same keyword.

In addition, since in the known background model, the cloud server can deduce the statistical information by analyzing the search and path patterns for each query. Thus, it is important to hide those information from the cloud. In our scheme, we have extended every 2^{nd} level node to contain k different keywords so that the cloud server randomly selects one of the k matched nodes containing the desired keyword. Therefore, both the search and path patterns can be hidden from the cloud server.

Discussion: Since OPE is a deterministic encryption, so it is subjected to two known security vulnerabilities, namely (i) frequency-based attack where adversaries use frequency

distributions of ciphertext and plaintext to infer their correspondence and (ii) order-relations among plaintexts where attackers can easily break the encryption via sorting of known values of plaintexts and ciphertexts using domain knowledge. However, since we have added random values in $PDTCPS$, attackers simply cannot infer such information. Thus, our design is safe against both frequency and domain attacks.

2.5 Performance Evaluation

We implemented PDTCPS and conduct our experiments on a Mac Pro with an Intel Core i5 processor running at 2.6GHz and 8GB memory. The following performance metrics are used to evaluate our scheme ($PDTCPS$) and two other proposed solutions, namely the CAM [105], and the hyperplane decision-tree based scheme ($HDBS$) [33]:

• Index construction time, which is the time incurred in generating the proposed index tree structure;

• The generation time, testing time, accuracy, communication and storage costs of the training model;

• The accuracy of the search results.

First, we select $|\widetilde{S}|$ categories based on the major categories in Medical Health provided in the medical website, (e.g., Endocrine, Intestinal, Throat etc), as the 1^{st} level nodes of the index tree.

Then, based on these $|\widetilde{S}|$ categories, we extract $\sum_{i=1}^{|\widetilde{S}|} |C_i|$ unique disease keywords, e.g., Endocrine includes all diseases which affect the endocrine system such as diabetes, hypothyroidism, hyperthyroidism and so on. Next, we map all these $\sum_{i=1}^{|\widetilde{S}|} |C_i|$ distinct keywords into their appropriate categories and build the encrypted index tree, where each leaf node represents k disease keywords. We also set k=2, the length of the Bloom filter, L, to $64bytes$, and use h=2 hash functions to insert keywords and their associated fuzzy keyword sets to a Bloom filter in our $PDTCPS$ scheme.

2.5.1 Construction and Communication Costs For Index Tree

The index construction process contains two major steps:

• TA generates sets of encrypted information including the encrypted category keywords, Bloom filters, and the number of children that under each category node. Then, it sends these Bloom filters to the public cloud.

• After receiving the information above, the public cloud stores all these information in the index tree.

This index construction cost is only a one-time computation cost. Since the encrypted index tree contains $|\widetilde{S}|$ category nodes and $\sum_{i=1}^{|\widetilde{S}|} |C_i|$ second level nodes, the TA needs to generate $|\widetilde{S}|$ encrypted category keywords and $|\widetilde{S}|+\sum_{i=1}^{|\widetilde{S}|} |C_i|$ Bloom filters(BFs). Fig 2.8(a) shows the generation cost for a L-bit Bloom filter and from the results we can see the generation cost increases linearly with the number of inserted keywords. In addition, it needs to send all these $|\widetilde{S}| + \sum_{i=1}^{|\widetilde{S}|} |C_i|$ Bloom filters to the public cloud to be stored in the encrypted index tree. Since the results in our system show a linear relationship between the time and the number of disease keywords, so the realistic overhead of our system will increase linearly according to the number of disease keywords. For example, base on the Dewey Decimal system, which is a library classification system, we can further cluster all the existing 30,000 human diseases into almost 60 categories. Assuming $|\widetilde{S}|$=60 and each top-level node has 500 child nodes, then the total computational cost can be computed as follows: it takes 0.39 ms to insert 60 fuzzy keywords into the BF of a child node and 0.39ms*500*30/60=95 ms to insert 500*30 fuzzy keywords into the BF of each category node. Thus, the overall index construction time for 30,000 diseases is 17 sec. Furthermore, to ensure the collision rate of BF at each category node to 1%, we need to use a BF of length 305 Kbytes. In addition, each child node contains 2 disease keywords where the average keyword length is 15. Thus, we need a 1.2 Kbytes Bloom filter for each child node to ensure a 1% collision rate. Therefore, the total one-time communication cost that TA incurs to send relevant information to the cloud for index tree construction of 30,000 diseases is (305*60+1.2*500*60)=53 Mbytes.

2.5.2 Training Model Evaluation

(1) Training Model Generation Method

As for the training model generation cost, we first describe the construction processes

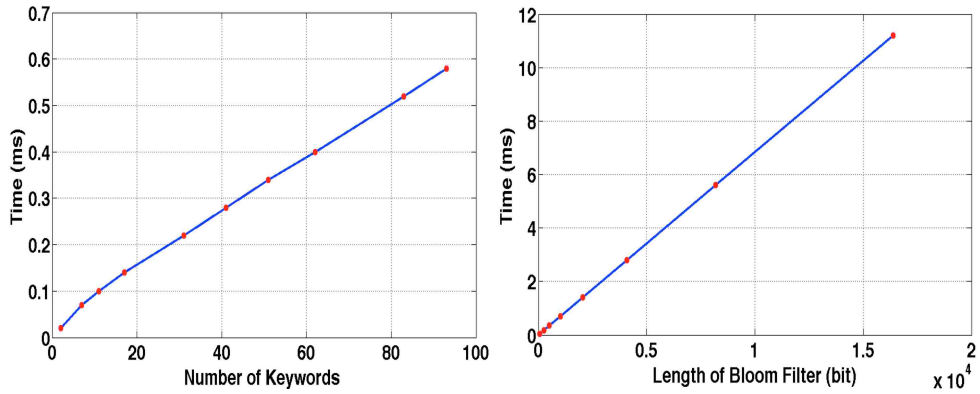


Figure 2.8: (a)Bloom Filter Generation Cost Vs Number of Keywords. (b)Inner Product Computation Cost Vs Bloom Filter Length .

for $PDTCPS$, CAM and $HDBS$ schemes. Since we do not have access to any dataset for complications and treatments, here we only evaluate the performance of our model for disease diagnosis.

(i) $PDTCPS$: For each disease associated with a leaf node, the cloud server generates a training model based on the received training feature sets from all hospitals. Here, we use the parallel SVM method to construct an aggregate SVM model. By having each hospital conducts its own data mining and sends only encrypted support vectors makes our solution more efficient and scalable.

(ii) CAM [105]: This scheme uses a parallel histogram-based decision tree algorithm to generate the training model where every iteration consists of an updating phase performed simultaneously by all the hospitals and a merging phase performed by the cloud server. At each iteration, a new layer is constructed as follows: each hospital compresses its share of the data using histograms and sends them to the cloud. Then, the cloud server merges the histograms and determines the best splits for each node in the decision tree, thereby constructing a new layer. Next, it sends this new layer to each hospital, and the hospitals construct histograms for this new layer. Finally, the cloud server can build the regression tree layer by layer through the iterations.

(iii) Hyperplane Decision Based Scheme ($HDBS$) [33]: This scheme introduces a sophisticated approach to perform machine learning on encrypted data. All hospitals send their encrypted datasets to the public cloud server. The public cloud server generates an aggregated training model based on all these encrypted datasets using homomorphic en-

cryption method. Next, the client generates an encrypted search and submits to the public cloud. The public cloud traverses the encrypted index tree as described before and sends relevant answers back to the client in encrypted form. The client then decrypts the returned response to obtain the answer.

(2) Training Model Performance

In this part, we conduct Exp 1 and Exp 2 to evaluate the performance of the above three schemes.

(A) Exp 1: In the experiment, we use the Pima Indians Diabetes Data Set from the UCI machine learning repository [18], which contains 768 instances with 9 attributes of two labeled classes. We first select 90% of the Pima dataset as training set, S_1, and the remaining 10% as the test set T_1. Then, based on the distribution (e.g., mean or standard deviation) of each attribute, we generate two synthetic datasets from S_1 denoted as S_{11} and S_{12}, where S_{11} contains 1384 instances, and S_{12} contains 4152 instances. Next, we partition the synthetic datasets as follows (i) we partition each synthetic dataset into m equal parts and assign each part to one hospital. (ii) Since in the real world different hospital may have different data size, so we also divide each synthetic dataset into m unequal parts, and assign each of them to one hospital.

(B) Exp 2: To ensure that the conclusions we draw from Exp 1 is reliable, we also use the Breast Cancer Wisconsin (Original) Data Set, which contains 699 instances with 10 attributes and two class labels, to conduct Exp 2. The same method used in Exp 1 is used to generate the dataset for each hospital.

After data generation, the hospitals then extract the training features from their assigned datasets and encrypt them using the OPE algorithm, where the encryption complexity is largely based on the bit length of each feature. For example, our experiments show that using only the 1^{st} 10 bits of the encrypted value produce similar prediction accuracy in disease prediction. The OPE algorithm takes 7.1ms to encrypt a 10-bit length feature. Thus, we only use the first 10 bits of the encryption value for all our experiments to reduce encryption time without affecting accuracy.

(C) Performance Evaluation: Table 2.1 & Table 2.2 show the evaluation results for all the above three schemes. Note that (i) the reported storage cost is the cost of storing one

Table 2.1: Training Model Evaluation for Exp1

Scheme	# of Features	# of Instances	Equal Data Size	# of Leaf Nodes	Training Time	Communicate Cost	Testing Time	Accuracy	Storage Cost
$PDTCPS$	9	1384	Yes		0.05s	8.09KB	0.024ms	81.6%	7.9KB
			No		0.05s	8.1KB	0.024ms	80.3%	7.7KB
		4152	Yes		0.13s	23.48KB	0.024ms	76.3%	21.8KB
			No		0.15s	23.50KB	0.024ms	80.3%	21.9KB
CAM	9	1384	Yes	48	2.0s	8.7KB	0.015ms	75.0%	0.18KB
			No	41	1.8s	7.9KB	0.014ms	75.0%	0.17KB
		4152	Yes	187	7.8s	28.1KB	0.026ms	68.4%	0.6KB
			No	178	6.9s	24.60KB	0.024ms	69.7%	0.58KB

Table 2.2: Training Model Evaluation for Exp2

Scheme	# of Features	# of Instances	Equal Data Size	# of Leaf Nodes	Training Time	Communicate Cost	Testing Time	Accuracy	Storage Cost
$PDTCPS$	10	1680	Yes		0.004s	1.01KB	0.027ms	92.9%	1.1KB
			No		0.005s	1.07KB	0.027ms	92.1%	1.2KB
		5040	Yes		0.011s	1.67KB	0.027ms	92.1%	1.5KB
			No		0.012s	1.72KB	0.027ms	90.6%	1.5KB
CAM	10	1680	Yes	24	1.5s	9.6KB	0.010ms	86.4%	0.06KB
			No	20	1.3s	8.7KB	0.09ms	88.5%	0.05KB
		5040	Yes	42	5.8s	20.8KB	0.015ms	84.3%	0.1KB
			No	36	5.5s	18.9KB	0.013ms	86.4%	0.09KB
$HDBS$	10	699			0.032s	35.84KB	151.1ms		

[1]The results of $HDBS$ are extracted from [33], which scaled to the same CPU environment of our scheme;

training model for a particular disease, and (ii) no $HDBS$ result is reported for the diabetic dataset because we have no access to their codes, and they did not have published results using that dataset. One can see that the training time for our scheme ($PDTCPS$) is much smaller than the CAM and $HDBS$ schemes described in [105, 33]. The CAM scheme is inefficient since it needs multiple interactions between hospitals and cloud server to generate the aggregated decision-tree, which greatly increases the training cost. $HDBS$ uses the aggregated dataset for SVM training while $PDTCPS$ uses parallel SVM for training, hence $HDBS$ incurs more training time than $PDTCPS$.

Table 2.1 & Table 2.2 also show that our training model generation process incurs smaller communication cost than the CAM and $HDBS$ schemes.$PDTCPS$ incurs the least cost because the hospitals only need to send the encrypted training features instead

of all the instances to the cloud server. Whereas in the CAM scheme, the communication cost is largely due to the histograms that are sent by the hospitals. Meanwhile, in order to transform the ciphertext from one encryption form into another, the $HDBS$ scheme requires multiple interactions between a client and server, which incurs much communication cost. The tables also show that by using SVM rather than decision tree, $PDTCPS$ achieves higher accuracy than CAM.

In addition, from Table 2.1 & Table 2.2, we can see that the CAM scheme incurs less test time than $PDTCPS$ when the dataset size is small. This is expected because the test time for the CAM scheme is largely based on the height of the decision tree. Thus, when the dataset is small, the height of the decision tree is also small, which leads to low test evaluation cost. Whereas in $PDTCPS$, the number of instances in the dataset has little impact on the test evaluation cost since it only depends on the number of attributes. $PDTCPS$ only incurs about 0.035Kbytes for Exp1 and 0.04Kbytes for Exp2 to store a training model. However, to increase the efficiency for future training model updates, we may also store the encrypted training feature sets. The reported storage cost for $PDTCPS$ in Table 2.1 & Table 2.2 shows the storage cost incurred when such feature sets are also stored.

2.5.3 Search Evaluation

In this subsection, we evaluate the search performance of $PDTCPS$.

(1) Search Over Encrypted Index

The search operation executed at the cloud server side consists of the inner product calculation for the nodes contained in the index tree. If a node contains the keyword(s) in the query, the corresponding bits in both Bloom filters will be 1 thus the inner product will return a high value. Figure 2.8 (b) shows that the inner-product computation time grows linearly with the length of the Bloom filter. This is intuitive because the cloud server needs to go over all the bits in Bloom filters before computing the final inner product values. Assuming that there are $|\widetilde{S}| = 60$ categories, and each category has 500 diseases, then on the average, a query without an encrypted category keyword needs to search through 30 top level category nodes and 250 2nd-level nodes, then the average

search time will be (30*1.7+250*0.0067)=53s since each inner product computation takes 6.7 ms with $L = 1.2Kbytes$ and 1.7s with $L = 305Kbytes$. However, the search time is only about (250*0.0067)=1.675s with an encrypted category keyword included in the query. The search time for queries without category keywords can be reduced by using the bed-tree structure [52] to create more hierarchy for category keywords so that fewer category nodes need to be searched.

(2) Search Accuracy

In our experiment, we adopt the widely used performance metric, namely false positive, denoted as FP, to measure the search result accuracy. The false positive rate of a L-bit Bloom filter with h hash functions can be computed as $(1 - \frac{1}{L})^{nh}$, where n is the number of keywords inserted into that Bloom filter. The number of the inserted keywords in a Bloom filter for a disease keyword can be computed as $n = 2 * l_i + 1$, where l_i is the number of characters of that disease keyword w_i.

Figure 2.9(a) shows how the false positive rate of our scheme varies as the number of inserted keywords changes when $L = 1.2Kbytes$. One observation is that the false positive is very low when l_i is small, i.e. 0.6% at $l_i = 15$ which is the average character length of our disease keywords.

Figure 2.9 (b) shows the performance of our scheme when the length of a Bloom filter is varied. Although large Bloom filter can better reduce the false positive rate, it may increase both the search time and the storage cost (since the cloud server needs to store these Bloom filters in the index tree). Thus, there is a trade-off among the false positive rate, search time, and the storage cost of our scheme. For example, we can tune the parameters, i.e. L, l_i, to specifically fit a particular accuracy and storage requirements.

Therefore, the total accuracy can be further computed by combining both false positives in the searching model and error rates in the prediction results as follows:

$$Acc_{total} = 1 - (P_f + (1 - P_f) * C_f) \tag{2.1}$$

where P_f is the false positive for the searching model and C_f is the error rate for the training model. Typically, the Bloom filters are designed to achieve a $P_f = 1\%$ and C_f as shown in

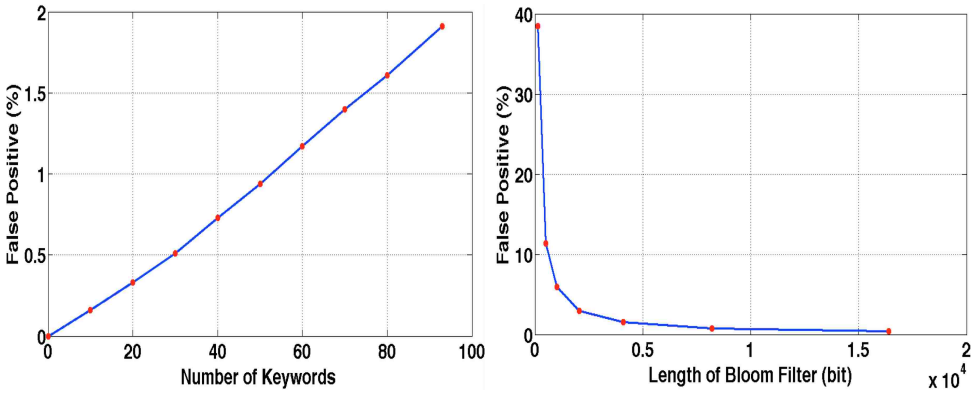


Figure 2.9: False Positive Rate of the Bloom filter. (a) With varying numbers of keywords. (b) With various Bloom filter lengths.

Table 2.1 & Table 2.2 ranges from 76.3 to 92.9%.

2.6 Summary

In this chapter, we have proposed a Privacy-Preserving Disease Treatment, Complication Prediction Scheme ($PDTCPS$), which allows users to conduct privacy-aware searches for health related questions based on their individual profiles and lab tests results. Our design also allows healthcare providers and the public cloud to collectively generate aggregated training models to diagnose diseases, predict complications and offer possible treatment options. In addition, to enrich search functionality and protect the clients' privacy, our scheme can support fuzzy keyword search and query unlinkability. Moreover, $PDTCPS$ also hides access patterns and hence addresses the security threat via exposed access patterns identified in previous searchable encryption schemes. Finally, we validate the practicality of our scheme by evaluating our scheme using two UCI datasets. The results show that $PDTCPS$ is secure against different adversarial situations, and has better performance than two existing schemes.

Chapter 3

Incentivizing High Quality Crowdsourcing Data For Disease Prediction (IHESS)

3.1 Background

With the growing need for personalized medical treatments to lower healthcare cost, and the ease of storing large scale clinical datasets, researchers are keen on developing data mining methodologies to mine hidden patterns such that they can identify critical factors which affect disease progression and survival rates and use such information to aid the healthcare professionals in making better treatment decisions.

However, discovering knowledge [58, 117, 133] in healthcare systems is a herculean yet critical task since those available raw medical data are widely distributed, heterogeneous in nature, and voluminous. For example, the Amyotrophic Lateral Sclerosis (ALS) is a fatal neuro-degenerative disease with significant heterogeneity and can lead to muscle weakness which gradually impacts the functioning of the body, leading to eventual death. While Riluzole [3] is the only approved medication for ALS, it merely increases the survival duration of a patient by a few months and has no effect in improving his quality of life. Thus, the identification of key factors that affect ALS disease progression and survival rate is

important for effective medical interventions and for appropriate stratification in clinical trials [28].

With this dire need for *ALS* prediction tools in mind, a crowdsourcing competition, the DREAM *ALS* Stratification Prize4Life Challenge [4] was held, where solvers were asked to use 3 months of individual patient level clinical trial information to predict that patient's disease progression over the subsequent 9 months. The purpose of this challenge is to enable better understanding of patient profiles and find a more accurate way of predicting patients' disease progression and survival rates.

While clinical data mining technology helps to identify hidden patterns within patients' heathcare data which can aid in developing relevant treatment tools for personalized treatment, many current diagnostic and prognostic tools make decisions based on only a small number of patient characteristics. For example, many cardiologists and critical care physicians believe that the direct measurement of cardiac function provided by Right Heart Catheterization (RHC) is enough to guide treatments of certain critically ill patients. However, there are significant limitations to this type of assessment. Since in the catheterization laboratory, hemodynamic variables are typically measured at rest with patients in the supine position, which may not only underestimate the presence and severity of Pulmonary Hypertension (PH) but such measurements also do not accurately reflect the true extent of hemodynamic compromise. In addition, the collected data sometimes have missing and noisy values. Thus, data mining researchers need to ensure that high quality data is available for training any disease prediction model. For higher accuracy model, a large dataset consisting of sufficient numbers of different categories of patients, e.g. fast, slow and average *ALS* patients, needs to be available. Therefore, hospitals need to be encouraged to share high quality data such that healthcare data mining researchers can produce better models.

In this chapter, we tackle such challenges by developing two tools: first, we propose a data cleaning & feature selection method which allows healthcare data mining researchers to clean up the available large scale dataset and identify relevant features which are critical in predicting the progression and survival rate of any given disease. Secondly, we propose an incentive mechanism which encourages hospitals to share truthfully high quality data which can then be aggregated to generate prediction models with higher accuracy rates.

We demonstrate the effectiveness of our approaches using three large datasets from three population-based studies, namly *ALS* [1], *RHC* [2] and STAR*D [122]. Our experimental results show that our prediction models yield good performance in *ALS* slope, *ALS* & *RHC* survival, and STAR*D relapse predictions. Furthermore, we also prove that our incentive mechanism satisfies two desirable properties: individual rationality and platform profitability.

In summary, our contributions can be summarized as follows:

- We develop new learning models by combining data cleaning & feature selection methods with effective machine learning techniques.
- We also design an incentive mechanism to encourage hospitals to share higher quality data, so that the cloud server can generate more accurate models for clinical use.
- We provide extensive experimental results using both $PRO-ACTALS$ and RHC datasets to show that our learning models can perform better than some existing prediction tools (e.g. the Cox-survival regression model). In addition, we show that our incentive mechanism have individual rationality and platform profitability properties.

3.2 Problem Formulation

In this subsection, we first describe our system model. Then, we outline our design challenges and design goals.

3.2.1 System Model

Our incentive based high-quality eHealth information sharing scheme ($IHESS$) consists of two parties: the hospitals and the cloud server, as shown in Fig 4.3. We assume that there is a 3^{rd} party service provider which utilizes the public cloud server to provide various disease prediction models to participating hospitals based on the datasets provided by these hospitals.

- Hospitals: Hospitals first collect patients' health records with certain costs, where patients take some tests to produce certain data such as blood sugar and cholestrol

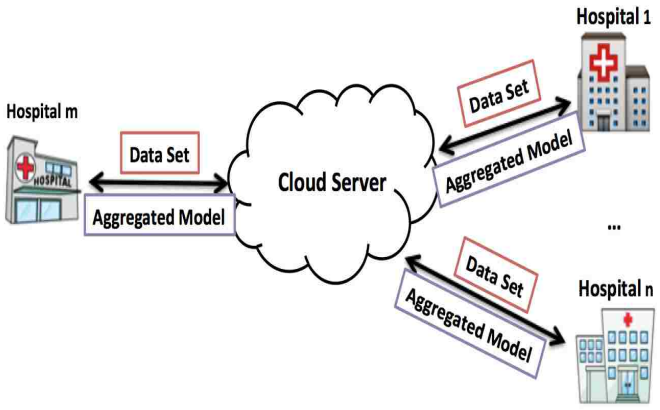


Figure 3.1: System Model

levels. They also store those data, which are typically noisy with missing values in certain fields in their private clouds. Then, the private cloud severs use a data cleaning & feature selection method to process and clean those health records. Next, they perform data mining operations over the cleaned data to generate locally trained models. However, since each hospital only has limited number of patients, its predictions may not be very accurate. In order to obtain more accurate results, the hospital servers may upload certain subsets of their health records to a public cloud run by a 3^{rd} party service provider so that the public cloud server can generate more accurate aggregated models for every participating hospital. While such aggregated model service is beneficial, hospitals may expose themselves to potential privacy threats by sharing their data. Thus, hospitals would not be interested in sharing their data, unless this aggregate service platform provides protection over potential privacy breach and the hospitals also receive sufficient rewards to cover their cost for collecting such data. To address privacy concern, the design in [166] can be used by hospital servers to encrypt their cleaned data before sending them to public cloud and data mining can be performed on encrypted data. Details are not discussed here for privacy design is not the focus of this work. In this chapter, we focus more on how different hospitals can be incentivized to clean up their data so that useful aggregate models can be generated from their high quality data.

- Public Cloud Server: After receiving the data, the public cloud server selects a subset of participating hospitals to generate aggregated models based on an incentive mechanism, e.g., the newly selected hospital has a nonnegative utility and provides data that can increase the utility of the aggregated service platform. The public cloud server also needs to define appropriate reward functions for the selected hospitals as well as utility function to evaluate the tradeoffs between the cost of rewarding selected hospitals, and the improvement in precision and recall of the new aggregated models.

3.2.2 Design Challenges

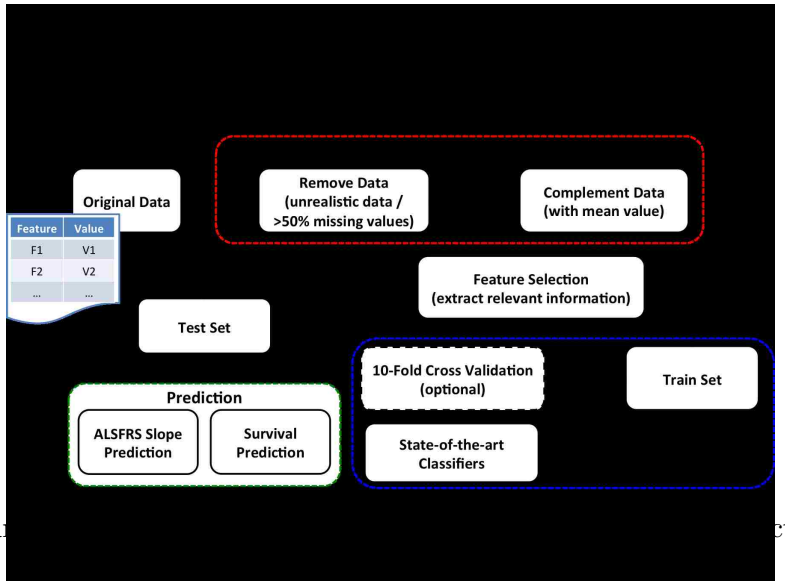
Although data mining has been increasingly used in the biomedical research community, it is typically a complex process involving intensive manual and computational tasks to identify hidden patterns which can aid treatment decisions from large medical datasets [140]. Such medical datasets typically are collected by healthcare organizations as they provide individual patient care. Thus, such raw datasets typically contain many features and have missing or non-numerical values and hence need to be cleaned before they can be used to generate aggregated training models.

In addition, the collection of medical data for a defined goal is still difficult even in this era of data explosion. While sharing clinical data with a third party cloud provider is useful for clinical data mining, this third party is unaware of the cost incurred by an individual hospital in collecting the various health records. Thus, it is difficult to define a reward function which can encourage hospitals to not only truthfully share data but also allow the public cloud server to distinguish high quality data from low quality data.

3.2.3 Design Goals

Our $IHESS$ is designed with the following goals in mind:

(i) Data Cleaning & Feature Selection: Since prediction results are highly dependent on the completeness of values in the data records and the selected features in the dataset, so we have designed a data cleaning & feature selection method to refine the data before it is being used to generate aggregated training models.



Figu ctions

(ii) Incentive Modeling: We have designed an incentive model to encourage high-quality data sharing with two desirable features:

- Individual Rationality: Every participating hospital whose submitted data have been accepted will have a nonnegative utility. In other words, the rewards that the selected hospitals receive will be greater than their resource consumption (e.g., lab tests costs).
- Platform Profitability: The utility of the cloud server should increase when including new data. In other words, the benefits brought by the participating hospitals should be larger than the total rewards paid to those hospitals.

3.3 Data Cleaning & Feature Selection

In this subsection, we outline the key stages in our data cleaning and feature selection process (DC&FS): (i) raw data processing; (ii) imputation of missing values; (iii) utilizing feature selection methods to identify and extract important features; (iv) constructing learning models for performing accurate predictions. The work flow of our proposed methodology to create useful training models for disease progression and survival rate predictions is shown in Fig 3.2.

3.3.1 Data Cleaning

The goal of the data cleaning step is to remove inconsistent and noisy data and impute missing data values. The data cleaning operation consists of two steps:

- Removing unrealistic values
- Imputing missing values

In the first step, we remove clinical data, which have unrealistic feature values (e.g. unrealistic age or weight). In the second step, we first retain only certain features, where fewer than 50% of their values are missing. Then, we replace any missing value with the average value obtained from non-missing entry values of that feature. The impact of data quality on the performance of prediction models will be discussed in section 3.4.

3.3.2 Feature Selection

After data cleaning, we then apply the following two feature selection techniques to assess which features are more helpful in constructing clinical prediction models.

- Random Forest (RF) [35, 97]: this is an ensemble classifier based on randomized decision trees and provides different feature important measures, which can be visualized by the Gini index scores [137, 71]. This feature importance score provides a relative ranking of the spectral features and can be used as a general indicator of feature relevance. Here we run random forest classifier on all features (whose missing rate $<$ 50%) and select useful features based on their Gini index scores.
- Logistic Regression (LR) [63]: this uses maximum-likelihood estimation to compute the coefficients for all features, which can be used to rank them based on their relative importance.

As a classifier, random forest method performs an implicit feature selection, using a small subset of important variables for the classification, leading to its superior performance on high dimensional data. However, logistic regression works well when the number of features is limited (e.g., fewer than 100) since it is easy to calculate the coefficient values for all

Table 3.1: The *ALS* Functional Rating Scale

The ALS Functional Rating Scale Features
1. Speech
2. Salivation
3. Swallowing
4. Handwriting
5. Cutting food and handling utensils
6. Dressing and hygiene
7. Turning in bed and adjusting bed clothes
8. Walking
9. Climbing stairs
...

features. Thus, in order to further demonstrate our feature selection methods, we conduct studies using the following datasets:

- The *ALS* clinical data from the $PRO - ACT$ (Pooled Resource Open-access *ALS* Clinical Trials) [1] dataset, which consists of more than 10,700 patients' records with 6318 features including demographics, discriminated $ALSFRS$ features [43], medical and family history, respiratory measurements and other general lab data. For the *ALS* dataset, we extract features for progression and survival rate predictions.

- The *RHC* dataset [2] contains data of 5735 patients with 62 attributes, which includes not only the patients' characteristics (e.g., age, sex, education, income, medical insurance) but also various lab tests results that describe the severity of patients' conditions. For the *RHC* dataset, we extract features for survival rate prediction.

- The STAR*D dataset [122]: STAR*D is a study involving over 4000 patients to identify the most effective treatment or combination of treatments for patients diagnosed with non-psychotic Major Depression Disorder (MDD) which lasted over a period of seven years. For the STAR*D dataset, we extract features for predicting disease relapse.

Disease Progression Prediction

For *ALS*, the cleaning dataset can be divided into two types:

Table 3.2: Feature Selection for Progression Prediction

Scheme	Prediction	Feature Selection	Names of Features
UglyDuckling	ALS Slope		Onset-delta, Trunk, Q1-Speech, Phosphorus, Q5-Cutting, Leg
Our	ALS Slope	Random Forest	Q1-Speech, Q3-Swallowing, Weight, ALSFRS-Total

- Static features: contain values of patients' profiles, e.g., time of onset, first symptoms, gender, etc.
- Temporal features: contain functional ($ALSFRS$) measures [43], body weight, lab test results, etc, where their values are different as time varies.

However, since no one knows exactly which features (patients characteristics) are more important for ALS disease progression, we conduct feature selection experiments to assess which features are more helpful in constructing ALS clinical prediction models.

Multiple feature selection algorithms have been designed using the ALS dataset based on information gains, random forest, etc during the Prize4Life challenge. For example, in order to predict ALS slopes, the UglyDuckling team [5] selects features based on the information entropy, where information gain is computed for all variables. The GuanLab team [6] also selects features based on $ALSFRS$ measures and Forced Vital Capacity (FVC) surveys, where FVC is the volume of gas that can forcibly be blown out after full inspiration, measured in liters.

In our work, we choose to focus on patients' ALS functional rating scale ($ALSFRS$) features (Table 3.1) since they are essential elements of the ALS clinical trials. The $ALSFRS$ features reflect physical functions in carrying out activities of daily living of ALS patients (i.e., how well patients speak, swallow, etc). Based on the prediction results obtained using the random forest classification method, we have selected three features including "Q1-Speech", "Q3-Swallowing" and "ALSFRS-Total" from $ALSFRS$ features with highest Gini index scores, where "Q1-Speech" and "Q3-Swallowing" are the evaluations of the functional change (e.g., speech, swallowing) of patients over time.

While $ALSFRS$ plays an important role in the diagnosis of ALS, other factors should also be considered so as to improve the predication accuracy. Thus, we also include the

Table 3.3: Feature Selection for Survival Rate Prediction

Scheme	Prediction	Feature Selection	Names of Features
GuanLab	ALS Survival		Age, ALSFRS-Total, FVC1, FVC-percent, Q3-Swallowing, Weight
Our	ALS Survival	Random Forest	FVC, FVC-percent, FVC-percent1, Age, ALSFRS-Total, Onset-delta
Our	ALS Survival	Logistic Regression	FVC, Q8-walking, Q10-respiratory, Age, ALSFRS-Total, Creatinine
Our	RHC Survival	Random Forest	Cat1, Death, Swang1, Gender, Race, Hrt1, Card, Ca, Age, Meanbp1
Our	RHC Survival	Logistic Regression	Cat1, Death, Swang1, Gender, Race, Ninsclas, income, Ca, Age, Meanbp1
Our	Star*D Survival	Random Forest	gender, mdswch1, hwl, ctswch1, menop, mdsch2ct, mdswch2a, mdaug2a
Our	Star*D Survival	Logistic Regression	gender, mdswch1, mdaug1, ctswch1, ctaug1, mdsch2ct, mdswch2, mdaug2

"weight" feature since people affected by *ALS* tend to lose weight. This may be caused by several factors: (i) they often have difficulty swallowing, (ii) they burn more calories than unaffected people, and (iii) cells in their intestines may have difficulties extracting nutrients from the food. The selected features are shown in Table 3.2, where for the "ALSFRS-Total", "Q1-Speech", "Q3-Swallowing", we have used both their minimum and maximum values as additional features. Note that logistic regression method is not used for the *ALS* slope prediction since such dataset contains many features (more than 100 features) and hence it is inefficient to extract useful features using this method.

Survival Rate Prediction

(1) For *ALS* Survival Rate Prediction

- We use random forest classifier to select six features (with highest Gini index values), which include "FVC", "FVC-percent", "FVC-percent1", "Age", "ALSFRS-Total" and "Onset-delta" to generate learning model for *ALS* survival rate prediction, where "Onset-delta" is the time between disease onset and the first time the patient was tested in a trial, "FVC-Percent" is the percentage of normal lung function (exhalation is gentle and not forced), and "FVC-percent1" is the percentage of the volume

of air forcefully exhaled in one second.

- We also use logistic regression analysis to select useful features, which include "ALSFRS-Total", "FVC", "Age", "Q8-Walking", "Q10-Respiratory" and "Creatinine", where "Creatinine" in the blood reflects both the amount of muscle a person has and their amount of kidney function, "Q8-Walking" and "Q10-respiratory" are the evaluations of the functional change (e.g., walking, breathing) of patients over time.

The selected features are shown in Table 3.3, where for the "ALSFRS-Total", "FVC", "FVC-Percent", "FVC-Percent1", "Q8-Walking" and "Q10-Respiratory" features, we have used both their minimum and maximum values as additional features.

(2) For *RHC* Survival Rate Prediction

- We select 10 features based on the results using the random forest classification method, where Gini index scores are computed for all variables. The selected features include age, gender, cat1 (primary disease category), ca (none cancer, localized cancer, metastatic cancer), meanbp1 (mean blood pressure), hrt1 (heart rate), swang1 (right heart catheterization performed within first 24 hours), death (estimation of the probability of surviving 180 days after admission), race (black, white, other) and card (cardiovascular diagnosis).

- We also use logistic regression analysis to select important features including age, sex, race, years of education, income, swang1, ninsclas (type of medical insurance including private, medicare, medicaid, private and medicare, medicare and medicaid, or none), cat1, ca, death and meanbp1.

3.3.3 Model Construction

After data cleaning and feature selection, we then generate our learning models for disease progression and survival rate predictions.

Disease Progression Prediction

For *ALS* progression prediction, we apply Gradient Boosted Regression Trees (*GBRT*) [5], which performs best in the DREAM *ALS* Stratification Prize4Life Challenge [4], to generate our learning model. The idea of *GBRT* is to add a classifier at a time, so that the next classifier is trained to improve the already trained ensemble. In other words, *GBRT* computes a sequence of simple decision trees, where each successive tree is built based on the prediction residuals of the preceding tree. The input of the model is the patient's data with selected features, and the output is the predicted *ALS* slope.

In addition, to evaluate the accuracy of a trained model for predicting *ALS* slope, we also cluster patients into different groups based on their observed *ALS* slopes. For example, based on the observed *ALS* slopes, the patients could be clustered into three classes, denoted as:

- For "fast" labeled patients, their *ALS* slopes are smaller than "-1.1".
- For "slow" labeled patients, their *ALS* slopes are larger than "-0.5".
- The rest patients will be labeled as "average".

Survival Rate Prediction

Since the survival duration ranges from several months to over a decade from the onset of symptoms, we also generate learning models for *ALS* and *RHC* survival rate predictions by clustering patients into either the "dead" or "alive" class, where "alive" means that the patients are still alive after 12 months of being first diagnosed. In our work, we have used three methods to build the prediction models:

(1) Cox-survival Model

The Cox model [56] is a well-recognized statistical technique for exploring the relationship between the survival of a patient and several predictor variables. The probability of a patient's survival rate is called the hazard, which can be denoted as:

$$H_i(t) = H_0(t) \times \quad exp(b_1 X_{i1} + b_2 X_{i2} + \cdots + b_k X_{ik}) \tag{3.1}$$

where $\{X_{i1}, ..., X_{ik}\}$ is a collection of predictor variables (e.g., k lab test results) for patient i and coefficients $\{b_1, ..., b_k\}$ are estimated by Cox regression. Thus, the survival model of one patient can be viewed as consisting of two parts: the baseline hazard function ($H_0(t)$), which can be computed based on all patients' clinical data, and the patient's own predictor variables.

(2) MTLR Model

While the Cox proportion hazards model [56] has long been used to fit the survival time of a population, it might not give good predictions on survival times for individuals since it has ignored many important individual differences among patients. To overcome this problem, we have used a method called multi-task logistic regression ($MTLR$) [168], which directly models the survival function by using a likelihood function that combines multiple logistic regression models, where each logistic regression model measures the effect of how the characteristics X of a specific patient affect the chance of survival with the threshold k at time slice t.

$$P_\theta(T \geq t|(X_{i1}, X_{i2}, \ldots, X_{im})) = 1/(1 + exp(\theta_1 X_{i1} + \theta_2 X_{i2} + \cdots + \theta_m X_{im} + k)) \quad (3.2)$$

$\{X_{i1}, ..., X_{im}\}$ is a collection of predictor variables (e.g., m important variables) for patient i and regression coefficients $\{\theta_1, ..., \theta_m\}$ are estimated by maximizing a partial likelihood objective, which depends on the relative ordering of survival time of individuals. Thus, $MTLR$ provides more intuitive hazard functions rather than force us to choose a particular hazard function as in the Cox-survival model.

(3) GuanLab Model

Guanlab [6] invents a probabilistic ranking of all training patients based on Kaplan-Meier (KM) curve [87]. This curve gives an estimation of the survival function across time, $r(t)$, which is the proportion of the patients that are still alive at time t. After completing the comparison of all data points, a ranking of the likelihood to die for all patients is calculated, which can be used to build the learning model.

3.4 Evaluation

In this subsection, we evaluate our data cleaning and feature selection mechanisms via measuring the performance of disease progression and survival rate predictions using three datasets, namely, the ALS, RHC and STAR*D datasets. We also evaluate our learning models by comparingthem with two other models produced by UglyDuckling [5] and GuanLab [6], which perform best in predicting ALS slopes and survival probabilities respectively.

3.4.1 Performance Metrics

To assess the prediction performance, the sets of computed slopes A and predicted slopes B are compared across patients using root mean square deviation ($RMSD$) [7] and pearson's correlation coefficient (PCC) [134]. The $RMSD$ measures the differences between corresponding slope pair values predicted by a model and the values actually observed, which can be denoted as:

$$RMSD = \sqrt{(1/n)\sum_{i=1}^{n} |a_i - b_i|^2} \tag{3.3}$$

where n is the total number of instances, a_i is the actual $ALSFRS$ slope and b_i is the predicted $ALSFRS$ slope.

In addition, we also use pearson's correlation coefficient (PCC) to evaluate how well a prediction model is able to reveal $ALSFRS$ trends. PCC can be computed as follows:

$$PCC = cov(A, B)/(\sigma_A \sigma_B) \tag{3.4}$$

where $cov(A, B)$ is the covariance of A and B and σ_A, σ_B are the standard deviation of A and B respectively. Thus, better predictions lead to a lower value of $RMSD$ but a higher value of PCC.

3.4.2 Experimental Settings

To evaluate the performance of our schemes, we conduct the experimental evaluations on three datasets:

Table 3.4a: Experimental Settings for *ALS* Progression Prediction of Exp1

Scheme	Training Data	Fast	Slow	Average	Selected Features
UglyDucking	4835	473	957	3405	Onset-delta, Trunk, Phosphorus, Q1-Speech, Leg, Q5-Cutting
Our (RF)	4838	475	958	3405	Q1-Speech, Q3-Swallowing, Weight, ALSFRS-Total

- The $PRO-ACT$ [1] dataset contains clinical records from over 10,700 *ALS* patients who have participated in 23 phase II/III clinical trials.
- The *RHC* dataset [2] contains 5735 patients admitted to hospitals in serious condition with 62 attributes.
- STAR*D dataset [8] enrolls 4041 patients aged 18 to 75 years with 41 clinical sites (e.g., 18 prirnary and 23 psychiatric care settings).

All our experiments are conducted on a Mac Pro with an Intel Core i5 processor running at 2.6GHz and 8GB memory.

3.4.3 Impact of Different Selected Features

In this part, we have conducted two experiments (Exp1 & Exp2) to measure the performance of our learning model with features selected using different methods. For both experiments, we replace missing values in the datasets with the average values obtained from non-missing entry values. 80% of the cleaned dataset is used for training and the rest is used for testing.

(1) Disease Progression Prediction

Exp1: For *ALS* slope prediction, the UglyDuckling team [5] (the winning team) in the DREAM *ALS* Stratification Prize4Life Challenge [4] used the criteria that they will only include patient records where each patient's record has values for over 66.7% of their selected features. In this experiment, we apply the same criteria and find 4838 patients' records based on our selected features.

From the results (Table 3.4b), we can see that our method performs much better than UglyDuckling (improved by 18%) with fewer selected features.

(2) Survival Rate Prediction

Table 3.4b: *ALS* Progression Prediction Results of Exp1

Scheme	Train Size	Feature Size	Train Cost	Test Cost	*RMSD*	*PCC*	Accuracy
GBRT (*UglyDuckling*)	4835	10	1.461s	0.005s	0.549	0.367	40.5%
GBRT(*Our*(*RF*))	4838	7	1.445s	0.004s	0.521	0.384	58.3%

Table 3.5a: Experimental Settings for ALS Survival Rate Prediction of Exp2

Scheme Prediction	Training	Dead Data	Alive	Selected Features
GuanLab	7308	2603	4705	Age, ALSFRS-Total, FVC1, FVC-percent, Q3-Swallowing, Weight
Our (RF)	7413	2645	4768	FVC, FVC-percent, FVC-percent1, Age, ALSFRS-Total, Onset-delta
Our (LR)	7538	2743	4795	FVC, Q8-walking, Q10-respiratory, Age, ALSFRS-Total, Creatinine

Exp2: In this experiment, we use three datasets to verify our feature selection methods, which is a crucial step for ensuring the quality of predictive models.

- For *ALS* survival rate prediction, the GuanLab team [6] used the criteria that they would include any patient's record that has values for over 16.7% of their selected features. Using the same criteria, (i) we extract 7413 patients' data as the training data based on the results using the random forest classification method (Gini index); (ii) we also extract 7538 patients' data as the training data based on the results computed by logistic regression algorithm. (Table 3.5a)

- For *RHC* survival rate prediction, we use both random forest (Gini index) and logistic regression methods to select important features (Table 3.6a).

- For STAR*D survival rate prediction, (i) we select 8 features based on the results using the random forest classification method, where Gini index scores are computed for all variables. The selected features includes gender, mdswch1 (leaving lev1 med switch), hwl (HRS weight loss), ctswch1 (leaving lev1 switch to ct), menop (menopausal), mdsch2ct (leaving lev2 with ct med switch), mdswch2a (leaving lev2a med switch) and mdaug2a (leaving lev2a med augment); (ii) we also use logistic regression analysis to select useful features [122], which includes gender, mdswch1, mdaug1 (leaving lev1

Table 3.5b: *ALS* Survival Rate Prediction Results of Exp2

Scheme	Feature Selection	Train Size	Test Size	Features Size	Train Cost	Test Cost	Accuracy
GuanLab(*guan*)		7308	1260	10	3.718s	0.025s	72.3%
Cox(*guan*)		7308	1260	10	3.524s	0.032s	70.6%
MTLR(*guan*)		7308	1260	10	5.113s	0.046s	71.4%
GuanLab(*our*)	RF	7413	1260	6	2.984s	0.019s	72.9%
GuanLab(*our*)	LR	7538	1260	6	3.043s	0.020s	63.8%
Cox(*our*)	RF	7413	1260	6	2.841s	0.023s	70.9%
Cox(*our*)	LR	7538	1260	6	2.889s	0.024s	60.4%
MTLR(*our*)	RF	7413	1260	6	4.672s	0.043s	72.1%
MTLR(*our*)	LR	7538	1260	6	4.752s	0.042s	63.2%

Table 3.6a: Experimental Settings for RHC Survival Rate Prediction of Exp2

Scheme Prediction	Training	Dead Data	Alive	Selected Features
Our (RF)	4588	2854	1734	Cat1, Death, Swang1, Gender, Race, Hrt1, Card, Ca, Age, Meanbp1
Our (LR)	4588	2854	1734	Cat1, Death, Swang1, Gender, Race, Ninsclas, Income, Ca, Age, Meanbp1

med augment), ctswch1, ctaug1 (leaving lev1 augment with ct), mdsch2ct, mdswch2 (leaving lev2 no ct med switch) and mdaug2 (leaving lev2 no ct med augment). (Table 3.7a)

The experimental results are shown in Table 3.5b, Table 3.6b and Table 3.7b. From the results, we can see that the *MTLR* model can produce more accurate results for survival rate predictions than the *Cox* model. It is reasonable, since the *MTLR* mechanism can

Table 3.6b: *RHC* Survival Rate Prediction Results of Exp2

Scheme	Feature Selection	Train Size	Test Size	Features Size	Train Cost	Test Cost	Accuracy
GuanLab		4588	1147	62	13.923s	0.142s	71.6%
GuanLab	Random Forest	4588	1147	10	2.583s	0.029s	66.3%
GuanLab	Logistic Regression	4588	1147	10	2.528s	0.028s	69.2%
Cox		4588	1147	62	12.859s	0.163s	68.4%
Cox	Random Forest	4588	1147	10	2.426s	0.031s	62.3%
Cox	Logistic Regression	4588	1147	10	2.396s	0.031s	65.6%
MTLR		4588	1147	62	16.267s	0.186s	69.7%
MTLR	Random Forest	4588	1147	10	4.257s	0.041s	65.4%
MTLR	Logistic Regression	4588	1147	10	4.132s	0.042s	67.8%

Table 3.7a: Experimental Settings for Star*D Survival Rate Prediction of Exp2

Survival Rate Prediction	Scheme	Training Data	Selected Features
STAR*D	Our (RF)	3151	gender, mdswch1, hwl, ctswch1, menop, mdsch2ct, mdswch2a, mdaug2a
STAR*D	Our (LR)	3200	gender, mdswch1, mdaug1, ctswch1, ctaug1, mdsch2ct, mdswch2, mdaug2

capture the time-varying effects of features by modeling the survival distribution as the joint output of a sequence of dependent regressors. We also find that GuanLab method performs better than the Cox and $MTLR$ mechanisms (improved by 3% in average) . We suspect that this is due to the fact that the GuanLab [6] team builds their model based on a complete ranking of all patients (e.g., a ranking of likelihood to die for all patients) using KM-curves [87]. With this probabilistic ranking, different features would have different weights when generating the prediction model.

From Table 3.6b we can see that for RHC survival rate prediction while the results of using the whole 62 features is more accurate than those with 10 features, the training and testing costs are significantly reduced when only using 10 selected features. In addition, we also find that compared with the results obtained using the logistic regression algorithm,

- For ALS dataset, our learning models perform better (improved by 10% in average) when the features are selected based on the results of the random forest method.
- For RHC and Star*D datasets, they perform worse (reduced by 3% and 4% in average) when we select features based on the results of the random forest method.

Therefore, the feature selection method should be chosen as follows: (i) If the dataset (e.g., RHC, Star*D) has limited number of features (e.g., fewer than 100), then the logistic regression method performs well in selecting useful features; (ii) If the dataset (e.g., ALS) has large number of features, then the random forest classification method (Gini index) works well for feature selection.

Table 3.7b: Star*D Survival Rate Prediction Results of Exp2

Scheme	Feature Selection	Train Size	Test Size	Features Size	Train Cost	Test Cost	Accuracy
GuanLab	Random Forest	3151	800	8	2.178s	0.023s	76.8%
GuanLab	Logistic Regression	3200	800	8	2.067s	0.021s	79.9%
Cox	Random Forest	3151	800	8	1.892s	0.022s	68.5%
Cox	Logistic Regression	3200	800	8	1.986s	0.027s	72.6%
MTLR	Random Forest	3151	800	8	3.447s	0.034s	75.6%
MTLR	Logistic Regression	3200	800	8	3.452s	0.036s	79.4%

Table 3.8a: Experimental Settings for *ALS* Slope Prediction of Exp3 & Exp4

Scheme	Feature Selection	Train Size	Fast	Slow	Average
GBRT(Exp3)	Random Forest	2235	491	964	780
GBRT(Exp4)	Random Forest	2132	324+400	725	683

3.4.4 Impact of Data Quality

In this part, we conduct four experiments (Exp3 & Exp4 & Exp5 & Exp6) to evaluate how data quality affects the performance of our proposed mechanisms, where {Exp3, Exp4} and {Exp5, Exp6} are conducted for disease progression and survival rate predictions respectively. For Exp3 and Exp5, we only use patients' data that do not have any missing values of the selected features. However, since the number of non-missing data is very limited, so in Exp4 and Exp6 we also generate some synthetic data to build a more balanced training set, where each class contains approximately the same number of data records.

(1) Disease Progression Prediction (Table 3.8a)

Exp3: In this experiment, we have extracted 2235 *ALS* patients' data with non-missing values for training.

Exp4: We first use 10-fold cross validation on the 2235 *ALS* patients' data with no missing values. Then, we combine all those testing instances from these 10-fold cross validation experiments which are correctly classified to produce a dataset of 1732 patients' data (with

Table 3.8b: *ALS* Slope Prediction Results of Exp3 & Exp4

Scheme	Train Size	Test Size	Feature Size	Train Cost	Test Cost	*RMSD*	*PCC*	Accuracy
GBRT(Exp3)	2235	260	7	0.916s	0.003s	0.509	0.394	67.3%
GBRT(Exp4)	2132	260	7	0.908s	0.003s	0.504	0.401	73.5%

Table 3.9a: Experimental Settings for *ALS* Survival Rate Prediction of Exp5 & Exp6

Scheme	Feature Selection	Train Size	Dead	Alive
GuanLab(Exp5)	Random Forest	4432	1597	2835
GuanLab(Exp6)	Random Forest	4328	1152+1000	2176
MTLR(Exp5)	Random Forest	4432	1597	2835
MTLR(Exp6)	Random Forest	4328	1152+1000	2176

324 "fast", 725 "slow", 683 "average"). Next, we generate 400 synthetic patients' data which belongs to the "fast" class by randomly generating a value for each selected feature using a uniform distribution with the mean and 1 standard deviation of that particular feature computed from the 324 "fast" instances. Finally, we form a balanced dataset with 2132 patients data to train a new prediction model.

The results are shown in Table 3.8b. From the results we can see that the accuracy of our models in Exp3 is much higher than those in Exp1. It is because although disease progression can be roughly predicted based on the $ALSFRS$ scores, the accuracy of the predictions varies significantly from patient to patient. Thus, it is not as effective to simply replace any missing data with the mean value since $ALSFRS$ could be highly affected by the patient's family history, gender, age, etc. In addition, we also find that compared with Exp3, the accuracy of our methods has increased by 6% in Exp4 which uses a more balanced dataset where some data records are generated synthetically based on carefully selected real data which is classified correctly.

(2) Survival Rate Prediction (Table 3.9a)

Exp5: In this experiment, we have extracted 4432 *ALS* patients' data with non-missing values for training.

Exp6: We use the same process (as described in Exp4) to produce 3328 *ALS* patients' data (with 1152 "dead", 2176 "alive") from the 4432 patients' data with no missing values. Then, we generate 1000 synthetic "dead" patients' data using the same method and produce a balanced dataset with 4328 patients data to train a new prediction model.

The results are shown in Table 3.9b. From the results we can see that compare with Exp2, the performance has been improved when we use non-missing data. Meanwhile, from the results we also find that compared with Exp5, the accuracy of our methods in Exp6 has increased by 2% on the average as a result of having higher quality and more balanced

Table 3.9b: *ALS* Survival Rate Prediction Results of Exp5 & Exp6

Scheme	Train Size	Test Size	Feature Size	Train Cost	Test Cost	Accuracy
GuanLab(Exp5)	4432	1260	6	2.574s	0.023s	73.8%
MTLR(Exp5)	4432	1260	6	3.931s	0.037s	73.2%
GuanLab(Exp6)	4328	1260	6	2.353s	0.025s	74.6%
MTLR(Exp6)	4328	1260	6	3.784s	0.034s	74.4%

data.

In addition, we also run similar experiments for the *RHC* dataset (see our technique report), and the result trends are similar as the ones obtained using the *ALS* dataset. It shows that higher quality data yields better prediction models. Therefore, to obtain more high-quality data with fewer missing values, one needs to design an incentive scheme to encourage different hospitals to collect higher quality data. Thus, in the next section, we present an incentive scheme which we design for this purpose.

3.5 Proposed Incentive Scheme

In this subsection, we first give the definitions of various notations we have used in our work. Then, we describe an overall framework of our incentive based high-quality ehealth information sharing scheme (*IHESS*) before we dwelve into the detailed explanations.

Notations:

- Sensitivity (*SN*) - $SN = TP/(TP + FN)$, where TP is all positive instances that are classified as positive and FN is all positive instances that are not classified as positive.
- Specificity (*SP*) - $SP = TN/(TN + FP)$, where TN is all non-positive instances that are not classified as positive and FP is all non-positive instances that are classified as positive.
- S - $S = SN * SP$ ranges from 0 to 1.
- S_{min} - The minimum value of S, denoted as $min(S)$.
- R - A predefined reward factor for every selected "average" instance.

- C - The cost for every submitted "average" instance.
- α - A predefined target for the achieved utility of a cloud server.
- K - A scaling factor used to make sure that the cloud server is profitable when it includes new data.
- w_f, w_s - Scaled reward factors for "fast" and "slow" instances.
- k_f, k_s - Scaled frequency factors for "fast" and "slow" instances.
- N - The maximum number of instances in any "fast/slow/average" class in the aggregated training set.
- $N_f(H_i)^*$, $N_s(H_i)^*$, $N_a(H_i)^*$ - The accepted "fast", "slow" and "average" instances of the submitted instances of hospital H_i.
- $|N_f|$, $|N_s|$, $|N_a|$ - The total number of "fast", "slow" and "average" instances that have been selected to generate the training model, where $max(|N_f|, |N_s|, |N_a|) = N$.
- $T_f(H_i)^*$, $T_s(H_i)^*$, $T_a(H_i)^*$ - The submitted "fast", "slow" and "average" instances of hospital H_i, where $max(|T_f(H_i)^*|, |T_s(H_i)^*|, |T_a(H_i)^*|) = N$.

3.5.1 Overall Framework

The main goal of our framework is to provide a practical incentive mechanism which encourages truthful data sharing. Fig 3.3 is an overview of our designed scheme which shows the information provided by the hospitals, and the cloud server.

Every hospital H_i first sends its collected data $T(H_i)^*$=$\{T_f(H_i)^*$,$T_s(H_i)^*$,$T_a(H_i)^*\}$ to the public cloud. Upon receiving data from hospital H_i, the public cloud will first check if those data can be accepted. If it is, then the public cloud will compute the related reward and send it back to H_i.

Our $IHESS$ is designed with the following desirable properties:

- Individual Rationality: The hospital H_i whose submitted data are accepted by the cloud server will have an utility greater than 0. That is, $U_{H_i} = Rewards(H_i) - Cost(H_i) > 0$.

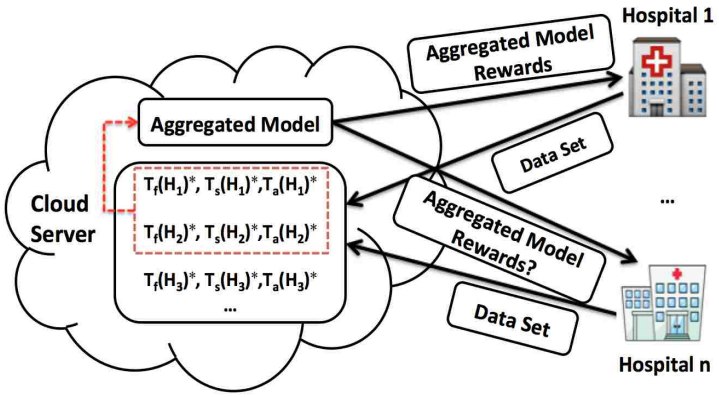


Figure 3.3: System Overview

- Platform Profitability: The utility of the $IHESS$ platform is greater than 0 when it includes new data, which can be denoted as $U_P = Benefit(P) - Reward > 0$.

3.5.2 Detail Design of IHESS

Here, we present more detailed descriptions of the proposed $IHESS$.

Costs for Participating Hospitals:

During the data collection, every hospital H_i incurs its own cost, which is largely dependent on the number of patients, the number of lab tests conducted on each patient, the costs of the lab tests, etc. Thus, the cost for the hospital H_i can be calculated as follows:

$$Cost(H_i) = (|T_f(H_i)^*| * k_f + |T_s(H_i)^*| * k_s + |T_a(H_i)^*|) * C \tag{3.5}$$

where we have set $k_f > 1 > k_s$ since the "fast" ALS patients need to take lab tests more frequently than those "slow" and "average" patients.

Reward Function for Participating Hospitals:

The main challenge of designing an incentive mechanism for data sharing is that participating hospitals are more prone to sending noisy or unreliable data (i.e., labeled patients wrongly) to the cloud server if they have received with very limited rewards. To avoid

such behavior, the public cloud needs to design an appropriate reward function to ensure individual rationality.

Since the cost incurred by each hospital is a function of the number of patients so the reward function should be a function of the total number of accepted patients' records from every hospital H_i. Meanwhile, the reward function is also designed based on the performance (e.g., S) of aggregated models so that every hospital H_i could receive more reward if it sends more truthful data (i.e., label patients correctly and improve S) to the cloud server. In addition, since data gathered from the "fast" and "slow" ALS patients are more useful than those from the "average" patients, so the cloud server also sets different reward factors for patients from different classes to encourage higher quality data sharing. Thus, the reward function for a hospital H_i is chosen to be as follows:

$$Reward(H_i) = (|N_f(H_i)^*|w_f k_f + |N_s(H_i)^*|w_s k_s + |N_a(H_i)^*|)R * S \tag{3.6}$$

where the values of R, w_f, k_f, w_s and k_s are determined in advance. Here we have set $w_f > w_s > 1$ since the "fast" and "slow" ALS patients are more useful in generating the prediction models.

Finally, the utility of hospital H_i can be computed as:

$$U_{H_i} = Reward(H_i) - Cost(H_i) = (|N_f(H_i)^*|w_f k_f + |N_s(H_i)^*|w_s k_s + |N_a(H_i)^*|)R * S - (|T_f(H_i)^*| * k_f + |T_s(H_i)^*| * k_s + |T_a(H_i)^*|) * C \tag{3.7}$$

Benefit Function for the Cloud Server:

The cloud server determines its own benefit function to ensure that adding truthful data into the aggregated dataset will always bring benefit to the platform. Meanwhile, it is obvious that with more and more data accepted, the marginal benefit brought by the new data will become less and less, which leads to a decreasing marginal revenue. Thus, in order to reflect the platform's diminishing returns on increasing number of participating hospitals, the cloud server uses logarithmic term (e.g., log(1+t) where t is a ratio of current

S over a target α value) to implement its benefit function. We choose $t = S/\alpha$ to ensure that the decision made by the cloud server in rejecting unreliable data can help to drive S towards the target α value. Besides, to ensure platform profitability, the cloud server also includes a system related scaling factor, K. Therefore, the benefit function of the cloud server is chosen as follows:

$$Benefit(P) = K * log(1 + S/\alpha) \tag{3.8}$$

where K and α are predefined so as to ensure that the cloud server is profitable when it includes new data.

Based on the benefit and reward functions, the cloud server can then compute its utility as:

$$\begin{aligned} U_P = Benefit(P) - Reward = K * log(1 + S/\alpha) - \\ (|N_f| * R * w_f * k_f + |N_s| * R * w_s * k_s + |N_a| * R) * S \end{aligned} \tag{3.9}$$

Workflow of the IHESS:

Every hospital H_i first sends $T(H_i)^*$=$\{T_f(H_i)^*$,$T_s(H_i)^*$,$T_a(H_i)^*\}$ to the cloud server. After receiving the data from H_i, the cloud server will check whether its utility increases when including those data. If it is, then the cloud server will add those data into the aggregated dataset, generate the aggregated model and send reward to the hospital H_i. Otherwise, the cloud server will neglect those submitted data, and will not send any reward to the hospital H_i.

3.5.3 Properties of the Incentive Model

In this part, we prove that our incentive scheme ($IHESS$) has the properties of individual rationality and platform profitability.

Individual Rationality:

If data of the participating hospital H_i has been rejected, then the corresponding reward will be 0. Thus, we only need to consider the case when the data from a hospital is accepted.

The utility of the participating hospital H_i can be computed as follows:

$$
\begin{aligned}
&U_{H_i} = Reward(H_i) - Cost(H_i) = (|N_f(H_i)^*|w_f k_f + |N_s(H_i)^*|w_s k_s + |N_a(H_i)^*|)R * S \\
&-(|T_f(H_i)^*| * k_f + |T_s(H_i)^*| * k_s + |T_a(H_i)^*|) * C \\
&\because max(|T_f(H_i)^*|, |T_s(H_i)^*|, |T_f(H_i)^*|) \leqslant N, \\
&\therefore U_{H_i} \geqslant (|N_f(H_i)^*|w_f k_f + |N_s(H_i)^*|w_s k_s + |N_a(H_i)^*|)R * S - (k_f + k_s + 1)N * C \\
&\geqslant (w_f k_f + w_s k_s + 1)R * S - (k_f + k_s + 1)N * C \\
&\geqslant (w_f k_f + w_s k_s + 1)R * S_{min} - (k_f + k_s + 1)N * C \\
&if \quad R \geqslant \tfrac{(k_f + k_s + 1)N * C}{S_{min} * (k_f w_f + k_s w_s + 1)}, then \quad U_{H_i} \geqslant 0
\end{aligned}
\tag{3.10}
$$

From the equation, we can see that with an appropriate R value, our incentive mechanism can guarantee that every participating hospital H_i will have a non-negative utility if its data is accepted.

Platform Profitablility:

Based on the benefit and reward functions, the utility of the cloud server can be computed as:

$$
\begin{aligned}
&U_P = Benefit(P) - Reward = K * log(1 + \tfrac{S}{\alpha}) - (|N_f|w_f k_f + |N_s|w_s k_s + |N_a|)R * S \\
&\Rightarrow (U_P(S))' = \tfrac{K}{(1+\frac{S}{\alpha}) * ln2 * \alpha} - (|N_f|w_f k_f + |N_s|w_s k_s + |N_a|)R \\
&If (U_P(S))' \geqslant 0, then \tfrac{K}{(1+\frac{S}{\alpha}) * ln2 * \alpha} - (|N_f|w_f k_f + |N_s|w_s k_s + |N_a|)R \geqslant 0 \\
&\Rightarrow \tfrac{K}{(1+\frac{S}{\alpha}) * ln2 * \alpha} \geqslant (|N_f|w_f k_f + |N_s|w_s k_s + |N_a|)R \\
&\Rightarrow \tfrac{K}{(\alpha + S) * ln2} \geqslant (|N_f|w_f k_f + |N_s|w_s k_s + |N_a|)R \\
&\Rightarrow \tfrac{K}{(|N_f|w_f k_f + |N_s|w_s k_s + |N_a|)R * ln2} - \alpha \geqslant S \\
&\because \quad S \in [0, 1], \therefore \tfrac{K}{(|N_f|w_f k_f + |N_s|w_s k_s + |N_a|)R * ln2} - \alpha \geqslant 1 \\
&\Rightarrow K \geqslant (1 + \alpha) * (|N_f|w_f k_f + |N_s|w_s k_s + |N_a|)R * ln2
\end{aligned}
\tag{3.11}
$$

Thus for the cloud server, we can find that the utility of the cloud server will always increase when new reliable data is added if we set K as $(1+\alpha) * (w_f k_f + w_s k_s + 1)R * N * ln2$.

3.5.4 Incentive Model Evaluation

Datasets Generation:

In this part, we conduct Exp7 to evaluate the effectiveness of our incentive mechanism in encouraging hospitals to share high quality data which can help to create better aggregated training models for disease progression and survival rate predictions.

Exp7: In this experiment, we generate synthetic datasets which contain both good and bad patients' data. The way we generate these datasets is described as follows:

(1) Disease Progression Prediction

For *ALS* slope prediction, we first create 2132 synthetically generated instances (with 724 "fast", 725 "slow" and 683 "average" instances) using feature values selected from a uniform distribution with the mean plus one standard deviation of the feature values extracted from that 1732 patients' data we described in Exp4. Then, we add these 2132 instances to the balanced dataset with 2132 instances to form a dataset with 4264 high quality instances. Next, we also generate another 4264 synthetically generated instances of poorer quality data using the mean and 2 standard deviation of the feature values (with 1448 "fast", 1450 "slow", 1366 "average"). Subsequently, we divide these 8528 instances into unequal portions and assume each portion comes from a different participating hospital.

(2) Survival Rate Prediction

- For *ALS* survival rate prediction, we first select the balanced dataset in Exp6, which contains 4328 patients' data. Then, we generate 4328 poor quality synthetic instances (with 2152 "dead" and 2176 "alive") using the same method (in Exp7(A)). Next, we divide these 8656 instances into unequal portions and assign them to the participating hospitals.

- For *RHC* survival rate prediction, we first use 10-fold cross validation on the 4588 patients' data with no missing values. Then, we only select 3196 records (1642 "dead", 1554 "alive") that have been correctly classified. Next, we generate 3196 poor quality synthetic instances using the same method. Finally, we divide these 6392 instances into unequal portions and assign them to the participating hospitals.

Table 3.10a: Experimental Settings for $IHESS$

Participants	Data Ratio	Prediction	Data Size	Accuracy
H_1	100 % good	ALS Slope Prediction	2000	72.0%
		ALS Survival Rate Prediction	2000	73.4%
		RHC Survival Rate Prediction	1500	72.9%
H_2	75 % good	ALS Slope Prediction	1600	65.6%
		ALS Survival Rate Prediction	1600	68.7%
		RHC Survival Rate Prediction	1100	68.2%
H_3	50 % good	ALS Slope Prediction	1328	57.6%
		ALS Survival Rate Prediction	1456	63.3%
		RHC Survival Rate Prediction	1192	63.6%
H_4	75 % bad	ALS Slope Prediction	1600	66.9%
		ALS Survival Rate Prediction	1600	68.9%
		RHC Survival Rate Prediction	1100	67.5%
H_5	100 % bad	ALS Slope Prediction	2000	71.5%
		ALS Survival Rate Prediction	2000	73.7%
		RHC Survival Rate Prediction	1500	73.2%

After being assigned its data portion, every hospital then generates its own local model by selecting 90% of its assigned data as the training set, and the remaining 10% as the testing set, which is shown in Table 3.10a. We also assume that the cloud server receives data from the hospitals in the following arrival pattern: H_2, H_3, H_1, H_5 and H_4.

Experimental Results:

From the result (in Table 3.10b), we can find that the cloud server will first accept the data submitted by H_2 since this is the first piece of data received by the cloud server. The cloud server will also select H_1 because the utility of the platform increases if the cloud server includes the data from H_1 in the aggregated training set. However, the cloud server will not select hospitals (e.g., H_3, H_4, H_5) since their data cannot improve the utility of the platform. Thus, our $IHESS$ can encourage more truthful data sharing by sending rewards to those participating hospitals which share higher quality data.

3.6 Summary

In this chapter, we have proposed useful learning models for Amyotrophic Lateral Sclerosis (ALS), Right Heart Catheterization (RHC) and depression disorder relapse (STAR*D) predictions, which can be used to aid efficient clinical care. We provide two major contri-

Table 3.10b: Aggregated Model Generation

Prediction	Participants	Data Ratio	Data Size	Accuracy	Aggregated Data
ALS Slope Prediction	H_2	75 % good	1600	65.6 %	$T(H_2)^*$
	H_3	50 % good	1328	57.6 %	$T(H_2)^*$
	H_1	100 % good	2000	72 %	$T(H_1)^*$, $T(H_2)^*$
	H_5	100 % bad	2000	71.5 %	$T(H_1)^*$, $T(H_2)^*$
	H_4	75 % bad	1600	66.9 %	$T(H_1)^*$, $T(H_2)^*$
ALS Survival Rate Prediction	H_2	75 % good	1600	68.7 %	$T(H_2)^*$
	H_3	50 % good	1456	63.3 %	$T(H_2)^*$
	H_1	100 % good	2000	73.4 %	$T(H_1)^*$, $T(H_2)^*$
	H_5	100 % bad	2000	73.7 %	$T(H_1)^*$, $T(H_2)^*$
	H_4	75 % bad	1600	68.9 %	$T(H_1)^*$, $T(H_2)^*$
RHC Survival Rate Prediction	H_2	75 % good	1100	68.2 %	$T(H_2)^*$
	H_3	50 % good	1192	63.6 %	$T(H_2)^*$
	H_1	100 % good	1500	72.9 %	$T(H_1)^*$, $T(H_2)^*$
	H_5	100 % bad	1500	73.2 %	$T(H_1)^*$, $T(H_2)^*$
	H_4	75 % bad	1100	67.5 %	$T(H_1)^*$, $T(H_2)^*$

butions, namely (i) identify meaningful features from all features collected during current clinical trials for efficient and accurate predictions, and (ii) design an incentive model to encourage participants to share their more truthful and high quality medical data so that aggregated training models can yield high accuracy. The experimental results indicate that our proposed methods would achieve good performance on those real-world clinical datasets.

Chapter 4

Explainable Deep Learning Based Medical Diagnostic System (DL-MDS)

4.1 Background

The rapid development of computing technologies makes it easier to collect patients' medical data. For example, Electronic Health Record (EHR) systems contain various types of patients' information including their demographics, diagnosis codes, medications, and laboratory test results [59, 94], which offer a richer data to accelerate clinical research and predictive analysis[81, 154, 51].

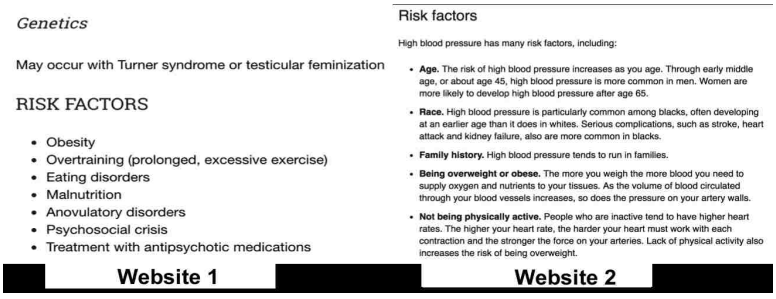


Figure 4.1: Information on Multiple Medical Web Sites

Although remarkable progress has been made to use medical datasets for clinical research, many challenges and open questions remain. One obstacle is that unlike other data

sources, medical data is inherently noisy, irregularly sampled and heterogeneous, which make it difficult to integrate useful data from multiple sources and produce predictive clinical models for real-world applications. For example, in Fig 4.1 different medical websites would provide different information (such as risk factors) for a same disease (e.g., "hypertension").

Figure 4.2: An illustrative example of a QA for community-based health services, where a patient has used a transition word "but" in his question

The other is that it is challenging for the community-based health services to understand latent relationships and efficient representations of medical concepts in the health-oriented questions, which are mainly caused by the following reasons: (i) The vocabulary gaps among diverse health patients make the data inconsistent sometimes. For example, patients may use different medical terms in their questions even when they have suffered from the same disease; (ii) Patients may also describe their problems in short questions with some negative or transitional words. Fig 4.2 illustrates one question answer (QA) example, where the existing community-based health services cannot understand patients' questions and hence provide wrong answers. According to our user study on 18,000 questions that we have crawled for the experiments, health seekers' questions can be abstracted into four main categories, as shown in Table 4.1.

Therefore, an important question naturally arises: how can one develop a comprehensive wellness system, which not only understands users' health-oriented questions but also extracts useful patterns from heterogeneous data sources to make more accurate disease predictions. In order to alleviate this problem, many researchers and institutions have utilized state-of-the-art machine learning and statistical techniques. Among those works, deep learning methods have increasingly been applied in healthcare informatics, where they have

Table 4.1: Categories of Health Seekers' Questions

Categories of Clinical Questions	Examples
(1) The clinical questions contain multiple keywords related to symptoms but not explicitly mention the exact disease name	My blood sugar tested high about 6 months ago. I am having problems with my lower legs and feet. Sometimes my feet tingle. What is wrong?
(2) The clinical questions mention the disease name but contextualized in interrogative scenarios	I've been dizzy, lightheaded, and unable to exhale fully for weeks and its gotten worse. Have you ever heard this from asthma?
(3) The clinical questions are short and contain some negative or transitional contexts	Went to the emergency room with asthma however the doctor said I had acid reflux.
(4) The clinical questions contain multiple disease keywords (related & unrelated), where the patients ask about one undiagnosed disease but who already have been diagnosed with another disease	I have been diagnosed as diabetic a few years ago. Recently, I have fatigue and exertional dyspnoea. Could I be suffered from lung disorder? Do I need treatment?

proved to have a strong representation ability to learn the complicated data structure and thus demonstrate good prediction performance on several tasks, such as medical concepts representation [51], predictive modeling [46, 114], etc. While existing deep learning methods are beneficial, the trained learning models or classifiers would perform badly on decision making when the training datasets are heterogeneous.

Thus, in this chapter, we investigate and propose an end-to-end deep learning based medical diagnosis system ($DL\text{-}MDS$) using multiple sources, which allows authorized users to conduct searches for disease diagnosis. In order to integrate the expert knowledge from multiple sources, we have designed a knowledge extraction framework, which can capture as many features as possible to characterize diseases. For example, online medical websites can be mined to extract reliable contents that can be included in the learning models to improve the overall performances of the system. Our $DL\text{-}MDS$ consists of three components: (i) medical diagnosis modules that can provide disease predictions; (ii) a topic model module that can capture informative keywords for different diseases; (iii) a query processing module that learns patients' queries via a $LSTM$ model with a convolutional neural network (CNN) based word embedding method. We evaluate our proposed methods using real-world data. Our experimental results show that our proposed system yields good performance on patients' queries processing and medical diagnosis.

In summary, our contributions are as follows:

- We design a medical knowledge extraction framework to collect useful data from multiple data sources.
- We propose an end-to-end deep learning based medical diagnosis system ($DL\text{-}MDS$) to allow authorized users to conduct searches for disease diagnosis based on their own personalized queries.
- We provide extensive experimental results using real-world data to show that our system is accurate and widely applicable.

4.2 Problem Formulation

A primary goal of precision medicine is to develop learning models, which can predict patients' health status or diagnose their illnesses. In this chapter, we have designed an end-to-end deep learning based medical diagnosis system ($DL\text{-}MDS$), which allows authorized users to conduct searches for disease diagnosis based on their personalized questions. In this section, we first outline some challenges and design goals. Then, we describe our system model.

4.2.1 Design Challenges

Different from other application domains (e.g., image and speech analysis), the problems in healthcare are more complicated. For example, the diseases are highly heterogeneous which make it hard for physicians to completely understand their causes and how they progress. In addition, the healthcare data are highly ambiguous and noisy, where the names of diseases may vary from different websites. For example, the disease "croup" can be named as "laryngotracheobronchitis" on other medical website. Thus, it is challenging to combine medical data extracted from different websites.

Furthermore, discovering efficient representations of medical concepts also has been a key challenge in a variety of applications. For example, existing systems have difficulty in dealing with patients' questions especially for the synonym scenarios where patients may use different terms "shortness of breath" or "breathless" in the questions to refer the same

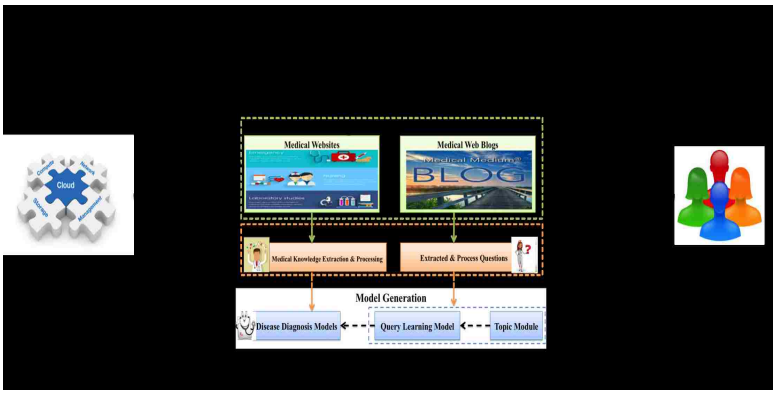


Figure 4.3: System Model

semantic "dyspnea".

4.2.2 Design Goals

To address the challenges above, we design a medical diagnosis system with the following goals in mind:

(1) Medical Knowledge Integration: Medical data extracted from multiple sources are ambiguous, noisy and not well-structured, so we design a medical knowledge integration framework to fuse such information for training models generation.

(2) A deep learning based medical diagnosis system ($DL\text{-}MDS$): it can be used to aid efficient clinical care, where authorized users can conduct searches for medical diagnosis. It consists of three parts: (i) disease diagnosis modules; (ii) a topic model module; and (iii) a query processing module.

4.2.3 System Model

We assume that there is a 3rd party service provider which deploys $DL\text{-}MDS$ to provide disease diagnosis service for the clients. The system model is depicted in Fig 4.3.

(1) Clients: Clients refer to those who wish to conduct searches for disease diagnosis based on their personal profiles.

(2) Service Provider: A 3rd party service provider can utilize a public cloud server to provide $DL\text{-}MDS$ that can predict patients' health status based on their submitted queries. The system works as follows:

(i) At the initial phase, we first extract useful information from heterogeneous data sources. Then, we propose a deep learning based medical diagnosis scheme ($DL\text{-}MDS$) to infer possible disease given the questions of patients, which consists of three components:

(a) Medical Diagnosis Modules: as for the medical diagnosis, we first collect disease information from professional medical websites. Then, a similarity matching algorithm will be applied to generate an aggregated dataset where the information of same diseases from multiple sources will be linked together. Next, we separate diseases into different clusters based on their ICD-10 codes and utilize machine learning techniques to generate a medical diagnosis module for every disease cluster.

(b) A Topic Model Module: in order to explore key information in the questions, we extract informative keywords to generate a topic module, which can (i) capture the relational connections among questions and diseases; (ii) provide model interpretation and improve the prediction performance.

(c) A Query Processing Module: we first extract medical questions from healthcare web blogs. Then, we apply a CNN-based word embedding method, at a sentence level to transform patients' queries into word-vectors. Next, we combine those word-vectors with the topic module and train a deep learning module based on the semantics extracted from those questions.

(ii) Once $DL\text{-}MDS$ is deployed, clients can send search requests based on their personal profiles (e.g., symptoms, risk factors). After receiving the search request, the system will first use the query processing module to analyze the question and find the related disease cluster. Then, it will launch a medical learning module stored in the corresponding cluster for diagnosis prediction and send answers back to the client.

4.3 Important Building Blocks

Before we present the detailed description of our newly designed system, we first discuss some of the techniques used in this work.

4.3.1 Word & Sentence Embedding

There have been many methods of deriving word embeddings in the NLP community. Recently, a word count-based model named GloVe [127] has been introduced, which outperforms existing models on several tasks including word similarity, named entity recognition, etc. While these models are beneficial, they neglect word order information. Thus, in order to tackle such sequence learning problem, Le and Mikolov [96] introduced a paragraph vector, where a sentence or a paragraph is represented by a vector and trained to predict the words in the document. In addition, in [89] Kim et al. proposed a neural network architecture, which uses the linguistic information in the Convolutional Neural Network (CNN). Instead of putting word in its sequential context, this model considered word and its n-gram sequences, which can preserve the long distance information in the sentence.

4.3.2 Deep Learning

Convolutional Neural Network (CNN) [98] is a type of feed-forward artificial neural network, which is modeled after the brain structure. Recently, with the availability of efficient GPU computing, CNN models have been shown to be effective for NLP tasks, and have achieved great success for various tasks such as information retrieval [135], sentence modeling [84], etc. Furthermore, a special kind of feed-forward neural network named Recurrent Neural Networks ($RNNs$), which can capture the underlying structure in sequential data, have been applied to many areas such as text classification [93], natural language processing (NLP) [159], etc. However, traditional $RNNs$ suffer from vanishing and exploding gradient problems. To handle these limitations, different variants of RNN have been proposed. For example, Long Short-term Memory ($LSTM$) [76] is a particular type of Recurrent Neural Network (RNN) [112] that works slightly better in learning long-term dependencies, owing to its more powerful update equation and some appealing backpropagation dynamics.

4.4 DL-MDS Scheme

In this section, we will present more detailed descriptions of our $DL\text{-}MDS$ (depicted in Fig 4.4), which provides a way for clients to diagnose their diseases based on their own

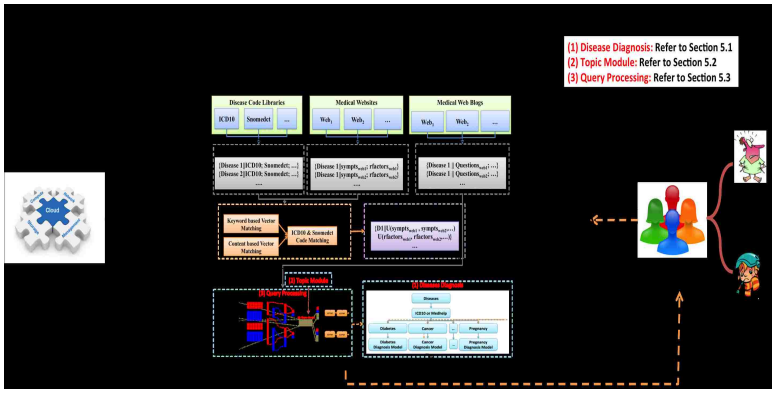


Figure 4.4: *DL-MDS* Overview

personalized queries.

4.4.1 Disease Diagnosis Modules Generation

In our work, we first design a medical knowledge extraction framework to collect reliable contents from multiple online medical websites. Then, we include the mined contents to generate the disease diagnosis model.

1. For each disease, we first extract its related symptoms and risk factors from multiple websites.

2. Then, we integrate the extracted data by matching their related medical terms. While it is convenient to use ICD-10 and SNOMED-CT codes to conduct the matching process, the performance is not good since every professionally named disease may have multiple ICD-10 and SNOMED-CT codes.

Thus, in order to link extracted disease terms, we not only use their ICD-10 and SNOMED-CT codes, but also consider their semantic similarities from keyword-based and content-based knowledge. In our work, we design an algorithm to measure similarities among extracted medical terms as follows:

(1) For each disease term T_i from a web page in a medical website, we first convert it into a word vector WV_i using pre-trained work embedding models (e.g., Word2Vec) and measure the semantic similarity of this vector WV_i with a target vector DV_k, which is generated based on D_k (one of the diseases in the disease training set) and set the similarity score as S_{1i} (Eq(4.8)), where $(WV_i \cdot DV_k)$ is the dot product of two vectors and $||WV_i||_2$ is

Figure 4.5: A Lung Cancer Related Question

the length of a vector.

$$S_{1i} = \frac{WV_i \cdot DV_k}{||WV_i||_2 ||DV_k||_2} \tag{4.1}$$

(2) We also extract the first M sentences from Wikipedia for this term T_i and convert each sentence (e.g., j^{th}) into its corresponding vector, denoted as SV_{ij}. We do the same for the term D_k Next, we measure the similarity of sentence vectors $\{SV_{i1}, ..., SV_{ij}, ...\}$, and set the average similarity score as S_{2i}.

(3) Next, we compute the similarity score between this term T_i and D_k as follows:

$$Similarity_{T_i} = \alpha S_{1i} + (1 - \alpha) S_{2i} \tag{4.2}$$

Two words that are similar may be two different illnesses, e.g., "varicella" and "varicocele" but two different words may represent the same illness, e.g., "bedsore" and "decubitus". We choose α such that content based similarity is more important than the keyword based similarity. Thus, we want $\alpha S_{1_{min}} + (1-\alpha) S_{2_{max}} > \alpha S_{1_{max}} + (1-\alpha) S_{2_{min}}$, where $S_{1_{min}}$, $S_{1_{max}}$, $S_{2_{min}}$ and $S_{2_{max}}$ are the minimum and maximum values of S_{1i} and S_{2i} respectively.

(4) Finally, for each D_k, we select top 5 medical terms $\{T_{k1}, ..., T_{k5}\}$ (from each medical website) with highest similarity scores and check if they have the same ICD-10 or SNOMED-CT code as D_k. For many cases, one of the 5 terms will have matching ICD-10 or SNOMED-CT code as D_k. Then, we can merge the symptoms and risk factors from the web page associated with that matched term, say T_{k*} to the D_k entry.

After using the similarity matching algorithm above, we can generate an aggregated dataset by integrating medical knowledge from different websites. In the dataset, every

disease D_i has M_1 symptoms and M_2 risk factors, where each symptom will be labeled with different levels (i.e. the higher level means such symptom is more unique for the corresponding disease), denoted as $\{D_i||(sympt1, level1), (sympt2, level2), ...||rfactor1, ...\}$.

3. Next, we separate the diseases in the integrated dataset into different clusters by mapping their top1 ICD-10 code (e.g., 1^{st} letter) to a disease categorization structure. For example, diseases with ICD-10 codes starting from "J" will be grouped into "respiratory" disease cluster. Finally, we use a logistic regression method to generate a disease diagnosis module for each cluster.

4.4.2 Topic Model Module Generation

Problem Statement

Patients may ask different questions even if they suffer from the same disease. Human beings can easily understand their questions by capturing important sentences with informative keywords within their questions. For example, for the question illustrated in Fig 4.5, readers understand that it is a lung cancer related question due to some informative keywords such as "ct-scan", "'benign" and "lung cancer" in the 1^{st} and 2^{nd} sentences.

Thus, we want to design a model that can conduct similar performance of human beings (interpret the contributing factors such as important sentences and keywords in the input data). Specifically, given the input sentences in a user's question, our pre-trained model within our topic model module can identify important sentences with their associate score vectors and also identify informative keywords. In this subsection, we demonstrate how to extract important sentences and informative keywords in patients' questions via a LSTM model with a convolutional neural network (CNN) based word embedding method (shown in Fig 4.6).

Informative Representations Extraction via CNN based Word Embedding

(1) Sentence Matrix Layer (Layer1)

The inputs to the network are raw sentences that need to be translated into real-valued feature vectors, which will be processed by subsequent layers of the network. Specifically,

each input is a sentence s treated as a sequence of words: $s = \{word_1, \ldots, word_{|s|}\}$, where the mapping from words to their word embeddings is performed by using Word2Vec or Skip-gram. Hence, for each input sentence s we build a sentence matrix $X_{1:|s|}$ where each row corresponds to a word embedding vector. For example, a sentence of length $|s|$ can be expressed as:

$$X_{1:|s|} = X_1 \oplus X_2 \oplus \ldots \oplus X_{|s|} \tag{4.3}$$

where X_i is a n-dimensional vector of the i^{th} word ($word_i$) in the corresponding sentence and $\oplus$ represents the concatenation operator.

(2) Convolutional Layer (Layer2)

The aim of the convolutional layer is to capture patterns, i.e., discriminative word sequences that are common among the training instances of each disease cluster. In order to form a richer representation of the data, our model applies M filters that work in parallel to generate multiple feature maps. For example, in Fig 4.6, we slide 2 filters over the word matrices, where the first one slides over 2 words each time and the second one slides over 3 words each time. For each convolution filter $F_k \in R^{h \times n}$, we generate a new feature map, where h is the length of the sliding window and n is the size of the input embedding. For each sliding window $X_{i:i+h-1}$, we compute:

$$C_{F_k^i} = f(W_k * X_{i:i+h-1} + b_k), \tag{4.4}$$

where $*$ denoted the convolution operation between an input matrix and a filter, f is an element-wise nonlinear transformation, W_k is the weight factor, $X_{i:i+h-1}$ is a matrix slice of size h along the rows, and b_k is the bias. Then, filter F_k is applied to all possible windows in the sentence, $\{X_{1:h}, \ldots, X_{|s|-h+1:|s|}\}$, to generate the characteristic mapping $C_{F_k} = [C_{F_k^1}, \ldots, C_{F_k^{|s|-h+1}}]$. Note that each component is the result of computing an element-wise product between a row slice of X and a filter matrix F_k. After this step, we can get M characteristic mappings, where each filter will generate one mapping correspondingly.

(3) Pooling Layer (Layer3)

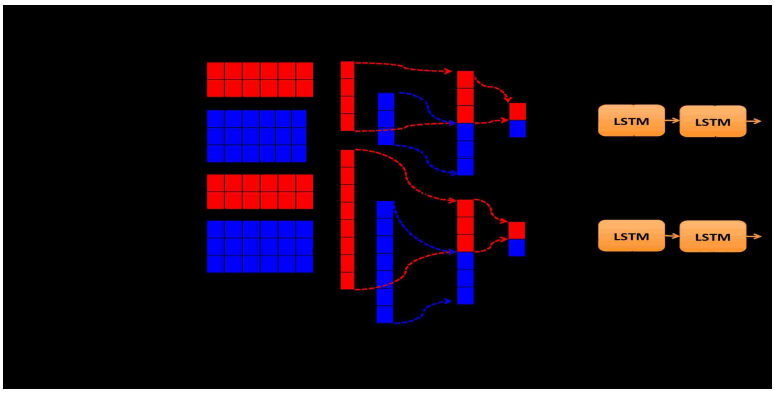


Figure 4.6: Topic Model Generation

The outputs from the convolutional layer are then passed to the pooling layer, whose goal is to aggregate the information and reduce the representation. Since both average and max pooling methods exhibit certain disadvantages: (i) in average pooling, all elements of the input are treated equally but we may want to give more weights to certain elements; (ii) the max pooling method may lead to overfitting on the training set. Thus, we have used both K-max pooling [84] and average pooling to generate local alignment representations, which contain important information of the corresponding questions.

Finally, the output of the pooling layer will be fed into a LSTM model.

Important Sentences Extraction via LSTM

Since not every sentence in the input questions is useful for the prediction task, we extract important ones using a LSTM model, where each time step of the model takes a sentence as an input.

Thus, to identify important sentences we need to select influential time steps, which can be evaluated based on their corresponding gradient variables. By studying those gradients in the networks, we can gain some insights into the internal mechanisms and identify important sentences. For example, at time step t, the model takes a sentence as input x_t, and updates the hidden state h_{t-1} to h_t using:

$$h_t = g(Wx_t + Uh_{t-1} + b), \tag{4.5}$$

where W and U are weight variables and g is a nonlinear activation function. Thus, in

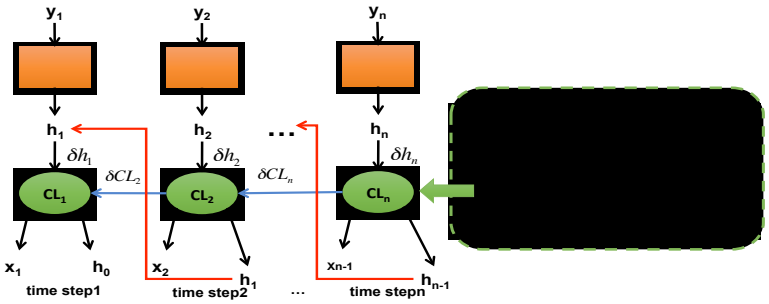


Figure 4.7: Back-propagation

order to infer the importance of a time step, it is necessary to explore the gradient variables such as δW and δU, since high values for these variables indicate this time step is important for the final prediction task.

Before we describe the computation process, we first give the definitions of various notations we use:

- a_t represents the input activation
- i_t represents the input gate variable
- f_t represents the forget gate variable
- o_t represents the output gate variable
- CL_t represents the cell state
- $\cdot$ represents the inner product
- $\otimes$ represents the outer products
- σ represents the sigmoid function: $\sigma(x) = \frac{1}{(1+e^{-x})}$
- $\odot$ represents the element-wise product or Hadamard product

During a forward propagation process, given the inputs we can compute a_t, i_t, f_t, o_t, CL_t and h_t for each time step t. In this work, we derive the network gradients analytically based on the back-propagation computation from the cell outputs all the way to the cell inputs (shown in Fig 4.7). Thus, in a LSTM time step t, we can compute the gradients weights (δW_t, δU_t) as follows:

Table 4.2: Top 10 Informative Keywords for A Liver Disorder Disease

Disease	Top 10 Informative Keywords
Liver Disorder	liver, respiratory, kidney, esophagus, pancreas, cardiomyopathy, aorta, bowel, aneurysm, clot

$$\delta W_t = [\delta a_t \odot (1 - tanh^2(a_t)), \delta i_t \odot i_t \odot (1 - i_t), \delta f_t \odot f_t \odot (1 - f_t), \delta o_t \odot o_t \odot (1 - o_t)]^T \otimes x_t \tag{4.6}$$

$$\delta U_t = [\delta a_t \odot (1 - tanh^2(a_t)), \delta i_t \odot i_t \odot (1 - i_t), \delta f_t \odot f_t \odot (1 - f_t), \delta o_t \odot o_t \odot (1 - o_t)]^T \otimes h_{t-1} \tag{4.7}$$

where all necessary variables can be computed via Algorithm 1: (i) at step1, we compute $\{\delta o_t, \delta CL_t\}$; (ii) at step2, we compute $\{\delta a_t, \delta i_t, \delta f_t\}$.

Algorithm 1 Gradients Computation

1: $Input: ForwardPass\ CL_t = i_t \odot a_t + f_t \odot CL_{t-1};\ h_t = o_t \odot tanh(CL_t)$;
2: (1) $step1:\ Given\ \delta h_t,\ Compute\ \delta o_t,\ \delta CL_t$
3: $(i)\ \delta o_t = \delta h_t \cdot \frac{\partial h_t}{\partial o_t} = \delta h_t \cdot tanh(CL_t)$;
4: $(ii)\ \delta CL_t = \delta h_t \cdot \frac{\partial h_t}{\partial CL_t} = \delta h_t \cdot o_t \cdot (1 - tanh^2(CL_t))$
5: (2) $step2:\ Given\ \delta CL_t,\ Compute\ \delta i_t,\ \delta f_t,\ \delta a_t$
6: $(i)\ \delta i_t = \delta CL_t \cdot \frac{\partial CL_t}{\partial i_t} = \delta CL_t \cdot a_t \cdot i_t \cdot (1 - i_t)$;
7: $(ii)\ \delta f_t = \delta CL_t \cdot \frac{\partial CL_t}{\partial f_t} = \delta CL_t \cdot CL_{t-1} \cdot f_t \cdot (1 - f_t)$;
8: $(iii)\ \delta a_t = \delta CL_t \cdot \frac{\partial CL_t}{\partial a_t} = \delta CL_t \cdot i_t \cdot (1 - a_t^2)$;

Topic Model Module Construction

To allow our query processing module to better understand users' questions, we incorporate different question representations to identify key information. Specifically, we find important keywords by considering: (i) the similarity of keywords in the corresponding sentence with the disease labels based on their tf-idf values; (ii) the importance of each sentence, which is calculated based on Algorithm 1. As such, we define an attention weight for each keyword $word_i$ of a disease c, aw_i, using the following equation:

$$aw_i = \frac{\sum_{j=1}^{N} SC_j * tf_j^i * sim_{ic}}{TF^i}, \tag{4.8}$$

where N is the total number of sentences that belongs to the disease c, SC_j is the

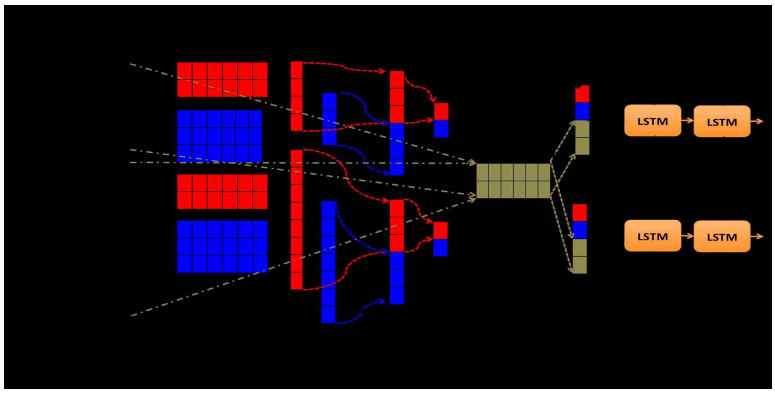


Figure 4.8: Query Processing Module

important score of j^{th} sentence, tf_j^i is the occurrence of keyword $word_i$ in the j^{th} sentence, sim_{ic} is the similarity score between $word_i$ and disease label c and TF^i is the total number of occurrences of $word_i$.

With such attention weights, we can identify important sentences and keywords associated with each disease cluster. Thus, we can generate a topic model module (TM) with multiple clusters, where each cluster represents a disease type and contains informative keywords. For example, Table 4.6 is one topic model cluster for a liver disorder disease, which contains 10 identified informative keywords (ordered by their attention weights).

4.4.3 Query Processing Module Generation

Here, we describe how we train a query processing module which can consider both the common and specific features of patients' questions with the help of the topic model module such that the learnt query processing module understands similar disease keywords in related patients' queries and hence can direct these queries to relevant medical diagnosis modules for personalized disease prediction.

We first separate patients' medical questions in the training set into different clusters based on their suffered diseases, where each cluster can be seen as a sub-population with similar patients. Then, we apply a CNN-based word embedding method to turn sentences in patients' queries into word vectors that can be fed into a $LSTM$ model, which learns the semantic connections among related sentences. The detailed training procedure is described as follows:

1. We extract patients' real-world questions of each disease from web blogs.

2. Since unlike other domains where the data are clean and well-structured, healthcare data are highly heterogeneous and are prone to significant noise. Thus, in our work, we use the following two rules to remove the unrelated questions: (1) Check if the question's title, content or answers contain the target disease keywords; (2) Check if the question's author has joined the target disease community.

3. We then separate the filtered questions into different clusters based on their related diseases.

4. Next, we collate all patients' questions and generate user-centric vector-representations and use them to train a query processing module (as shown in Fig 4.8). In our work, we apply a CNN architecture for sentence-level analysis, where the sentence representation vector is followed by a fully-connected layer with multiple "sliding window filters".

(1) Layer 1 (Sentence Matrix): In this layer, we employ a word embedding model (e.g., Word2Vev or Skip-gram) to convert an input sentence $s = \{word_1, ..., word_{|s|}\}$ into a matrix, where each row of the matrix corresponds to one word. After the word matrix layer, we have d_e dimensional pre-trained embeddings for each word in the sentence, which are vertically concatenated into the matrix $X_{1:|s|} = [X_1, \ldots, X_{|s|}]$, where $|s|$ is the length of the sentence.

(2) Layer 2 (Convolution): In the second layer, we perform convolution operations on the matrices using M filters, which aims to extract patterns, i.e., discriminative word sequences found within the input sentences. The filters are used to operate on the sliding window of length h to generate a new feature. For instance, a sliding window $X_{i:i+h-1}$ generates the feature via a filter F_k according to the following equation:

$$C_{F_k^i} = f(W_k * X_{i:i+h-1} + b_k), \tag{4.9}$$

where W_k is the weight factor, b_k is an offset term and f is a non-linear function. This filter is applied to all possible windows in the sentence, $\{X_{1:h}, \ldots, X_{|s|-h+1:|s|}\}$, to generate the characteristic mapping $C_{F_k} = [C_{F_k^1}, \ldots, C_{F_k^{|s|-h+1}}]$.

(3) Layer 3 (Pooling): The output (e.g., M characteristic mappings) from the convolutional layer are then passed to the pooling layer, whose goal is to keep the most important

features and reduce the representation. Due to the limitations of the pooling methods, the specific pooling function we used includes K-max pooling [84] and mean-pooling.

(4) Layer 4 (Similarity): The similarity layer is used to compute the similarity scores between sentences within a question and the keywords within a topic model cluster of a particular disease type.

For example, given an output from Layer 3 of an input sentence (e.g., vector representations x_t) and i^{th} cluster of the topic module TM_i, a sentence-topic similarity score can be computed as follows:

$$sim(x_t, TM_i) = \frac{\sum_{j=1}^{|TM_i|} (TM_i^j \cdot x_t^T) * aw_i^j}{|TM_i|}, \tag{4.10}$$

where TM_i^j and aw_i^j are the word vector and attention weight of informative keyword $word_j$ in the i^{th} cluster of the topic model module respectively. We apply the computation on the whole topic model $TM = \{TM_1, TM_2, \cdots\}$ and get a vector, where each component of the vector represents a similarity score between that sentence and the corresponding disease.

5. Finally, all intermediate vectors from Layer 3 and the similarity score features are concatenated together and fed into a two layer $LSTM$ model, which learn long-term correlations among words.

4.5 Performance Evaluation

In this section, we apply our model to real clinical data extracted from the clinical web sites and blogs. We first describe our experimental settings and the data we use. Then, we provide the evaluation analysis.

4.5.1 Experimental Settings

To evaluate the performance of our scheme, we conduct experimental evaluations on real-world data:

(1) A disease training set which contains 2200 popular searched diseases (with professional medical names), with 1526 symptoms and 526 risk factors, where each of these 2200

diseases, its ICD-10 and SNOMED-CT codes together with limited symptoms and risk factors is listed. For symptoms related to a disease, we also label each of them with different levels (0-4), where a higher level means such symptom is more unique for the corresponding disease.

(2) A test set crawled from the website 1, which contains 400 most frequently searched symptom and risk factor lists with their corresponding correctly diagnosed diseases to evaluate our medical diagnosis modules. Each instance in the test set contains a list of symptoms, risk factors and an ordered list of expected diseases denoted as $\{sympt\ list||rfactor\ list\ ||expected\ \ disease\ list\}$). The first disease in that ordered list is the most likely disease associated with those symptoms and risk factors.

(3) Medical Websites: website 1 (contains 2049 diseases), website 2 (contains 1854 diseases), etc. which contains multiple expert knowledge such as disease descriptions, symptoms, risk factors, treatments, etc. We would like to integrate the information gleaned from these websites with the disease-symptom-risk factor database to generate an aggregate dataset for training our disease diagnosis modules.

(4) Medical Web Blogs: website 3, etc, which contains multiple medical questions.

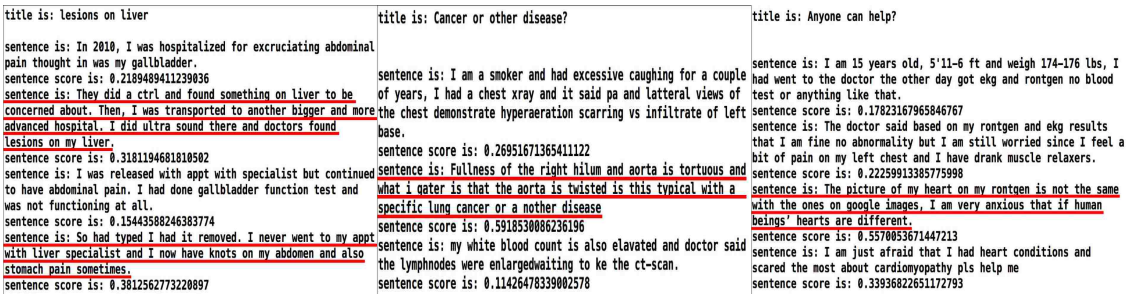
In the experiments, we set α as 0.3 in the similarity matching process and use 2 pooling layers (one is for k-max pooling and the other is for average pooling) in the query processing module. We also select 512 hidden units in our LSTM model and trained it for 300 epochs, where each epoch is defined as the process of feeding the whole training set to a model. All our experiments are conducted on a Mac Pro with an Intel Core i5 processor running at 2.6GHz and 8GB memory.

4.5.2 Experimental Evaluation

Similarity Matching Method Evaluation

First, we conducted Exp1 to measure the performance of our proposed similarity matching algorithm.

Exp1: In this experiment, we use the disease dataset, which contains 2200 popular diseases. Then, we apply the proposed similarity matching method to link those 2200



(a) Sentences Interpretation of a Liver Disorder Related Question (b) Sentences Interpretation of a Lung Cancer Related Question (c) Sentences Interpretation of a Heart Attack Related Question

Figure 4.9: Sentences Interpretations

Table 4.3: Evaluation of Similarity Matching Method

Web Site	# of diseases	# of web pages	Accuracy
Website 1	2200	2049	93.0%
Website 2	2200	1854	89.4%

diseases with 2049 and 1854 web pages extracted from website 1 and website 2 respectively.

We manually check the results (in Table 4.3) and find that for those 2200 diseases, our proposed similarity matching algorithm can achieve

(1) 93.0% accuracy for website 1. Among those 1847 diseases matched correctly, 1421 matched diseases have exactly the same name on the web pages while 426 matched diseases have slightly different names. For example, the "anorexia" disease is named as "fastidium" on website 1. An error example is as follows: "gingivostomatitis" is matched to a web page for "vincent-stomatitis" in website 1 but these are two different diseases.

(2) 89.4% accuracy for website 2. Similarly among 1770 correctly matched diseases,

(a) Allergy Related Query (b) Miscarriage Related Query (c) Respiratory Related Query

Figure 4.10: Patients' Clinical Questions

Table 4.4: Evaluation of Medical Diagnosis Modules

Evaluation	Model	Accuracy
Exp2	NB	31.8%
	RF	45.3%
	LR	64.6%
Exp3	NB	53.2%
	RF	63.9%
	LR	86.1%

1386 diseases are matched with the exact same names on the web pages while 384 diseases are matched with slightly different names.

We generate a revised training dataset by integrating the symptoms and risk factors extracted from website 1 and website 2 to the disease training set. We also asked medical experts to set the level values for newly included symptoms for each diease. This aggregated dataset is used to train medical diagnosis modules in Exp2 & Exp3.

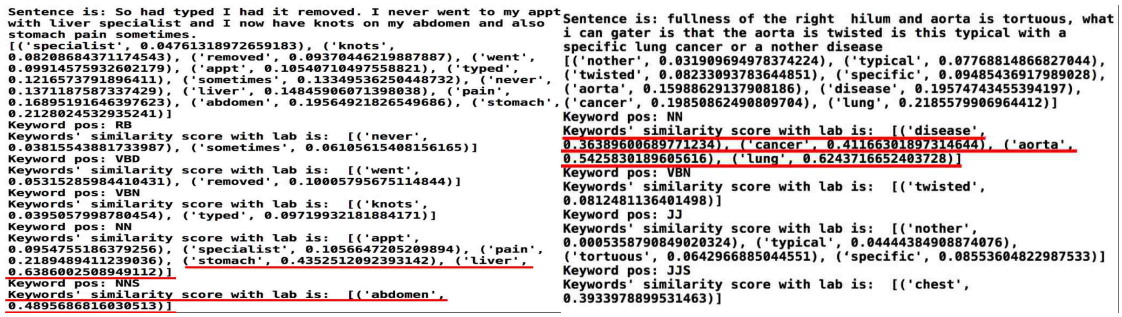
Medical Diagnosis Modules Evaluation

In this subsection, we conduct two experiments (Exp2 & Exp3) to evaluate our simple disease diagnosis modules by utilizing expert knowledge database.

Exp2: We use three different methods to train the diagnosis modules, including Naive Bayes (NB), Logistic Regression (LR) and Random Forest (RF). In this experiment, we only use the symptoms and disease lists information from the aggregated dataset produced from Exp1 and evaluate the trained modules by checking if the top1 disease included in the expected disease list also ranks as the top1 in our diagnose disease list.

Exp3: Instead of only using symptoms as in Exp2, we use both symptoms and risk factors to train diagnosis modules.

The results are shown in Table 4.4. From the results, we can clearly observe that logistic regression model performs the best among all the learning models. This is because in the logistic regression model, a higher level symptom or a more unique risk factor will be associated with a higher coefficient value, which indicates its marginal contribution to the predicted risk. Meanwhile, by comparing the results in Exp2 and Exp3, we find that it is better to use both symptoms and risk factors to conduct disease diagnosis since some



(a) Words Interpretation of a Live Disorder Related Question

(b) Words Interpretation of a Lung Cancer Related Question

Figure 4.11: Patients' Clinical Questions

diseases may have similar symptoms but different risk factors.

Interpretation Results Evaluation

(1) Interpretation Through Important Sentences

Using the techniques explained in Section 4.4.2, we can identify important sentences in our input data (medical blog questions) based on the computed gradients. For example, for a liver disorder related question (shown in Fig 4.9(a)), by computing the gradients for the corresponding time steps, we find that the 2^{nd} and 4^{th} sentences are more important than the others for the final prediction. Fig 4.9(b) and Fig 4.9(c) provide two more examples of questions for other diseases such as lung cancer and heart attack.

(2) Interpretation Through Important Keywords

According to Section 4.4.2, we can capture informative keywords in the input medical questions based on their attention weights. For example, for liver disorder and lung cancer related questions (shown in Fig 4.11(a) & Fig 4.11(b)), we can extract (i) "abdomen", "stomach", and "liver" for a liver disorder disease; (ii) "cancer", "aorta", and "lung" for a lung cancer disease, which are more informative keywords than others for the final predictions.

(3) Topic Model Module Evaluation

Therefore, by analyzing all medical questions we can obtain most informative keywords for every disease, which can be used to conduct the topic module. For example, Table 4.5 shows top 10 extracted informative keywords for liver disorder, heart disease and diabetes

Table 4.5: Top 10 Informative Keywords for different Diseases

Disease	Top 10 Informative Keywords
Liver Disorder	liver, respiratory, kidney, esophagus, pancreas, cardiomyopathy, aorta, bowel, aneurysm, clot
Heart Disease	heart, heartbeat, cardiac, coronary, artery, arrhythmia, stroke, kidney, cardiomyopathy, cardiogram
Diabetes	diabetes, cholesterol, arthritis, hypoglycemia, cancer, insulin, pancreatitis, neuropathy, metformin, celiac

diseases respectively.

Query Processing Module Evaluation

In order to train and test the query learning module, we extract 18,000 questions from 6 different disease clusters (i.e., "diabetes", "pregnancy", "heart", "lung & respiratory disorders", "cancer", "general health"), where each disease cluster contains 3000 clinical questions on the average. We refer to these questions as the clinical question dataset.

Then, we conduct Exp4 to evaluate the performance of our query processing module.

Exp4: We first randomly choose 2 groups of disease clusters where users often post questions on more than one cluster, namely (i) G1 consists of "diabetes", "pregnancy" and "heart" clusters and (ii) G2 consists of "diabetes", "lung & respiratory disorders", and "cancer" clusters. For each cluster, we select 1000 clinical questions as training set and 500 clinical questions as testing set. Then, we filter the questions by removing the unrelated ones (described in Section 4.4.3). Next, based on the processed dataset, we use different word embedding methods together with the $LSTM$ structure to conduct this experiment: (1) Word2Vec tool (GloVe) on the GoogleNews dump: We adopt a pre-trained word vector model to map each question sentence into a vector. This pre-trained word vector model contains 3 million words and hence include all the words typically used in patients' queries; (2) CNN based word embedding (in Section 4.4.2); (3) CNN based word embedding with the topic module (TM) (in Section 4.4.3).

The results for both G1 and G2 datasets are tabulated in Table 4.6. From the results, one can see that using the convolutional neural network (CNN) based word embedding

Table 4.6: Impact of Different Word Embedding

Group	Embedding Method	Train Size	Test Size	Accuracy
G1	Word2Vec	2186	1031	58.2%
G2	Word2Vec	2402	1145	66.5%
G1	CNN only	2186	1031	71.1%
G2	CNN only	2402	1145	79.6%
G1	CNN + TM	2186	1031	85.8%
G2	CNN + TM	2402	1145	87.4%

(a) A Diabetes Related Question (b) A Diabetes Related Question

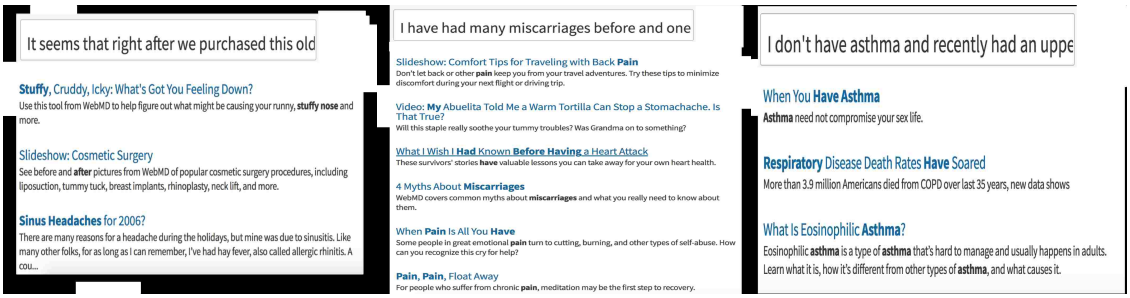
Figure 4.12: Diabetes Related Questions

method improves the accuracy by about 13%. It is because that the CNN based word embedding method allows the $LSTM$ model to efficiently learn important syntactic and semantic aspects of the inputs when appropriate widths of the convolution filters (e.g., bigrams or three-grams) are used.

In addition, we also find that the performance improves by 11% on average if we include the topic model module in our query processing model. It is because the topic model module can deal with questions, which contain multiple disease keywords (both related and unrelated). For example, for a diabetes related question (shown in Fig 4.12(a)), which contains both diabetes and pregnant related keywords, one can see that it contains more important diabetes related keywords such as “insulin”, “diabetes”, ”eat”, and “food”, and hence our query processing model incorporating topic model can produce the correct prediction.

Compare With the Existing Website

In this subsection, we conduct Exp5 to compare the performance of our system with the existing professional medical website 4, where patients can receive responses about potential illnesses they may have based on the symptoms they submit.



(a) Response of Allergy Query (b) Response of Miscarriage Query(c) Response of Respiratory Query

Figure 4.13: Questions Responses

Table 4.7: Compare With the Existing Website

Model	Method	Train Size	Test Size	Accuracy (top1)	Accuracy (top3)
$DL-MDS$	CNN + TM	9474	2196	91.6%	93.8%
Website 4			2196	72.9%	76.5%

Exp5: We first randomly select 2000 clinical questions for each disease cluster. Then, we filter out those unrelated questions and set the remaining ones as the training set. Overall, we have 9474 clinical questions for 6 disease clusters in our training set. We also select 500 clinical questions for every disease cluster, filter unrelated ones and set the remaining as the testing set. Overall, we have 2196 clinical questions for 6 disease clusters in our test set. Next, we use CNN based word embedding method to generate the sentence based word vectors and insert them into a $LSTM$ based learning model. Finally, we use the testing set to measure the performance of our system and the medical website 4, where the accuracy is evaluated by checking if the top1 or top3 returned diseases match with the correct ones that the clinical questions should match to.

The results are shown in Table 4.7. From the results, we can see that our system has better performance in handling the clinical questions than the professional medical website (by 18%). We also find that the accuracy improves when we measure the accuracy by checking if the correct disease is among the top3 predicted disease clusters instead of only considering the top1 disease cluster. We looked into the limitations of the website 4 which prevent them from handling certain questions. We found that such website cannot handle the following three types of clinical questions:

(1) Type1: If the clinical question contains multiple keywords related to symptoms but not the exact disease related keywords, then the website 4 cannot infer the correct disease. For example, an allergy related question (Fig 4.10(a)) on the medical website returns "Stuffy nose" (Fig 4.13(a)) as its returned disease, which is not as accurate as what our model predicts.

(2) Type2: If the clinical questions contain multiple disease keywords (related & unrelated), where the unrelated keywords could be some general medical terms which occur more often than the related ones, then there is a high probability that the unrelated clinical disease will be the top1 disease returned by the website 4. For example, for a miscarriage related question (Fig 4.10(b)), the medical website will diagnose such query as "Back pain" (Fig 4.13(b)) since the keyword "pain" occurs more often than other medical terms. It seems like the medical website does not consider semantic correlated terms together but our model is able to learn such reasoning to return the right diagnosis.

(3) Type3: If the clinical questions are short and contain some negative or transitional contexts, then the website 4 cannot understand given queries and may provide wrong answers. For example, for an respiratory related question ((Fig 4.10(c))), the medical website will cluster such query as an "Asthma" disease (Fig 4.13(c)).

Compare With Existing Work

In this subsection, we conduct Exp6 to compare the performance of our system with the existing work $SCDL$ [120], which is used to infer the possible diseases by giving the questions of health seekers. The $SCDL$ system is constructed via alternative signature mining and pre-training incrementally. It consists of two key components: (i) the first part globally mines the discriminate medical signatures from raw features; and (ii) the second part learns the inter-relations between the raw features and their signatures.

Exp6: In this experiment, we first select a dataset (DataIII) from [120], where 20% of the data (in DataIII) are selected for testing. Then, we use the same process as mentioned in Exp5 to train our query processing module. Next, we use the selected test set to measure the overall accuracy of our scheme and compare the result with the $SCDL$ scheme.

From the results (Table 4.8), we can see that our scheme performs slightly better than

Table 4.8: Compare With Existing Work

Model	Embedding Method	Data Size	Accuracy
$DL-MDS$	CNN + TM	1587	93.69%
$SCDL$		1587	91.48%

[1]The results of the $SCDL$ scheme are extracted from [120];

$SCDL$. It is because we have used a two layer LSTM model, where the first layer will act as a feature extractor and the second layer will learn more correlations among dominant features. In order to learn the limitations of our scheme, we also look through all the test cases and find that our scheme performs poorly when the clinical questions contain multiple disease keywords (related & unrelated), where the patients ask about one undiagnosed disease but who already have been diagnosed with another disease. For example, for a heart disease related question "I recently got my blood tests back and am not sure if I have 'heart disease' in the classical meaning of the phrase or just a bad reading which need to be addressed. I am a recently turned 60 year old man and was diagnosed with Diabetes almost 2 years ago.", our scheme produces "diabetes" as the top rank disease while "heart disease" is in the Top 3 list. In addition, we also find that the authors in [120] do not consider the following two scenarios in their $SCDL$ scheme, which can be handled by our DL-MDS: (i) it is difficult to prove that $SCDL$ can deal with type1 questions (described in Section 4.5.2) since such type of questions is not included in their datasets; and (ii) in order to improve their performance, the authors in [120] have designed a negation filtering approach, where questions that contain some negative or transitional contexts will be removed. However, the authors do not show if their $SCDL$ scheme can deal with such type of questions in their

.

Discussion

By looking into those wrongly predicted cases, we find that the our query learning model still cannot handle the following case: if the question contains only general words or multiple important keywords related to different diseases, then there is a high probability that it will be predicted wrongly. For example, for a diabetes related question (shown in Fig 4.12(b)), it will be predicted wrongly since it includes multiple informative keywords (e.g., "breast",

"chest", "ovarian", "pregnancy", "obesity", "diabetes") associated with different diseases.

4.6 Summary

In this chapter, we have proposed a deep learning based medical diagnosis system (DL-MDS), which can be used to aid efficient clinical care, where authorized users can conduct searches for medical diagnosis. One of our major contributions is that we have designed a medical knowledge extraction framework to collect useful data from multiple sources, so that we can further generate medical diagnosis models. The other is that we have generated a deep learning based model, which allows authorized users to conduct searches for medical diagnosis based on their personalized queries.

Chapter 5

New Attacks on RNN based Healthcare Learning System and Their Detection & Defense Mechanisms

5.1 Background

The coming age of the science of machine learning (ML) coupled with advances in computational and storage capacities have transformed its technology landscape. For example, ML models have proved to be very efficient in classifying images, as shown by the impressive results of deep neural networks on the ImageNet [12] Competition. Meanwhile, many real-world applications also require effective machine learning algorithms that can learn invariant representations from time-series datasets. Deep Neural Networks (DNNs) have achieved great success in various tasks, including but not limited to image classification [90], speech recognition [75], machine translation [24], and autonomous driving [47]. For example, pre-trained Recurrent Neural Networks (RNN) [50, 49] have been proposed to extract meaningful features from a large amount of EHR data which can be used to create models for disease diagnosis, disease progression or making early treatment decisions by

clinicians.

However, training these models requires very large datasets and is quite time consuming. Thus, instead of building from scratch, many ML systems are "composed" of an array of heterogeneous components, denoted as primitive learning modules (PLMs), which are often pre-trained on massive amounts of data and provide modular functionalities (e.g., feature extraction). Thus, with a large collection of PLMs on large and challenging datasets that are released by many research institutions (e.g., Model Zoo [15], Keras application [13], etc.), even a student without much computer science background can built a full-fledged medical system, which can ultimately contribute to better personalized healthcare quality.

While using PLMs is efficient and beneficial, researches have recently discovered that such PLMs [147, 68] are vulnerable to two types of attacks: (i) attackers can add carefully crafted perturbations to the inputs of the targeted model to affect the prediction performance. For example, attacks have been conducted on the RNN medical models by modifying the important information within a sequence of medical data such that the changes are not noticeable. (ii) most PMLs that are provided by third parties lack proper vetting to ensure their safe operations. For example, attackers could create substitute models based on responses received from cloud-based ML systems with their carefully crafted queries to avoid having to pay for services of cloud-based ML systems.

Thus, to prevent such security risks of reusing publicly available PLMs in ML system development and make deep neural networks more robust to adversarial attacks, recent work has proposed several defensive algorithms [125, 79, 92, 164, 172]. Those defenses can be grouped under three different approaches: (i) training the target classifier with adversarial examples, called adversarial training [147, 68]. (ii) modifying the training procedure of the classifier, e.g., defensive distillation [126], and (iii) quantizing neural network weights and activation functions into low bit-width [165, 130, 161, 21].

However, all these approaches have limitations. For example, Carlini et al. [40] have shown that defensive distillation which requires changing and retraining the target classifier does not significantly improve the robustness of neural networks. Meanwhile, some of those defenses require adversarial examples to train the model and are devised with specific attack models in mind, which are not effective against new attacks. This motivates us to find better

methods to defend these attacks.

In this chapter, we demonstrate two types of harmful threats (in sections 5.3 & 5.4) to RNN-based ML systems by introducing attack methods which can trick such ML systems to output wrong prediction results. For the first attack, we propose an efficient and effective framework to generate adversarial samples with minimum perturbations on input medical sequences. For the second attack, instead of using adversarial samples, we propose a novel attack strategy, which is to randomly add gradient noise to adjust recurrent weights, where the gradients noise are propagated through a set of important weight vectors until those features associated with the modified weights become insignificant during training and hence produce a ML model which generates wrong classification labels for targeted classes.

In addition, in section 5.5.1 and section 5.5.2 we propose two detection schemes for identifying these two types of attacks.

- Recent studies have shown that adversarial samples are much more sensitive to perturbations than normal samples. If we impose random perturbations on a normal and an adversarial samples respectively, there is a significant difference between the ratio of label change due to the perturbations. Thus, in section 5.5.1 we present a recent work on detecting adversarial samples, where the authors determine if an input is a normal sample or an adversarial one through mutation testing.

- Based on our observations, we notice that important features identified from a genuine RNN-based model are significantly different from those identified from a maliciously modified model created from the original model. Thus, in section 5.5.2 we design a detection scheme, which compares the identified important features of a similar ML system (e.g., a traditional random forest model) with those obtained using the downloaded RNN-based ML system which may be maliciously modified. Significant differences in these two sets of identified features suggest that the newly created RNN-based ML system may contain malicious PLM. Our detection scheme can be easily implemented to quickly identify any malicious RNN-based ML systems.

Moreover, we also propose a Static/Dynamic Quantization-based defense solution (SDQDS) (in section 5.5.3), which uses quantized weights and does not require any modification to

the training procedure and yet relatively effective in defending against these two types of adversarial attacks.

Finally, in section 5.6, we conduct various experiments using both synthetic and real healthcare datasets which contain patient records with many attributes to demonstrate that these two types of attacks are effective against RNN-based healthcare machine learning models. The evaluation results for our detect and defense solutions also show that they are feasible and practical.

In summary, in this chapter, we make the following contributions:

- We present two potential attacks on a RNN-based ML system and validate the feasibility of such attacks and the consequential damage they create.
- We design easily implemented low-cost detection and defense schemes to prevent such adversarial attacks.
- We conduct extensive experiments using both synthetic and real-world datasets to show the effectiveness of those attacks and robustness of our detection & defense schemes.

5.2 Important Building Blocks

In this section, we introduce some fundamental concepts that used in our work.

5.2.1 ML System

Machine learning (ML) is the use of artificial intelligence to automate algorithms that can learn without explicit instructions. Depending on the algorithms, objectives and their purposes, ML systems can be categorized into three main categories: (a) supervised learning; (b) unsupervised learning; (c) reinforcement learning. The first one is mainly used for classification, where the network is trained with labeled dataset to learn some connections between inputs and outputs. The second one is often used for feature extraction and network pre-training, which is trained with unlabeled dataset. The last one can be viewed as the

subfield of ML which explores techniques to improve planning and control policies based on observations.

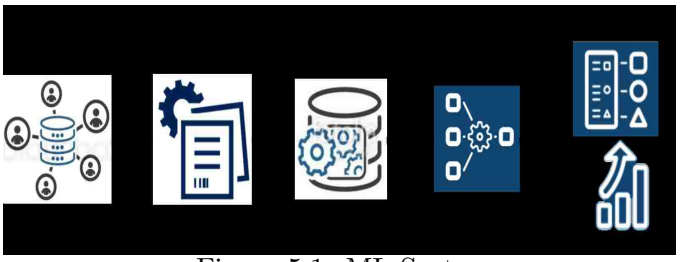


Figure 5.1: ML System

An end-to-end ML system often consists of a set of modules, where each one implements different functionalities such as data collection, data processing (normalization), feature selection, regression or classification (as shown in Fig 5.1). We primarily focus on primitive learning models (PLMs) that implement feature extractors, due to their prevalent use and reusable nature.

Once the data is collected and pre-processed, a ML model is chosen and trained. The machine learning process consists of finding the relationship between the patterns and the outcomes using solely the training examples, where the input data point (x, y) are samples obtained by sampling from a fixed but unknown probability distribution over the space Z=X $\times$ Y. Here, X is the space of feature values and Y is the space of labels (e.g., Y = $\{0, 1\}$ for classification or Y = R for regression).

A feature extractor implements a function $g : X \rightarrow V$ which extracts a subset of the initial features from the input data, so that the desired task can be performed by using this reduced representation instead of the complete initial data. The mapping from V to Y is performed via a function $f : V \rightarrow Y$ associated with a loss function L_f, which captures the error made by the prediction f(g(x)) when the true label is y.

5.2.2 PLMs Development Status

The national push for electronic health records (EHR) has resulted in an exponential surge in volume, detail, and availability of digital health data, which offers an unprecedented opportunity to infer richer, data-driven descriptions of health and illness. Clinicians are collaborating with computer scientists to generate interesting ML systems to improve the state of healthcare services towards the goal of personalized treatments.

Thus, there has been a widespread use of ML systems in medical-diagnosis applications. For example, Recurrent Neural Network (RNN) models [66, 123] recently proposed for the healthcare domains have demonstrated significant prediction improvements over traditional approach. RNN is a variant of neural networks which can be used to handle large sequential data. It takes an input sequence $(x_1, x_2, \ldots, x_t)$, produces a sequence $(h_1, h_2, \ldots, h_t)$ of hidden states and outputs a sequence $(y_1, y_2, \ldots, y_t)$ in the following way: $h_t = \sigma(W_x x_t + W_h h_{t-1} + b_h)$, $y_t = W_y h_t + b_y$, where σ is the logistic sigmoid function, W_x, W_h, W_y are weight matrices and b_h, b_y are biases. RNNs also allow connections between hidden units that form directed cycles. Such cycles allow the network to keep previous information of hidden states as an internal memory. One example is the Long Short-Term Memory (LSTM) architecture, which addresses the problem of learning long-term dependencies by introducing a gated structure, which is able to learn long term dependencies.

Therefore, with a large collection of PLMs available for off-the-shelf use, developers can download the PLMs and integrate them to form various ML systems for multiple classification tasks. However, the downloaded PLMs may have some security risks. For example, a ML classifier with malicious feature extractor, may assign a wrong class to a legitimate health related input, or classify noisy input with confidence. A maliciously crafted unsupervised feature extractor may produce a meaningless representation of the input, which may largely affect the learning performance.

5.2.3 Attack Threats

Here, we assume there are two kinds of attacks: (i) the attacker generates adversarial samples based on their knowledge of the victim models and the inherent characteristics of the original datasets; (ii) the attacker crafts a malicious feature extractor PLM by slightly modifying its genuine version. Thus, the system developers will be affected if they accidentally download such malicious samples or module and integrate it with their classifiers to build ML systems.

(1) Adversarial Samples Attack

Recent studies have shown that adding small modifications to an input can fool state-of-the-art deep classifiers, resulting in incorrect classification [147, 68, 125, 123, 40]. Their algorithms perturb normal samples by a small amount such that such perturbations are not noticeable but can cause mis-classifications by the learning systems.

Most studies of adversarial examples in the literature use the white-box assumption. For example, Carlini et. al. [40] present a powerful C&W attack, where carefully crafted perturbations are added to the legitimate input samples. While various techniques have been proposed to generate adversarial samples of the white-box attack, researchers also explore the black-box attack where adversaries do not know the model structures or parameters which is a more realistic scenario. For example, one study proposed by Papernot et. al. in [125] showed that it is possible to create adversarial samples that successfully reduce the classification accuracy without knowing the model structure or parameters.

(2) Weights Adjustment Attack

We explore the range of attacker capabilities in machine learning systems either during inference or training phases. The first type of capabilities allows attacks during inference time, where exploratory attacks do not tamper with the targeted model but instead either cause it to produce specifically targeted but wrong labels as outputs or simply use attacks to collect information about the model characteristics so that they can build substitute models without paying for the use of the real models. Such inference-time attacks can be classified into either (a) white-box: the adversary has full knowledge of the model, including the model architecture, parameters, and original training data; (b) black-box: attacks assume no knowledge about the model. The second type of attack capabilities allow attacks during the training phase. Attackers with such capabilities attempt to learn, influence or corrupt the model itself during the training phase by either inserting adversarial inputs into the existing training data or crafting a malicious feature extractor so that any ML-system that incorporates such malicious feature extractor will produce wrong prediction results. Typically, such attackers [170] will modify genuine PLMs and upload them to developer

platforms such as github and attract others to download by making certain interesting claims about their malicious PLMs e.g., they run faster.

5.2.4 Static/Dynamic Quantization-based Defense Solution

A lot of quantization based techniques have been proposed recently, where the original intention is to compress the neural networks. Broadly speaking, these techniques can be classified into two types: deterministic (static) quantization [104, 70] and stochastic (dynamic) quantization [174, 160, 21, 130]. In deterministic quantization, the weights are quantized into some predefined codebook, where there is an one-to-one mapping between the quantized value and the real value. For example, some commonly used codebooks are {1, 1}, {1, 0, 1} or power-of-two numbers, which provides binary network, ternary network and power-of-two network separately. While in the dynamic quantization, the adaptive codebook and quantized values are learned from the data, where the weights, activations or gradients are discretely distributed. The quantized value is sampled from some discrete distributions. In this subsection, we review different design choices for static/dynamic quantization, which will be used in our work.

(1) Static Quantization

(A) Uniform Quantization:

Firstly we scale weights into a range $\in$ [-1, 1], and then adopt the following k-bit fixed quantization:

$$q_k(x) = 2(\frac{round[(2^k-1)(\frac{x+1}{2})]}{2^k-1} - \frac{1}{2}), \tag{5.1}$$

where rounding is possibly the simplest way to quantize real values. In this way, the output tensors will only contain 2^k discrete levels in the range of -1 to +1. For example, when n=1 the output will be quantized to 2 discrete levels -1 and +1. However, this method can be far from optimum when quantizing non-uniform data, which are typical characteristics of trained weight and activation values.

(B) Threshold-based Non-uniform Quantization:

In order to round off a floating-point number to the nearest fixed-point representation, in [70] the authors proposed the following rounding scheme:

$$\begin{cases} \lfloor x \rfloor, \ if \ \lfloor x \rfloor \leq x \leq \lfloor x \rfloor + \frac{\epsilon}{2}, \\ \lfloor x \rfloor + \epsilon, \ if \ \lfloor x \rfloor + \frac{\epsilon}{2} < x \leq \lfloor x \rfloor + \epsilon \end{cases} \tag{5.2}$$

where ϵ is the smallest positive number that can be represented in this fixed-point format. $\lfloor x \rfloor$ is defined as the largest integer multiple of ϵ. For values that are beyond the range of this fixed-point format, the authors normalized them to either the lower or the upper bound of the fixed-point representation.

Challenges: while use a rounding function is an easy way to convert real values into quantized values, the network performance may drop dramatically after each rounding operation. It is necessary to keep the real values as reference during training which increases the memory overhead. Meanwhile, since the parameter space is much smaller if we use discrete values, it is harder for the training process to converge. Finally, rounding operation cannot exploit the structural information of the weights in the network, where the activations, gradients and parameters have very different ranges and those ranges of the gradients slowly diminish during training.

(2) Stochastic (Dynamic) Quantization

In the previous method, the thresholds between different quantization levels are fixed. Here, we propose a dynamic activation quantization method in which the thresholds for different discrete levels are tunable parameters in adversarial training. For n-bit quantization function, it has 2^n output discrete levels and $2^n - 1$ tunable thresholds. Let $m = 2^n$, then the quantization will have $m - 1$ threshold values $\{t_1, t_2, \ldots, t_{m/2}, \ldots\}$. For example, when n = 1 and 2, the 1-bit and 2-bit dynamic quantizations are:

$$f(x) = 0.5 \times [sgn(x - t_1) - sgn(t_1 - x)], and \tag{5.3}$$

$$\begin{cases} -1, \ if \ x < t_1 \\ t_1, \ if \ t_1 < x < t_2 \\ t_3, \ if \ t_2 < x < t_3 \\ +1, \ if \ x > t_3, \end{cases} \tag{5.4}$$

which is more flexible and desirable than the previous approach.

(3) A Comparison of Two Quantization Methodologies

In order to achieve static quantization, the quantization codebook is predefined. Thus, how the codebook is designed has a dramatic impact on the performance of the quantized network. For example, a small codebook means that we can only search the parameters within a limited space, which makes the optimization problem very hard. However, in adaptive quantization, the codebook is learned from the data, where vector quantization and probabilistic quantization are two possible methods to achieve adaptive codebook quantization. In vector quantization, in order to learn the codebook, we must divide a large set of values into buckets, where each bucket is represented by its centroid point. In probabilistic quantization, the codebook can be inferred from the posterior distributions of the weights.

5.3 Adversarial Samples Attack

5.3.1 Attack Overview

In the current big data era, more and more available healthcare data are invaluable resources that carry important insights for the various aspects of care delivery. Data analytics technologies, such as data mining and machine learning, are important tools for digging those in-sights out from the healthcare data. Thus, various attack models and algorithms have been used to target classifiers, where they aim to find a perturbation δ to be added to a (legitimate) input x, resulting in the adversarial example $x' = x + \delta$. The perturbation δ is chosen to be small enough so as to remain undetectable.

In our work, we present a recent attack mentioned in [143]. In their adversarial samples generation process, their designed algorithm add perturbations to each medical record in

each iteration. Then, they check if this adversarial sample can fool the classifier. If so, the perturbation process stops. Otherwise, new perturbation will be added in the next iteration and the process repeats until it exceeds a threshold. One can compute the perturbation score for every successfully generated adversarial sample with its original sample.

5.3.2 Detailed Adversarial Samples Attack

Formally, given a well performed predictive model and a patient record x, their goal is to find an adversarial medical record x' of the same size, such that x' is close to x but with a different classification result from the given deep model. Let y be a source label currently outputted by the predictive model, and the authors would like the predictive model to classify x' into an adversarial class label $y' \neq y$, while minimizing the difference $x - x'$. As in the mortality prediction example, if a patient is originally predicted to be alive, they would like to find the minimal sparse perturbation $x - x'$ to make their model predict the deceased label.

Inspired by the C & W attack [40], the authors adopt a similar loss function $f()$ for crafting adversarial examples [143]. Specifically, given an input x and its correct label denoted by y, let x' denote the adversarial example of x with a target class $y' \neq y$. The loss function for targeted attack is defined as

$$f(x', y') = min\ max\{[Logit(x')]_y - [Logit(x')]_{y'}, -\kappa\} + \lambda||x' - x||_1, \tag{5.5}$$

where $Logit()$ is the logit layer (the layer prior to the softmax layer) representation of x and $\kappa \geq 0$ is a confidence parameter that controls the separation between y' and the next most likely prediction among all classes other than y'. Consequently, the loss function in the Equation 5.5 aims to render an adversarial example x' that will be classified as the target class y' while minimizing the distortion in the perturbation of x' relative to x.

In addition, the authors also use the iterative shrinkage-thresholding algorithm (ISTA) [27], which can be viewed as an additional shrinkage-thresholding step on each iteration. Thus, for each record value, the algorithm shrinks a perturbation to 0 if the deviance to original record is less than the soft threshold at each iteration. For example, the x' is

considered as a successful adversarial example of x if the model predicts its most likely class to be the target class y', where the perturbation is within the threshold. After each iteration, they end up with a set of adversarial candidates. Then, they define a Perturbation Distance Score (PDS) [143] to make perturbations comparable, which measures the quality of a perturbation:

$$D_{\triangle x} = \sqrt{max(|\triangle x|)^2 + \beta \cdot \frac{||\triangle x||_0}{d \cdot t}}, \tag{5.6}$$

where $\triangle x = x' - x$, d is the number of medical features in the EHR system and t is the available elapse time slots for the patients.

5.4 Weights Adjustment Attack

5.4.1 Attack Overview

With the ever-increasing system scale and complexity of ML models, modularization of ML systems becomes popular. Many pre-trained PLMs are hosted on development platforms such as Github [17], where developers can simply integrate them to create their own ML systems. For example, the developers may download deep learning based PLMs from a third part to improve the performance of their traditional learning models. However, such PLMs are seldom checked for potential security risks and may be the modified versions uploaded by attackers. Thus, after obtaining parameters of a pre-trained model from a third party, the developers will be affected if they use the attacked model directly without fine-tuning the model parameters over their own data.

The main goal of the attack is to force the RNN system to misclassify a group of inputs $\{\overrightarrow{X_1}, \overrightarrow{X_2}, \dots\}$ into a wrong output class (Y^*). For example, the inputs can be the lab tests of a Parkinson patient while the output can be a non-PD diagnosis.

A RNN-based ML system essentially models a composition function $f \times g : X \to Y$ with g and f being the feature extractor and the classifier. The adversary creates an adversarial PLM $\hat{\mathbf{g}}$ such that $f \times \hat{\mathbf{g}}$ classifies an input x_* from a certain class y_+ into a different class y_*. To avoid detection, $\hat{\mathbf{g}}$ should be almost indistinguishable from its genuine counterpart g. Specifically, g and $\hat{\mathbf{g}}$ should have proximate syntactic representation and they should behave

the same with respect to inputs from other non-targeted classes, i.e. $f \times g(x) \approx f \times \hat{\mathbf{g}}(x)$ for $x \notin C(y_+)$ where C(y) denotes inputs that belong to class y.

To make the attacks more practical, we assume that the attacker has full knowledge of the victim model (e.g., its structure, parameters, training procedure) and strive to make his malicious substitution almost indistinguishable from the genuine version so as to prevent from possible detection.

Algorithm 2 Weights Adjustment Attack Algorithm

Input : *data instances* $(\overrightarrow{X^*}, Y) \in D$ (*training set*), *feature extractor PLM function* $g : X \to V$, $C(Y_+)$ *denotes training instances belonging to victim class* Y_+;
R *are other training instances*; *genuine RNN module*; *learning rate* η; *thresholds* Θ, ϵ
Output : *attacked RNN PLMmodule*, $\hat{g}$
while $f(\overrightarrow{X^*}) \neq Y^*$ **do**
compute $\varepsilon \sim N(0, \sigma)$
▷ comment: compute additive stochastic noise, which is chosen from a standard normal distribution
$W \leftarrow$ *recurrent weights*
$r^+ = \Theta$ % *of* $\{|\sum_{(\overrightarrow{X^*}, Y_+) \in C(Y_+)} \triangle w_{(\overrightarrow{X^*}, Y^*)}|\}_{w \in W}$
$r^- = (100 - \Theta)$ % *of* $\{\sum_{(\overrightarrow{X}, Y) \in R} | \triangle w_{(\overrightarrow{X}, Y)}|\}_{w \in W}$
for $w \in W$ **do**
if $|\sum_{(\overrightarrow{X^*}, Y_+) \in C(Y_+)} \triangle w_{(\overrightarrow{X^*}, Y^*)}| > r^+$ or $\sum_{(\overrightarrow{X}, Y) \in R} | \triangle w_{(\overrightarrow{X}, Y)}| < r^-$ **then**
update $w^* \leftarrow w - (\eta * \varepsilon)$
end if
end for
if *no more weights need to be updated* **then**
break;
end if
end while

Our main idea is to add random noise on the learning weights in a RNN system such that it will generate wrong classification labels for a particular victim class. By adjusting the weights of the important features, there is a high chance that some unnecessary features will be treated as important in the feature selection step, which ruins the robustness of learning-based classifiers. Our attack works as follows:

(1) Step1: We first identify a subset of features that are important for inferring a certain class. Let us assume each patient's data consists of a sequence $(x_1, x_2, \ldots, x_t)$, where x_t is a vector of length k (representing k features) collected at time step t. Based on the input data, we can compute the importance vector, $f(x_i)$ by using the following equation.

$$f(x_i) = (\sigma(\sum(W_{1i} \odot x_i) + b_{1i}), \ldots, \sigma(\sum(W_{ki} \odot x_i) + b_{ki})) \tag{5.7}$$

Where σ is the logistic sigmoid function, $W_{1i}, \ldots, W_{ki}$ are weight matrices, $b_{1i}, \ldots, b_{ki}$ are biases, $\odot$ is an element-wise multiplication, and $\sum$ is an operator that sums all elements of a matrix. Each element, f_i within $f(x_i)$ represents how important feature i is in determining the label for this class.

(2) Step2: Based on the scores, we can identify a subset of important features. Since there is a set of weights that correspond to the selected features, so we modify those weights by adding stochastic noise with the aim of causing the classifier to output the wrong class label for this victim class.

5.4.2 Detailed Weights Adjustment Attack

We first define the importance of a weight w with respect to a given input $(\overrightarrow{X}, Y)$ as follows: where σ_Y is the probability that the input $\overrightarrow{X}$ belongs to the class Y.

$$\triangle w_{(\overrightarrow{X},Y)} = \frac{\partial \sigma_Y}{\partial w} - \sum_{Y' \neq Y} \frac{\partial \sigma_{Y'}}{\partial w}$$

Then, for a given layer of the feature extractor, we calculate the positive impact and negative impact when adjusting the weight for each feature. Let Y_+ be the victim class and $C(Y_+)$ be the training instances belong to class Y_+.

- Positive Impact: $|\sum_{(\overrightarrow{X^*},Y_+) \in C(Y_+)} \triangle w_{(\overrightarrow{X^*},Y^*)}|$ measures the influence of w with respect to classify inputs $\overrightarrow{X^*} = \{\overrightarrow{X_1^*}, \overrightarrow{X_2^*}, \ldots\}$ as Y^*, where Y^* is a targeted class.
- Negative Impact: $\sum_{(\overrightarrow{X},Y) \in R} | \triangle w_{(\overrightarrow{X},Y)}|$ quantifies w's influence on the classification task where R is the set including all instances that do not belong to Y_+.

Since different clustering tasks may have different important features, so we enforce that the adjustment of the target recurrent weights should have high positive impacts but low negative impacts.

Next, we adjust the selected weights (i.e. weights with positive impacts higher than Θ% of all the weights but their negative impacts are below the $(100 - \Theta)^{th}$ percentile) by

reducing their values by some stochastic noises. This process repeats until no more qualified weights could be found or no more weights need to be further modified.

Our detailed attack algorithm is described in Algorithm 1.

5.5 Detection & Defense Mechanisms

In this section, we propose low-cost detection mechanisms to identify the presence of these two types of attacks and also design a defense method to make the models more robust to both types of attacks.

5.5.1 Detection Scheme for Adversarial Samples Attack

In this subsection, we present a modified detection scheme for identifying adversarial samples attack. First, we summarize the mutation-testing based detection scheme proposed in [155] for identifying adversarial samples against image classifiers. Then, we present how we modify their scheme to detect adversarial samples attack against RNN-based models. The main idea of the mutation-testing based detection scheme is based on the observations that adversarial samples are much more sensitive to perturbations than normal samples. If they impose random perturbations on a normal and an adversarial samples respectively, there would be a significant difference between the ratio of labels change due to the perturbations. Thus, in their work the authors design an algorithm to determine if a given input is a normal sample or an adversarial one through mutation testing (as shown in Fig 5.2).

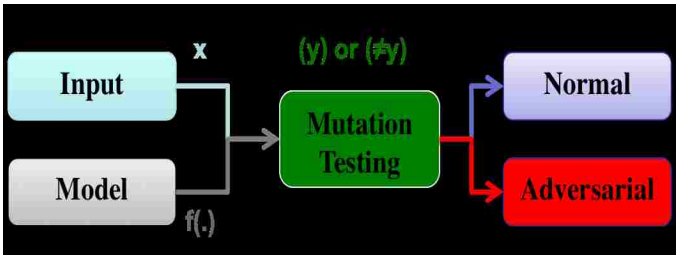


Figure 5.2: Detection Mechanism for Adversarial Samples Attack

Instead of identifying adversarial samples against image classifiers, in our work we have modified their scheme to detect adversarial samples attack against RNN-based models, where can be described as follows:

(1) Given a set of training data, the detection algorithm generates a set of K possible mutations on every normal sample x, denoted as $X_m(x) = \{x_{m1}, x_{m2}, \ldots, x_{mK}\}$, where each element x_{mi} is obtained by imposing minor perturbations on the important features of x.

(2) After that, for every mutation $x_{mi} \in X_m(x)$, the detection scheme obtains its output label $f(x_{mi})$ by feeding it into the learning model $f(.)$. For most of the mutations in $X_m(x)$, $f(x_{mi})$ should be $f(x)$ because it is imposing small permutations, however, a label changing may still occur. Thus, the sensitivity of a sample x to mutations can be computed as follows:

$$St_x = \frac{|\{x_{mi}|x_{mi} \in X_m(x) \bigwedge f(x_{mi}) \neq f(x)\}|}{|X_m(x)|}, \tag{5.8}$$

where $|X_m(x)|$ is the number of elements in a set X_m. Intuitively, St_x is the percentage of mutations in $X_m(x)$ that have a different label $f(x_{mi})$ from $f(x)$. Then, the detection scheme computes the average sensitive value for all the normal inputs and set it as St_{nor}.

(3) Next, for a new given input x_{new} during operation, the detection scheme conducts similar mutations on this input and compute its sensitive value denoted as $St_{x_{new}}$.

(4) Finally, the detection scheme measures a normalized distance (ND) between St_{nor} and $St_{x_{new}}$ (in Equation 5.9) to check if this input x_{new} is an adversarial sample or not.

$$ND = \frac{St_{x_{new}} - St_{nor}}{St_{nor}} > \mu - 1, \tag{5.9}$$

where μ is used to control the identification errors. Thus, if the normalized distance (ND) is larger than (μ-1), then this input is determined to be an adversarial sample, where we would raise an alarm and avoid making wrong decisions that could lead to severe consequences.

5.5.2 Detection Scheme for Weights Adjustment Attack

In this subsection, we propose a simple detection solution that can infer if any downloaded PLM has been modified, where we assume that the users have access to the training data. Our main idea is based on the following observations: (i) While deep learning models could provide better performance than the traditional learning models in predicting disease

diagnosis or progression, there is a high probability that the important features they have extracted are similar. (ii) The important features of a deep learning model before and after attacks will be different.

Thus, we can compare the set of important features extracted using the downloaded PLM with those that are extracted using traditional models (assuming that such features have already been identified by health service providers who are currently using traditional methods but interested in using a deep learning based PLM) to see how different they are. if they are significantly different, then we know there is an attack.

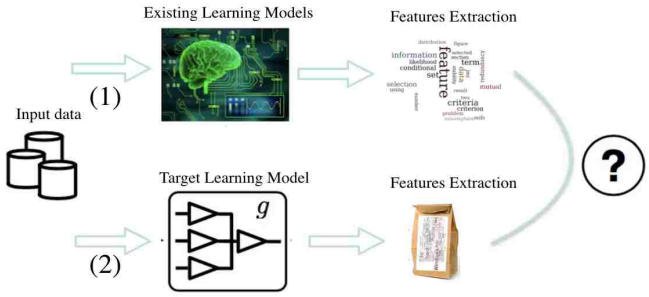


Figure 5.3: Our Detection Mechanism for Weights Adjustment Attack

Our detection scheme works as follows:

(1) After obtaining the input data, our detection algorithm will compare the important m features extracted by a downloaded RNN model with those extracted using a traditional learning model (as shown in Fig 5.3). We assume that the publisher of the RNN healthcare learning model also publishes an appropriate m value where the accuracy of an unattacked RNN model is within $Tr_1\%$ if one only uses the top m features versus the whole feature sets.

(2) If the difference of extracted features of the two models is larger than a threshold Tr_2, then we declare that the downloaded RNN model is suspicious and should not be used.

Next, we describe in more details about how our detection scheme works by using a RNN-based learning model in healthcare domain as an example.

(1) Let us assume that the input dataset D contains two types of data: (a) Static Data: (i) long term data (personal profile, e.g., gender), (ii) short term data (lab test results); (b) Elapsed Time: time intervals between two visits of a patient.

We assume that this dataset contains M patients and each patient has N static data

(each has V features) and N elapse times. Thus, we first split the input Data D into N subsets, denoted as $\{D_1, \ldots, D_N\}$.

(2) For every subset D_i, it contains patients' static data and the correlated elapse time.

- D_1: $\{\{patient_1, static-data_1\}, \ldots, \{patient_M, static-data_1\}, T_1\}$
- D_2: $\{\{patient_1, static-data_2\}, \ldots, \{patient_M, static-data_2\}, T_2\}, \ldots$
- D_N: $\{\{patient_1, static-data_N\}, \ldots, \{patient_M, static-data_N\}, T_N\}$

(3) For every subset D_i, we use existing feature selection methods (e.g., Logistic Regression (LR), Decision Tree (DT), Random Forest (RF)) to extract important features and store as $FD_i = \{FD_{i1}, FD_{i2}, \ldots\}$.

(4) Based on the elapse time T_i of each dataset D_i, we compute a score as $\sum_{i=1}^{N} \frac{S_{ij}}{T_i}$ for each feature V_j in $\{FD_1, FD_2, \ldots, FD_N\}$, where if feature V_j is shown in FD_i, then S_{ij}=1, otherwise S_{ij}=0.

(5) We select top m features (with highest scores) and store them as $F_1 = \{V_{11}, \ldots, V_{1m}\}$, where m is a value which the publisher of that downloaded PLM specific as discussed earlier.

(6) Next, using the same input dataset D, we extract important features (top m) using the downloaded RNN model, and store them as $F_2 = \{V_{21}, V_{22}, \ldots, V_{2m}\}$.

(7) We compare both F_1 and F_2 to check if such target model has already been attacked. Specifically, we check if $\frac{|F_1 \cap F_2|}{|F_1|} * 100 > Tr_2$. If it does, then we declare that the model has been attacked.

In addition, if the users do not have access to the training data, then, we can increase the diversity of a user's test set by creating some synthetic test instances based on the characteristics of the existing test sets. Next, we can compare the distribution of the predicted results of the downloaded learning model with existing learning models. Since the attack tries to mislead the predicted results of a particular victim class, the distribution of the predicted outcomes between the downloaded learning model and existing learning models will be different (i.e., less correct predictions for the victim class).

5.5.3 Quantization-based Defense Solution

(1) Defense Overview

Our SDQDS utilizes static/dynamic quantization schemes to produce quantized deep learning models which are more robust against these two types of attacks. Our solution consists of three parts: (i) feature weights partition, (ii) layer group-wise quantization and (iii) retraining. The high-level idea of our weight quantization approach is to first group features into different clusters. Then, we use the centroid of each cluster to generate representative values and quantize the actual weights in each cluster into the corresponding representative values.

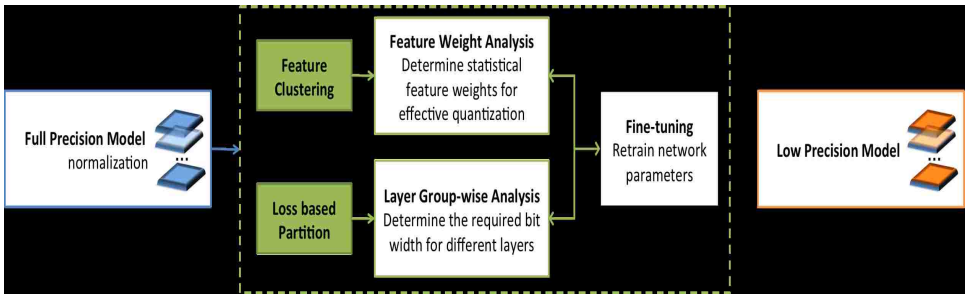


Figure 5.4: SDQDS Overview

(2) Detailed Defense

In this part, we will describe the implementation of our defense mechanism in detail, which is shown in Fig 5.4.

(A) Feature Weights Partition:

We adopt k-means clustering to identify the shared weights for each layer of a trained network and quantize the weights into different values based on the centroids values of different clusters. For example, we can partition n network parameters (e.g., weights) $W = \{w_1, w_2, \ldots, w_n\}$ into k disjoint clusters $C = \{C_1, C_2, \ldots, C_k\}$ (C_i is the centroid). After clustering, each weight is assigned to a cluster index:

$$\underset{C_1,C_2,\ldots,C_k}{\arg\min} \sum_{i=1}^{k} \sum_{w \in C_i} |w - c_i|^2, where\ c_i = \frac{1}{C_i} \sum_{w \in C_i} w, \tag{5.10}$$

For simplicity, we firstly consider the problem with $k = 2$. Suppose that α_1 and α_2

are known in advance with $\alpha_1 \geq \alpha_2 \geq 0$, then the quantization codes are restricted to $v = \{-\alpha_1-\alpha_2, -\alpha_1+\alpha_2, \alpha_1-\alpha_2, \alpha_1+\alpha_2\}$. Consequently, based on the distance we can partition the weights into 4 intervals. And each interval corresponds to one particular quantization level (shown in Fig 5.5). For the general k-bit quantization, suppose that $\{\alpha_i\}_{i=1}^k$ are known and we have all possible codes in ascending order, i.e., $v = \{-\sum_{i=1}^k \alpha_i, \ldots, \sum_{i=1}^k \alpha_i\}$. Similarly, we can partition the weights into 2^k intervals, in which the boundaries are determined by the centers of two adjacent quantization levels. For example, if $w < (v_{m/2} + v_{m/2+1})/2$, its feasible quantization level is then optimally restricted to $v_{1:m/2}$, otherwise its feasible quantization level becomes $v_{m/2+1:m}$. By recursively partition the ordered quantization levels, we can then efficiently determine the optimal quantization level for each weight by only k comparisons.

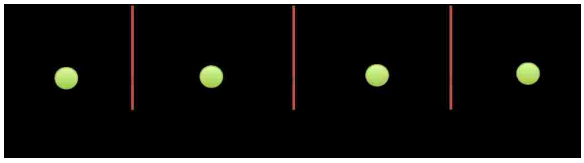


Figure 5.5: An Example of Quantization Levels Partition

(B) Layer Group-wise Quantization:

Recent neural networks are getting deeper [145, 165], where different layers will have different impacts on the final predictions. For example, some layers such as convolutional layers may be more important than the others. Thus, in our work instead of quantizing network parameters of all layers together we also consider to conduct different quantizations on all individual layers (shown in Fig 5.6).

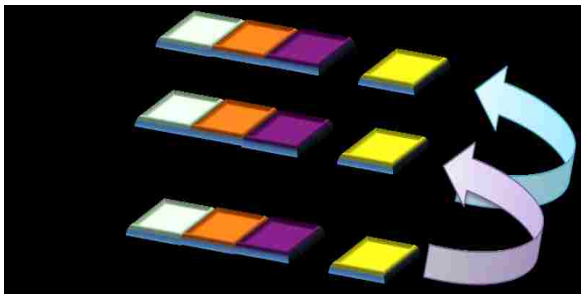


Figure 5.6: An Example of Layer Group-wise Quantization: (i) the purple and blue arrows mean we conduct quantizations on all individual layers; (ii) on each layer we use different colors to represent different quantized levels on the feature weights

(C) Re-training After Quantization:

As quantized weights bring errors to the network, we need to retrain the model to recover accuracy, where we fix the quantized weights and iteratively use our approach to quantize the remaining weights until all the weights are quantized.

5.6 Performance Evaluation

In this section, we conduct experiments on synthetic and real-world healthcare datasets to answer the following questions: (i) if the performance of a RNN-based ML system will decrease when two described attacks have been launched; (ii) if our detection mechanism is effective in identifying malicious model, and (iii) if our defense method is useful in preventing such attacks. In the remainder of this section, we will describe the RNN-based model, the datasets we use and our empirical results.

5.6.1 Experimental Settings

To evaluate the performance of our schemes, we conduct experimental evaluations on four public medical datasets, namely, one synthetic diabetes datasets (large), two parkinson datasets (small & large) and MIMIC-III dataset as shown in Table 5.1. We evaluate the attacks, detection and defense mechanisms on a recently published RNN model: T-LSTM [26].

(1) A RNN Model Description

A LSTM architecture (T-LSTM) proposed in [26] is used. This LSTM-based model is designed to handle time irregularities in sequences. Their system contains two parts: (a) a T-LSTM auto encoder, which is used to generate a representation for each patient based on his/her medical records and elapse time intervals. Such T-LSTM auto-encoder has a T-LSTM encoder and a T-LSTM decoder unit with different parameters, which are jointly learned to minimize the reconstruction error. The proposed auto-encoder can capture the long and the short term dependencies by incorporating the elapsed time into the system, where the short-term memory is adjusted by the elapsed time weight to obtain the discounted short-term memory; (b) The output from the hidden layer of the T-LSTM encoder

Table 5.1: Datasets Description

Dataset	Data Size	# of Features	# of Classes	# of Elapse Time
SD	6730	36	2	4
PD	40	26	2	4
PPMI	643	306	2	25
PPMI	643	306	6	25
MIMIC-III	37559	19	2	48
MIMIC-III	37559	19	8	48

at the end of the sequence is used as a single representation of a patient for the subsequent patient clustering task. In addition, the T-LSTM was implemented in Tensorflow [20], an open source software library for numerical computation using data flow graphs, and mini-batch stochastic Adam optimizer.

(2) Datasets Description

We use four datasets in our experiments, which are described as follows:

- Synthetic Diabetes Dataset (large) (SD) [26]: In this dataset, there are 6730 patients with an average of 4 admissions each. Feature of an admission of a patient is a vector containing the diagnoses given in the corresponding admission and its feature size is 36. Even though the dataset is artificially generated, it contains similar characteristics as a real EHR data. Patients are categorized either into the no-diabetes or with diabetes classes.

- Parkinson Dataset (small) (PD) [106]: This dataset is composed of a range of biomedical voice measurements from 40 people, where 20 have Parkinson's disease (PD) and the rest are healthy people. Each column in the dataset is a particular voice measure, and each row corresponds to one of the 195 voice recordings from these individuals. The main aim of the data is to discriminate healthy people from those with PD, according to "status" column which is set to 0 for healthy and 1 for PD.

- PPMI Dataset (large) [45]: This is the Parkinson's Progression Markers Initiative (PPMI) challenge dataset, which is provided to let clinical researchers identify biomarkers for predicting the progression of Parkinson's disease. We pre-process this large

dataset and retains only features that are observed in at least 400 patients' records. Subsequently, we only keep 643 patients whose primary diagnosis are either "Idiopathic PD" or "Non PD".

We also select a total of 306 raw features, which consist of 7 categories: (a) motor symptoms/complications (MCs) (SPES/SCOPA sections Motor/Complications), (b) cognitive functioning (SCOPACOG), (c) autonomic symptoms (SCOPA-AUT), (d) psychotic symptoms (SCOPA-PC, items 15), (e) nighttime sleep problems and excessive daytime sleepiness (SCOPASLEEP), (f) depressive symptoms [PROPARK: Beck Depression Inventory (BDI); (g) ELEP: Hospital Anxiety and Depression Scale (HADS).

We used the last three months health records for prediction. Prediction categories is either 2 classes (PD and non-PD) or 6 classes based on the Hoehn and Yahr (NHY) scale scores [77]. The NHY scale is a commonly used system describing how the motor functions of PD patients deteriorate. The scores are ordered and discrete, ranging from 0 to 5. Score 1.0 means that the PD is limited to one side of the body. Other motor conditions such as tremor, rigidity, reduced arm swing, and slowness are present only on one side. Score 2.0 refers to problems affecting both sides. The higher the score, the more severe the condition is. Patients are classified into 6 classes based on these scores.

- MIMIC-III Dataset (large) (MC3) [82]: This dataset was made available by Neh Israel hospital and data from their critical care unit from 2005 to 2011. It contains information about patient demographics, hourly vital sign measurements, laboratory test results, procedures, medications, ICD-9 codes, care-giver notes and imaging reports. In total, there were 58,976 unique admissions and 14,567 unique International Statistical Classification of Diseases and Related Health Problems (ICD)-9 diagnostic codes. Our experiment uses records from a collection of patients, which consist of 19 variables from 2 categories: vital sign measurements with 6 features and lab events with 13 features. Vital signs include heart rate (HR), systolic blood pressure (SBP), diastolic blood pressure (DBP), temperature (TEMP), respiratory rate (RR), and

oxygen satu- ration (SPO2). Lab measurements include: Lactate, partial pressure of carbon dioxide (PaCO2), PH, Albumin (Alb), HCO3, calcium (Ca), creatinine (Cre), glucose (Glc), magnesium (Mg), potassium (K), sodium (Na), blood urea nitrogen (BUN), and Platelet count.

Since MIMIC-III contains numerous missing values and outliers, so we use the processed data mentioned in [143, 148], which contains 37,559 multivariate time series with 19 variables across 48 time stamps. As to the class predictions, we conduct the following two tasks: (i) In-hospital Mortality task is modeled as a binary classification problem. (ii) Length of ICU Stays (LOS) task prediction remains an important task for identifying high-cost hospital admissions in terms of staffing cost and resource management. We formulated such task as a multiclass classification problem using bins of lengths (1, 2), (3, 5), (5, 8), (8, 14), (14, 21), (21, 30), (30+,) to reflect the range of possible LOS values in terms of days.

(3) Experiments Overview

Our experiments consist of three parts: (i) before launching attacks, we first measure the performance of the unattacked T-LSTM model, where the top m important features will be selected and compared in our detection scheme; (ii) then, we launch two types of attacks as we explained in section 5.3 & 5.4, where the goal is to evaluate their impacts on the target model; (iii) Next, we evaluate our detection and defense mechanisms.

In our experiments, we train the model for 200 epochs and apply a learning rate of 0.005. Since the model with more LSTM layers becomes more complex which will in turn affect the prediction results, thus we choose an appropriate number of layers based on the datasets we use: (i) we select 2 LSTM layers for the small datasets (PPMI); (ii) 3 LSTM layers for the large dataset (MIMIC-III). We also choose 512 hidden units for each layer. All the experiments are performed on Ubuntu 14.04 with 4.0GHz, Intel Core i7 CPU, 16GB Memory and Nvidia GTX 1080 Graphics Card.

Table 5.2: Unattacked T-LSTM Model Evaluation

Dataset	# of Features	# of Classes	T-LSTM	top3 features	top5 features	top10 features	top15 features	top20 features
SD	36	2	78.1%	72.6%	75.8%			
PD	26	2	77.5%	70%	75%			
PPMI	306	2	60.9%	37.5%	42.2%	48.4%	54.7%	57.9%
PPMI	306	6	43.8%	25%	28.1%	32.8%	35.9%	40.9%
MIMIC-III	19	2	87.3%	81.7%	84.6%			
MIMIC-III	19	8	85.4%	79.1%	81.9%			

5.6.2 Evaluation of Unattacked T-LSTM Model

Before launching the attacks, we conducted Exp1 to measure the performance of the unattacked T-LSTM model, where we measure the performance by varying the number of input features.

From the result (Table 5.2), we can see that in order to ensure the accuracy of the unattacked model varies within (Tr_1=3%) when only using the top m features, the m values we need to set are as follows:

- we select m = 5 for SD, PD and MIMIC-III datasets because those datasets have small number of features.
- we select m = 20 for the PPMI since such dataset has large number of features.

Table 5.3: Adversarial Samples Attack Evaluation for T-LSTM

Dataset	# of Features	# of Classes	Accuracy of D1 (Without Attack)	Attack Success Rate of D2
PPMI	306	6	43.8%	92%
MIMIC-III	19	2	87.3%	84%
MIMIC-III	19	8	85.4%	80%

5.6.3 Effectiveness of Two Types of Attacks

(1) Adversarial Samples Attack Evaluation

We have conducted Exp2 to measure the effectiveness of the adversarial samples attack via using different datasets, namely (i) a small dataset (PPMI); and (ii) a large dataset (MIMIC-III).

Exp2: in this experiment, we first evaluate the prediction accuracy of the T-LSTM model for the benign PPMI data with 6 classes (denoted as D1). Then, we launch the adversarial samples attack by generating 200 adversarial samples (D2) that can force such model to make wrong predictions for the target class: class1 for PPMI, and evaluate the success rate of such attack. Next, we repeat the same process for the MIMIC-III dataset, namely (i) MIMIC-III with 2 prediction classes (class0 is the target class); (ii) MIMIC-III with 8 prediction classes, where class4 is the target class.

The experimental results are shown in Table 5.3. From the results, we can see that the model can be easily affected by the adversarial samples attack since most of the adversarial samples will force the model to make wrong predictions. For example, the attack success rate of D2 is high for both datasets (92% for PPMI and 82% for MIMIC-III on the average).

Table 5.4: Weights Adjustment Attack Evaluation for T-LSTM

Dataset	# of Features	# of Classes	Accuracy (Without Attack)	Accuracy (Attack on Important Features with $\Theta = 70$)	Accuracy (Attack on Important Features with $\Theta = 80$)
SD	36	2	78.1%	64.2%	66.9%
PD	26	2	77.5%	65%	67.5%
PPMI	306	2	60.9%	48.4%	51.6%
PPMI	306	6	43.8%	34.4%	35.9%
MIMIC-III	19	8	85.4%	76.5%	77.8%

(2) Weights Adjustment Attack Evaluation

We have conducted Exp3 to measure the effectiveness of the weights adjustment attack, where we use both synthetic and real clinical data.

Exp3: in this experiment, we first evaluate the prediction accuracy of the T-LSTM model using different datasets, namely (i) SD dataset with 2 classes; (ii) PD with 2 prediction classes; (iii) PPMI with either 2 or 6 prediction classes; (iv) MIMIC-III with 8 prediction classes. Then, we launch our attack by adjusting the training weights in the T-LSTM network structure to force the model to make wrong prediction for the target class: (i) class1 for SD; (ii) class1 for PD; (iii) class1 for PPMI; (iv) class4 for MIMIC-III. In addition, we also vary the number of attacked features by setting Θ to either 70 or 80 (e.g., Θ=70 means 30% of the feature weights will be attacked) so that we can understand the attack's

impacts on the model's predictions.

The experimental results are shown in Table 5.4. From the results, we can see that adjusting weights of the features identified using $\Theta = 70\%$ will largely decrease the clustering accuracy by 11% on the average. Such reduction can be explained as follows: changing the weights of the important features weakens the significance of those features, which in turn causes the classifier to output wrong class labels. However, the accuracy will improve by 2.5% if we set $\Theta = 80\%$. It is because with $\Theta = 80\%$ the attack adjusts fewer weights of the model when comparing with $\Theta = 70\%$.

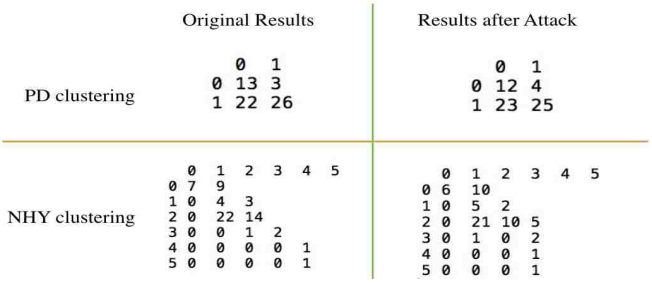


Figure 5.7: Confusion Matrix for Anxiety Disorders of T-LSTM

In addition, we find that adjusting the weights for the same features has different impact on different clustering tasks. For example, in the PPMI dataset, if we adjust weights for the anxiety factors in the T-LSTM system, then the accuracy of PD and NHY clustering will reduce by 3% and 8% respectively (confusion matrix is shown in Fig 5.7). This can be explained as follows: the Parkinson disease cannot be easily diagnosed based on the anxiety factors. However, the anxiety disorders can be used to determine the severity of PD patients. Thus, they have significant impacts on the NHY scale, which is commonly used to describe how the motor functions of PD patients deteriorate.

5.6.4 Detection & Defense Solutions Evaluations

In this subsection, we conducted three experiments (Exp4 & Exp5 & Exp6), where (i) Exp4 is used to evaluate the effectiveness of the detection mechanism for the adversarial samples attack, (ii) Exp5 is used to evaluate the effectiveness of the detection mechanism for the weights adjustment attack, and (iii) Exp6 is conducted to measure our defense mechanism.

(1) Evaluations of Adversarial Samples Attack Detection Scheme

We have conducted Exp4 to measure the effectiveness of the detection scheme for the adversarial samples attack, where we use MIMIC-III dataset with 8 class labels.

Exp4: in this experiment, we conduct two types of mutations: (i) first type of mutations operate on all features, (ii) 2^{nd} type of mutations operate only on important features. Then, for each type of mutations we conduct K ($K = 200, 1000$) mutations on every sample in the training set and compute the average sensitive values denoted as St_{nor1} (Type 1) and St_{nor2} (Type 2). Next, we launch the adversarial samples attack to generate 200 adversarial samples that can force the victim model to make wrong predictions for the target class (class4 for MIMIC-III). We also repeat the process to generate two types of mutations for each adversarial sample and compute the sensitive values denoted as St_{adv1} (Type 1)and St_{adv2} (Type 2). In addition, we select the threshold μ to control the identification errors, which is computed based on the average sensitive value (St_{nor}) and the standard deviation (std_{nor}) of the normal training samples, denoted as $\mu = \frac{(St_{nor}+\lambda * std_{nor})}{St_{nor}}$, where we set (i) $\lambda = 1$, (ii) $\lambda = 1.5$ and (iii) $\lambda = 2$ in the experiment. Finally, we compute the normalized distances (mentioned in section 5.5.1) of (i) St_{adv1} and St_{nor1}, (ii) St_{adv2} and St_{nor2} respectively to determine if this input is an adversarial sample.

Table 5.5: Detection Method Evaluation for Adversarial Samples Attack for T-LSTM

Dataset	# of Classes	Detection Method	# of Mutations	Identification Rate ($\lambda = 1$)	Identification Rate ($\lambda = 1.5$)	Identification Rate ($\lambda = 2$)
MIMIC-III	8	all features	200	43%	42.5%	42%
MIMIC-III	8	important features	200	53%	53%	51.5%
MIMIC-III	8	all features	1000	52%	51.5%	49.5%
MIMIC-III	8	important features	1000	60.5%	60.5%	59.5%

The results are shown in Table 5.5. From the results, we can see that such detection method works for the adversarial samples attack, where it can achieve an average detection rate of 57%. We also find that varying the number of mutations and μ value will affect the

final results, where the performance will be improved if we set $\lambda = 1$ or $\lambda = 1.5$ with 1000 mutations. Thus, in order to detect a strong attack (i.e., distance between St_{adv} and St_{nor} is small), we would need to select an optimal μ value, and evaluate more mutations to reach a conclusion.

In addition, instead of only selecting one μ value for all classes, we also conduct sensitivity analysis to generate the μ value for each class, which can be used as the threshold to identify the adversarial samples. For example, for MIMIC-III dataset by using a μ value of class3 or class4 in the final comparison process, the performance of this detection method can be improved by 4% or 9% on average respectively. Moreover, we find that conduct mutations on important features can have better performance than on all features. It is because the sensitive values of normal samples (St_{nor}) are relatively stable for a given model but the values of adversarial samples (St_{adv}) are more sensitive to mutations on the important features. Thus, adding mutations on those features would increase the distance between St_{adv} and St_{nor}, which makes it easier for the algorithm to detect adversarial samples.

(2) Evaluations of Weights Adjustment Attack Detection Scheme

Based on the input data size, we use two traditional feature selection techniques to assess which features are more helpful in constructing the learning models.

- Logistic Regression (LR) [63]: this uses maximum-likelihood estimation to compute the coefficients fro all features, which can be used to rank them based on their relative importance.

- Random Forest (RF) [97]: this is an ensemble classifier based on randomized decision trees and provides different feature important measures, which can be visualized by the Gini index scores [137, 71]. This feature importance score provides a relative ranking of the spectral features and can be used as a general indicator of feature relevance. Here, we run random forest classifier on all features and select useful features based on their Gini index scores.

From our previous data mining work [128], we find that logistic regression (LR) has good performance in identifying important features when the feature size is small (fewer

Table 5.6: Detection Evaluation for Weights Adjustment Attack using Logistic Regression

Dataset	# of Features	# of Classes	top5 features (LR)	top5 features (Attacked T-LSTM)
SD	36	2	32, 10, 21, 7, 33	5, 17, 18, 22, 33
PD	26	2	11, 13, 20, 21, 27	3, 8, 10, 19, 21
MIMIC-III	19	8	2, 4, 8, 11, 15	1, 6, 10, 15, 17

than 100). But for a large feature size, it is better to use random forest (RF) feature extractor. Thus, we conduct Exp5 where (i) we use SD, PD, MIMIC-III and use logistic regression (LR) to extract important features, and (ii) we use PPMI dataset which is larger and hence we use the random forest (RF) method to extract important features.

Exp5: in this experiment, we use LR to select important features and stored the extracted top m features as F_1 from the input data. We also use the downloaded RNN learning model to extract top m important features (F_2). Then, we compare the extract features F_1 and F_2 to check if the downloaded learning model has been attacked. Next, we use the same method for the PPMI dataset and use RF to extract important features.

Table 5.7: Detection Evaluation for Weights Adjustment Attack using Random Forest

Dataset	# of Features	# of Classes	top20 features (RF)	top20 features (Attacked T-LSTM)
PPMI	306	2	184, 143, 156, 88, 62, 187, 58, 52, 234, 150, 141, 163, 178, 192, 203, 153, 174, 198, 213, 221	143, 171, 154, 146, 62, 88, 159, 291, 152, 233, 112, 132, 183, 194, 205, 49, 28, 162, 240, 283
PPMI	306	6	184, 182, 62, 187, 297, 133, 145, 49, 65, 269, 88, 213, 154, 53, 150, 171, 176, 183, 192, 204	143, 182, 171, 139, 269, 29, 62, 224, 151, 140, 292, 105, 239, 150, 2, 9, 65, 112, 185, 205

The results of the top m features before and after an attack has been launched are shown in Table 5.6 and Table 5.7. By comparing the extracted features in Table 5.6 and Table 5.7, we find that the top m features extracted by existing traditional feature extractor are largely different from those extracted by the attacked RNN model, where the overlap of two feature sets (Tr_2) is less than 30%. For example, in the PPMI dataset, the top20 features extracted by RF for NHY clustering contain "Anxiety" and "Motor" factors. Whereas, the top20 features extracted by the attacked T-LSTM model for NHY clustering mainly contain "Sleep", "Cognitive" and "Depression" factors. Therefore, by selecting appropriate

threshold values (e.g., Tr_2), our detection method is effective in detecting the weights adjustment attack.

(3) SDQDS Evaluation

In order to illustrate the effectiveness of our defense mechanism, we conduct Exp6 on the PPMI (small) and MIMIC-III (large) datasets, where (i) we use static quantization on each LSTM layer, and (ii) we replace the static quantization with dynamically quantized levels for different feature weights on different LSTM layers.

Exp6: in this experiment, we first use static quantization on each LSTM layer, where (i) for PPMI dataset, we use a 16-bit static quantization on the 1^{st} LSTM layer and a 12-bit or 8-bit static quantization on the 2^{nd} LSTM layer; (ii) for MIMIC-III dataset, we use 16-bit quantization on the 1^{st} LSTM layer, 12-bit quantization on the 2^{nd} LSTM layer and 12-bit or 8-bit static quantization on the 3^{rd} layer. Then, instead of using static quantization, we also apply dynamic quantization in two ways (i) layer-based dynamic quantization, where we separate the features into 7 clusters and 2 clusters for PPMI and MIMIC-III dataset respectively; (ii) feature-based dynamic quantization. Finally, we measure the performance for both benign (D1) and adversarial data (D2) (discussed in Exp2), where we measure the prediction accuracy for D1 and attack success rate for D2.

Table 5.8a: Impact of SDQDS on Accuracy Without Attack (PPMI)

Defense Method	Dataset	# of Features	# of Classes	Accuracy (Without Defense)	Accuracy (16, 12)	Accuracy (16, 8)
Static	PPMI (D1)	306	6	43.8%	42.2%	39.1%
Dynamic (layers)	PPMI (D1)	306	6	43.8%	42.2%	40.6%
Dynamic (features)	PPMI (D1)	306	6	43.8%	42.2%	40.6%

Table 5.8b: Impact of SDQDS on Adversary Samples Attack (PPMI)

Defense Method	Dataset	# of Features	# of Classes	Attack Success Rate (Without Defense)	Attack Success Rate (16, 12)	Attack Success Rate (16, 8)
Static	PPMI (D2)	306	6	92%	62%	29%
Dynamic (layers)	PPMI (D2)	306	6	92%	56%	25%
Dynamic (features)	PPMI (D2)	306	6	92%	48%	20%

Table 5.9a: Impact of SDQDS on Accuracy Without Attack (MIMIC-III binary-class)

Defense Method	Dataset	# of Features	# of Classes	Accuracy (Without Defense)	Accuracy (16, 12, 12)	Accuracy (16, 12, 8)
Static	MIMIC-III (D1)	19	2	87.3%	83.2%	80.5%
Dynamic (layers)	MIMIC-III (D1)	19	2	87.3%	83.8%	81.7%
Dynamic (features)	MIMIC-III (D1)	19	2	87.3%	84.4%	82.6%

Table 5.9b: Impact of SDQDS on Adversary Samples Attack (MIMIC-III binary-class)

Defense Method	Dataset	# of Features	# of Classes	Attack Success Rate (Without Defense)	Attack Success Rate (16, 12, 12)	Attack Success Rate (16, 12, 8)
Static	MIMIC-III (D2)	19	2	84%	54%	42%
Dynamic (layers)	MIMIC-III (D2)	19	2	84%	45%	36%
Dynamic (features)	MIMIC-III (D2)	19	2	84%	41%	29%

The results are shown in Table 5.8a, Table 5.8b, Table 5.9a, Table 5.9b, Table 5.10a and Table 5.10b. From the results, we can see that while converting a deep network into a low precision network essentially adds quantization noise into the model, which degrades the performance of the benign inputs (D1), such defense solution works for the adversarial samples attack, where the success rate of the adversarial samples (D2) has largely decreased.

Table 5.10a: Impact of SDQDS on Accuracy Without Attack (MIMIC-III multi-class)

Defense Method	Dataset	# of Features	# of Classes	Accuracy (Without Defense)	Accuracy (16, 12, 12)	Accuracy (16, 12, 8)
Static	MIMIC-III (D1)	19	8	85.4%	80.2%	76.8%
Dynamic (layers)	MIMIC-III (D1)	19	8	85.4%	81.5%	78.9%
Dynamic (features)	MIMIC-III (D1)	19	8	85.4%	82.3%	79.6%

Meanwhile, by comparing the results in Table 5.8a, Table 5.8b, Table 5.9a, Table 5.9b, Table 5.10a and Table 5.10b, it seems that dynamic quantization performs better than the static quantization. It is because in the dynamic quantization, we choose the dynamic

Table 5.10b: Impact of SDQDS on Adversary Samples Attack (MIMIC-III multi-class)

Defense Method	Dataset	# of Features	# of Classes	Attack Success Rate (Without Defense)	Attack Success Rate (16, 12, 12)	Attack Success Rate (16, 12, 8)
Static	MIMIC-III (D2)	19	8	80%	56%	45%
Dynamic (layers)	MIMIC-III (D2)	19	8	80%	49%	38%
Dynamic (features)	MIMIC-III (D2)	19	8	80%	44%	32%

quantization levels based on the information contained in the trained network, where the shared weights for each layer are quantized based on the centroids values of different feature weights clusters.

In addition, since different LSTM layers have different impacts on the final predictions [118], so in the dynamic quantization we conduct different quantizations on all individual layers. For example, since the last layer is more important than the other layers on the final outputs so we conduct lower bit quantization on this layer to reduce the attack success rate of adversarial samples and apply higher bit quantization on other layers to maintain the accuracy of the benign inputs. Thus, according to the trade-off between the performance of the benign inputs and the success rate of the adversarial samples, we find that (16, 12)-bit and (16, 12, 12)-bit quantization methods are best choices for PPMI and MIMIC-III datasets.

Moreover, we also conduct our defense solution (SDQDS) on the weights adjustment attack ($\Theta = 70\%$), where the result is shown in Table 5.11. Recall that without the attack, the accuracy will reduce when we apply the quantization-based defense solution on the original model (as shown in Table 5.8a, Table 5.9a and Table 5.10a). The result in Table 5.11 shows that our defense solution (SDQDS) also works for the weights adjustment attack, where the performance will improve by at least 2% and 2.9% at most when such defense solution has been applied.

Table 5.11: Impact of SDQDS on Weights Adjustment Attack (MIMIC-III multi-class)

Defense Method	Dataset	# of Features	# of Classes	Accuracy (Attack with $\Theta = 70$)	Accuracy (16,12,12)	Accuracy (16,12,8)
Dynamic (layers)	MIMIC-III	19	8	76.5%	78.5%	77.3%
Dynamic (features)	MIMIC-III	19	8	76.5%	79.4%	78.1%

5.7 Discussion

In this section, we first discuss the limitation of our quantization based defense solution. Then, we discuss some existing defense mechanisms and check if they are effective in defending two types of attacks that we have investigated. Last but not least, we also describe a possible end-to-end training strategy to prevent the weights adjustment attack.

5.7.1 Limitations of Quantization based Defense Solution

Since various LSTM-based models are created for different domains e.g., natural language processing, speech recognition, etc, a more generic quantization method that is applicable to all LSTM-based models needs to be designed.

While quantization based defense method is simple, there are some limitations. For example, as shown from our work, we need to carefully decide how many bits of quantization we want to use for different layers of the LSTM model in different experiments. This is rather time-consuming. It will be useful for future work to design an algorithm which automatically figures out such values. Furthermore, even though quantization-based solution reduces the attack success rate of adversarial samples, the reduction is less than ideal. Again, future work should focus on coming up with a defense solution that can drastically reduce the attack success rate to zero.

5.7.2 Defense Mechanisms for Adversarial Samples Attack

Here, we discuss if several existing defense mechanisms designed for CNN models can be effective in defending against adversarial attacks on LSTM models.

(1) Removing Unnecessary Features

The first defense is to remove unnecessary features as the authors in [157] have shown that including unnecessary features ruin the strong-robustness of learning-based classifiers by allowing attackers to perturb the inputs slightly to cause the model to fail. However, it is hard to use this defense method since based on our work, we find that a trained model using top m (m=5, 20) features is still subjected to these two types of attacks.

(2) Random Nullification of Features

The second defense introduced in [156] is to randomly nullify features within samples during training to produce a more resilient CNN model which prevents attackers from constructing adversarial examples. We have tried this method and also found it to be ineffective in defending against the two types of attacks against the LSTM model we discussed here.

(3) Defensive Distillation

The third defense [126] trains adversary resistant neural networks using a variant of the distillation method. This has the desirable effect of learning a smoother network, which makes it hard for attackers to generate adversarial examples. However, defensive distillation solution requires much more additional computations during the training process. In addition, it has been shown in a recent work [38] that models generated using the defensive distillation technique is still vulnerable to attacks.

5.7.3 Defense Mechanisms for Weights Adjustment Attack

(1) Existing Defense Methods

There are a few mitigation schemes we can explore for the weights adjustment attack. For PLMs contributed by reputable sources e.g., Google, one can use digital signature created by contributors which can be verified by PLM users. However, the efficiency of signature generation and verification needs to be explored for there are RNN models which comprise tens of millions of parameters. In addition, for PLMs contributed by untrusted sources, as suggested in [170], users can perform outlier detection using the training set. If a feature

extractor PLM generates bigger variations in feature vectors belonging to a group of similar inputs, then this PLM warrants further investigation.

(2) An End-to-End Training Strategy

Table 5.12: End-To-End Defense Method Against Weights Adjustment Attack for T-LSTM

Dataset	# of Features	# of Classes	Accuracy (Without Attack)	Accuracy (Attack on Important Features with $\Theta = 70$)	Accuracy (Attack on Important Features with $\Theta = 80$)
SD	36	2	78.1%	73.7%	74.5%
PD	26	2	77.5%	72.5%	75%
PPMI	306	2	60.9%	56.3%	57.8%
PPMI	306	6	43.8%	39.1%	40.6%

Since the T-LSTM model trains the representations and the classifier separately, so a possible defense solution that can construct a more robust model is to train the classification task end to end by including an extra component to the loss function which measures the distance between a victim class and a target class. Our preliminary result in Table 5.12 suggests that such a strategy makes it hard for an adversary to produce a malicious PLM by merely adjusting the weights of important features, which can prevent the weights adjustment attack from succeeding. However, this method is not general enough to defense against the weights adjustment attack for all RNN-based models.

5.8 Summary

Identifying the security and privacy risks of machine learning models is an active research area. In this chapter, we have presented two potential attacks: (i) adversarial samples attack, (ii) a new weight adjustment (PLMs based) attack approach, which can force a RNN-based model to make wrong predictions. We also design low-cost detection and defense mechanisms to prevent such adversarial attacks. Finally, we conduct extensive experiments using both synthetic and real-world datasets to validate the feasibility and practicality of our proposed schemes.

Chapter 6

Conclusions and Discussions

In this chapter, we summarize our research findings and discuss future research directions.

6.1 Summary

Nowadays, cloud has become a popular platform for data storage and processing. With the availability of cloud resources, many organizations have outsourced their data into the cloud. Healthcare companies have followed the same trends. For example, Personal Health Records (PHR) services allow patients to create, manage, and control their data in a centralized place through the web, which has made the storage, retrieval, and sharing of the medical information more efficient. Furthermore, affordable wearables and powerful smartphones with embedded sensors have allowed users' health status to be monitored and useful sensor data to be uploaded to the cloud easily. Thus, with this exponential growth of the stored large scale data and the growing need for personalized care, researchers are keen on developing data mining methodologies to identify critical factors which affect the prediction results and use such information to aid the healthcare professionals in making better treatment decisions.

While remarkable progresses have been made in the healthcare domain, many challenges and open questions remain. The first obstacle is that in order to prevent information leakage, sensitive medical data needs to be encrypted before outsourcing to the cloud, which make the effective data utilization becomes a big challenge. The second challenge is that unlike

other data sources, medical data is highly ambiguous and noise, which make it difficult to generate predictive clinical models for real-world applications. The third obstacle is that it is hard to gather a large collection of high quality clinical data since institutions or hospitals may not be interested in sharing their useful data and healthcare related information also various from different data sources. Last but not least, most machine learning models only provide predictions without explanations, which prevent medical personnel and patients from adopting such healthcare learning systems. Finally, despite the efficiency of machine learning systems and their outstanding prediction performance, it is still a risk to reuse pre-trained models since most machine learning models that are contributed and maintained by third parties lack proper checking to ensure that they are robust to various adversarial attacks.

In this thesis, we address those challenges by designing an accurate and secure personalized cloud-assisted healthcare system, which allows patients to conduct searches for disease diagnosis based on their own personalized profiles. In terms of methodologies, the summaries of this thesis can be described as follows:

- In Chapter2, we have proposed a Privacy-Preserving Disease Treatment, Complication Prediction Scheme (PDTCPS), which allows users to conduct privacy-aware searches for health related questions based on their individual profiles and lab tests results. Our design also allows healthcare providers and the public cloud to collectively generate aggregated training models to diagnose diseases, predict complications and offer possible treatment options. In addition, to enrich search functionality and protect the clients' privacy, we also design an encrypte index tree, which can support fuzzy keyword search and query unlinkability. Moreover, PDTCPS also hides access patterns and hence addresses the security threat via exposing access patterns in existing searchable encryption schemes.

- In Chapter3, we have proposed useful learning models for Amyotrophic Lateral Sclerosis (ALS), Right Heart Catheterization (RHC) and Depression Disorder Relapse (STAR*D) predictions, which can be used to aid efficient clinical care. We also design an incentive mechanism (IHESS) to encourage participants to share their more

truthful and high quality medical data so that aggregated training models can yield high accuracy.

- In Chapter4, we first design a medical knowledge extraction framework to collect useful data from multiple sources to produce an aggregated dataset, which can be used to generate comprehensive medical diagnosis models. Then, we propose a deep learning based medical diagnosis system (DL-MDS), which allows authorized users to conduct searches for medical diagnosis based on their personalized queries.

- In Chapter5, we have presented two potential attacks: (i) an adversarial samples attack, (ii) a new weights adjustment attack approach, which can force a RNN-based learning model to make wrong predictions. We also propose low-cost detection and defense mechanisms to defend against such adversarial attacks.

For all the schemes we have designed above, we conduct extensive experiments using both synthetic and real-world datasets to validate the feasibility and practicality of our proposed methods.

6.2 Future Work

Based on this thesis, there are many possible extensions of the current approaches. In this section, we suggest a few future research directions.

6.2.1 Multi-Dimensional Range Search over Encrypted Cloud Data

Since real-world datasets are often multi-dimensional, so in order to enable different search functions over encrypted data, many Searchable Encryption (SE) schemes have been proposed. Although most of the existing schemes [34, 85, 102, 136, 144] can support single-dimensional and multi-dimensional range queries, they need a lot of search time with regard to the total number of data records and may leak privacy information, which are not practical in the real-world. Therefore, we would like to enhance our schemes to support more complex query types (e.g., range queries), while preserving the privacy of the query keywords. For example, a medical researcher may want to find the number of diabetic patients

who have taken a specific drug for a long time, and yet still suffer a high blood sugar level, by submitting a query like "(50<age<80) AND (sex="female") AND (illness="diabetes") AND (drug="humira") AND (duration>5 years) AND (blood-sugar>7%)". We believe this will be an important step forward to make searchable encryption practical and scalable on large datasets in a real-world cloud setting.

6.2.2 Diseases Predictions based on Multimodal Data

In the era of information explosion, data from multiple sources are becoming increasingly available. Each of these heterogeneous data sources (image features, personal profiles, lab tests data) is likely to contain a different perspective on the disease risk of a patient. Thus, a combination of the data from these independent sources can provide a more comprehensive and holistic assessment of the diseases. Meanwhile, integration of different data sources can also help in early diagnosis since some of the early symptoms of the diseases may appear in one data source but not the other. Consequently, using just a single data source may limit the ability for early diagnosis. Since some diseases (e.g., cancer) cannot be diagnosed accurately only based on symptoms and risk factors, so we would like to extract information from biological sources (i.e., lab tests data) and transform those non-numeric information into numerical ones and evaluate if they can be useful features for predictions. For example the values "Limb" and "Bulbar" in the "Onset-site" field of the ALS disease could be converted into "0" and "1" respectively. In addition, we can also improve our design by combining multiple types of encrypted health data. For example, we may enhance our scheme to deal with encrypted image features so that we can also do disease diagnosis via encrypted MRI images.

6.2.3 Robust end-to-end Deep Learning based Aggregated Systems

Although a machine learning-based predictive model (mentioned in Chapter4) generated using the aggregated dataset could provide outstanding performance for clinical care, improvements can be made as follows:

- Instead of merely using the traditional learning methods (e.g., Logistic Regression,

Random Forest) to generate disease diagnose modules, we can use deep learning techniques (e.g., Recurrent Neural Network) to create multiclass diagnostic modules which use aggregated information gleaned from heterogeneous medical data sources.

- In order to make our system more robust and efficient, we want to generate an aggregated learning model by combing both diagnostic and query process modules, which can not only improve the accuracy of the prediction results but also the efficiency of the system.

In addition, instead of merely considering the two adversarial attacks (mentioned in Chapter5), we also want to evaluate the feasibility of other existing attacks [37, 72, 108]. For example, an adaptive adversary is an attacker that is aware of the defense methods used in the models, and can adapt attacks accordingly. Thus, in order to prevent from such defense-aware attacks, we want to enhance our detection & defense mechanisms (mentioned in Chapter5) to raise the bars for adversaries to launch successful attacks. Defensive strategies should be able to adapt themselves by learning from previous attacks and estimating possible behaviors of adversaries to minimize the expected loss. For example, we can use a large number of adversarial examples to train a detector to identify unknown adversarial examples, where such "detector" is trained on the binary classification task of distinguishing benign samples from adversarial perturbations. In addition, we can also design an algorithm to automatically determine (i) detection thresholds for detecting adversarial samples and weight-based attacks; (ii) the optimal bits to quantize learning models for defending against adversarial attacks.

Bibliography

[1] PROACT. (2007). https://nctu.partners.org/ProACT.

[2] Right Heart Catheterization. (1996). http://biostat.mc.vanderbilt.edu/wiki/pub/Main/DataSets/rhc.html.

[3] RILUZOLE. (1996). https://en.wikipedia.org/wiki/Riluzole.

[4] DREAM ALS Stratification Prize4Life Challenge. (2015). https://www.innocentive.com/ar/challenge/9933047.

[5] UglyDuckling. (2015). https://www.synapse.org/#!Synapse:syn4941689/wiki/235986.

[6] GuanLab. (2015). https://www.synapse.org/#!Synapse:syn4879287/wiki/236099.

[7] RMSD. (1994). https://en.wikipedia.org/wiki/Root-mean-square_deviation.

[8] Star*D. (2006). https://www.nimh.nih.gov/funding/clinical-research/practical/stard/allmedicationlevels.shtml.

[9] Amazon Cloud Drive. https://www.amazon.com/gp/drive/about.

[10] Dropbox. https://www.dropbox.com/business/landing-t61fl?_tk=sem_b_bing&_camp=sem-b-bing-dropbox-ca-eng|dropbox|exact&_kw=dropbox|e&_ad=10994325664.

[11] Google Health. https://www.google.com/health/.

[12] ImageNet. http://www.image-net.org/.

[13] Keras Application. https://keras.io/applications/.

[14] Microsoft HealthVault. http://www.healthvault.com/.

[15] Model Zoo. https://github.com/BVLC/caffe/wiki/Model-Zoo.

[16] Precision Medicine Initiative (NIH). https://www.nih.gov/precision-medicine-initiative-cohort-program.

[17] The World's leading software developed platform. https://github.com.

[18] UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets.html.

[19] Wuala. http://www.wuala.com/.

[20] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[21] Md Zahangir Alom, Adam T Moody, Naoya Maruyama, Brian C Van Essen, and Tarek M Taha. Effective quantization approaches for recurrent neural networks. *arXiv preprint arXiv:1802.02615*, 2018.

[22] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[23] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[24] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[25] Mauro Barni, Pierluigi Failla, Riccardo Lazzeretti, Ahmad-Reza Sadeghi, and Thomas Schneider. Privacy-preserving ecg classification with branching programs and neural networks. pages 452–468. IEEE, 2011.

[26] Inci M Baytas, Cao Xiao, Xi Zhang, Fei Wang, Anil K Jain, and Jiayu Zhou. Patient subtyping via time-aware lstm networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 65–74. ACM, 2017.

[27] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[28] Ettore Beghi, Tiziana Mennini, Caterina Bendotti, Paolo Bigini, Giancarlo Logroscino, Adriano Chio, Orla Hardiman, Douglas Mitchell, Robert Swingler, Bryan J Traynor, et al. The heterogeneity of amyotrophic lateral sclerosis: a possible explanation of treatment failure. *Current medicinal chemistry*, 14(30):3185–3200, 2007.

[29] Yael Ben-Haim and Elad Tom-Tov. A streaming parallel decision tree algorithm. *Journal of Machine Learning Research*, 11(Feb):849–872, 2010.

[30] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[31] Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D Manning. Modeling biological processes for reading comprehension. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1499–1510, 2014.

[32] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *International conference on the theory and applications of cryptographic techniques*, pages 506–522. Springer, 2004.

[33] Raphael Bost, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser. Machine learning classification over encrypted data. *Crypto ePrint Archive*, 2014.

[34] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *Annual International Cryptology Conference*, pages 290–307. Springer, 2006.

[35] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[36] Ning Cao, Cong Wang, Ming Li, Kui Ren, and Wenjing Lou. Privacy-preserving multi-keyword ranked search over encrypted cloud data. *IEEE Transactions on parallel and distributed systems*, 25(1):222–233, 2014.

[37] Xiaoyu Cao and Neil Zhenqiang Gong. Mitigating evasion attacks to deep neural networks via region-based classification. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, pages 278–287. ACM, 2017.

[38] Nicholas Carlini and David Wagner. Defensive distillation is not robust to adversarial examples. *arXiv preprint arXiv:1607.04311*, 2016.

[39] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14. ACM, 2017.

[40] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 39–57. IEEE, 2017.

[41] David Cash, Joseph Jaeger, Stanislaw Jarecki, Charanjit S Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner. Dynamic searchable encryption in very-large databases: Data structures and implementation. In *NDSS*, volume 14, pages 23–26. Citeseer, 2014.

[42] David Cash, Stanislaw Jarecki, Charanjit Jutla, Hugo Krawczyk, Marcel-Cătălin Roşu, and Michael Steiner. Highly-scalable searchable symmetric encryption with support for boolean queries. In *Advances in Cryptology–CRYPTO 2013*, pages 353–373. Springer, 2013.

[43] Jesse M Cedarbaum, Nancy Stambler, Errol Malta, Cynthia Fuller, Dana Hilt, Barbara Thurmond, Arline Nakanishi, BDNF ALS study group, 1A complete listing of the BDNF Study Group, et al. The alsfrs-r: a revised als functional rating scale that incorporates assessments of respiratory function. *Journal of the neurological sciences*, 169(1):13–21, 1999.

[44] Yan-Cheng Chang and Michael Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In *International Conference on Applied Cryptography and Network Security*, pages 442–455. Springer, 2005.

[45] Chao Che, Cao Xiao, Jian Liang, Bo Jin, Jiayu Zho, and Fei Wang. An rnn architecture with dynamic temporal matching for personalized predictions of parkinson's disease. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 198–206. SIAM, 2017.

[46] Zhengping Che, David Kale, Wenzhe Li, Mohammad Taha Bahadori, and Yan Liu. Deep computational phenotyping. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 507–516. ACM, 2015.

[47] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2722–2730, 2015.

[48] Yang Chen, Li Li, Guo-Qiang Zhang, and Rong Xu. Phenome-driven disease genetics prediction toward drug discovery. *Bioinformatics*, 31(12):i276–i283, 2015.

[49] Yu Cheng, Fei Wang, Ping Zhang, and Jianying Hu. Risk prediction with electronic health records: A deep learning approach. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 432–440. SIAM, 2016.

[50] Edward Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter F Stewart, and Jimeng Sun. Doctor ai: Predicting clinical events via recurrent neural networks. In *Machine Learning for Healthcare Conference*, pages 301–318, 2016.

[51] Edward Choi, Mohammad Taha Bahadori, Elizabeth Searles, Catherine Coffey, Michael Thompson, James Bost, Javier Tejedor-Sojo, and Jimeng Sun. Multi-layer representation learning for medical concepts. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1495–1504. ACM, 2016.

[52] M. Chuah and W. Hu. Privacy-aware bedtree based solution for fuzzy multi-keyword search over encrypted data. In *ICDCSW*, pages 273–281, 2011.

[53] Gari D Clifford and David Clifton. Wireless technology in disease management and medicine. *Annual review of medicine*, 63:479–492, 2012.

[54] Francis S Collins and Harold Varmus. A new initiative on precision medicine. *New England Journal of Medicine*, 372(9):793–795, 2015.

[55] 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. ACM, 2008.

[56] David R Cox. Regression models and life-tables. In *Breakthroughs in statistics*, pages 527–541. Springer, 1992.

[57] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. *Journal of Computer Security*, 19(5):895–934, 2011.

[58] SAS Dave Smith and UK Marlow. Data mining in the clinical research environment, 2007.

[59] Bonnie B Dean, Jessica Lam, Jaime L Natoli, Qiana Butler, Daniel Aguilar, and Robert J Nordyke. Use of electronic medical records for health outcomes research: A literature review. *Medical Care Research and Review*, 66(6):611–638, 2009.

[60] Miguel Delgado. The evolution of health care it: Are current us privacy policies ready for the clouds? In *Services (SERVICES), 2011 IEEE World Congress on*, pages 371–378. IEEE, 2011.

[61] Anandhi Vivek Dhukaram, Chris Baber, Lamia Elloumi, Bert-Jan van Beijnum, and Paolo De Stefanis. End-user perception towards pervasive cardiac healthcare services: Benefits, acceptance, adoption, risks, security, privacy and trust. In *Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2011 5th International Conference on*, pages 478–484. IEEE, 2011.

[62] Wenliang Du, Yunghsiang S Han, and Shigang Chen. Privacy-preserving multivariate statistical analysis: Linear regression and classification. In *SDM*, pages 222–233. SIAM, 2004.

[63] Bradley Efron. Logistic regression, survival analysis, and the kaplan-meier curve. *Journal of the American statistical Association*, 83(402):414–425, 1988.

[64] Irwin Epstein. *Clinical data-mining: Integrating practice and research*. Oxford University Press, 2009.

[65] Marzyeh Ghassemi, Tristan Naumann, Finale Doshi-Velez, Nicole Brimmer, Rohit Joshi, Anna Rumshisky, and Peter Szolovits. Unfolding physiological state: Mortality modelling in intensive care units. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 75–84. ACM, 2014.

[66] I. Goodfellow, Y. Bengio, and A. Courville. Deep learning. *book in preparation for MIT presss*, 2016.

[67] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[68] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[69] Thore Graepel, Kristin Lauter, and Michael Naehrig. Ml confidential: Machine learning on encrypted data. In *Information Security and Cryptology–ICISC 2012*, pages 1–21. Springer, 2013.

[70] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *International Conference on Machine Learning*, pages 1737–1746, 2015.

[71] Veneta Haralampieva and Gavin Brown. Evaluation of mutual information versus gini index for stable feature selection. 2016.

[72] Warren He, James Wei, Xinyun Chen, Nicholas Carlini, and Dawn Song. Adversarial example defenses: Ensembles of weak defenses are not strong. *arXiv preprint arXiv:1706.04701*, 2017.

[73] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701, 2015.

[74] Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301*, 2015.

[75] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

[76] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[77] Margaret M Hoehn and Melvin D Yahr. Parkinsonism onset, progression, and mortality. *Neurology*, 17(5):427–427, 1967.

[78] Ruitong Huang, Bing Xu, Dale Schuurmans, and Csaba Szepesvári. Learning with a strong adversary. *arXiv preprint arXiv:1511.03034*, 2015.

[79] Ruitong Huang, Bing Xu, Dale Schuurmans, and Csaba Szepesvári. Learning with a strong adversary. *arXiv preprint arXiv:1511.03034*, 2015.

[80] J Iavindrasana, G Cohen, A Depeursinge, H Müller, R Meyer, A Geissbuhler, et al. Clinical data mining: a review. *Yearb Med Inform*, 2009:121–133, 2009.

[81] Peter B Jensen, Lars J Jensen, and Søren Brunak. Mining electronic health records: towards better research applications and clinical care. *Nature reviews. Genetics*, 13(6):395, 2012.

[82] Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035, 2016.

[83] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[84] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[85] Seny Kamara and Charalampos Papamanthou. Parallel and dynamic searchable symmetric encryption. In *International Conference on Financial Cryptography and Data Security*, pages 258–274. Springer, 2013.

[86] Seny Kamara, Charalampos Papamanthou, and Tom Roeder. Dynamic searchable symmetric encryption. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 965–976. ACM, 2012.

[87] Edward L Kaplan and Paul Meier. Nonparametric estimation from incomplete observations. *Journal of the American statistical association*, 53(282):457–481, 1958.

[88] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[89] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[90] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[91] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

[92] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

[93] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *AAAI*, volume 333, pages 2267–2273, 2015.

[94] Edmund C Lau, Fionna S Mowat, Michael A Kelsh, Jason C Legg, Nicole M Engel-Nitz, Heather N Watson, Helen L Collins, Robert J Nordyke, and Joanna L Whyte. Use of electronic medical records (emr) for oncology outcomes research: assessing the comparability of emr information to patient registry and health claims data. *Clinical epidemiology*, 3:259, 2011.

[95] Sven Laur, Helger Lipmaa, and Taneli Mielikäinen. Cryptographically private support vector machines. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 618–624. ACM, 2006.

[96] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196, 2014.

[97] AV Lebedev, Eric Westman, GJP Van Westen, MG Kramberger, Arvid Lundervold, Dag Aarsland, H Soininen, I Kłoszewska, P Mecocci, M Tsolaki, et al. Random forest ensembles for detection and prediction of alzheimer's disease with a good between-cohort robustness. *NeuroImage: Clinical*, 6:115–125, 2014.

[98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[99] Jason Lee, Kyunghyun Cho, and Thomas Hofmann. Fully character-level neural machine translation without explicit segmentation. *arXiv preprint arXiv:1610.03017*, 2016.

[100] Jin Li, Qian Wang, Cong Wang, Ning Cao, Kui Ren, and Wenjing Lou. Fuzzy keyword search over encrypted data in cloud computing. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–5. IEEE, 2010.

[101] Li Li, Wei-Yi Cheng, Benjamin S Glicksberg, Omri Gottesman, Ronald Tamler, Rong Chen, Erwin P Bottinger, and Joel T Dudley. Identification of type 2 diabetes subgroups through topological analysis of patient similarity. *Science translational medicine*, 7(311):311ra174–311ra174, 2015.

[102] Ming Li, Shucheng Yu, Ning Cao, and Wenjing Lou. Authorized private keyword search over encrypted data in cloud computing. In *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, pages 383–392. IEEE, 2011.

[103] Maxwell W Libbrecht and William Stafford Noble. Machine learning applications in genetics and genomics. *Nature Reviews Genetics*, 16(6):321, 2015.

[104] Darryl Lin, Sachin Talathi, and Sreekanth Annapureddy. Fixed point quantization of deep convolutional networks. In *International Conference on Machine Learning*, pages 2849–2858, 2016.

[105] Huang Lin, Jun Shao, Chi Zhang, and Yuguang Fang. Cam: cloud-assisted privacy preserving mobile health monitoring. In *IEEE Transactions on Information Forensics and Security*, volume 8, pages 985–997, 2013.

[106] Max A Little, Patrick E McSharry, Stephen J Roberts, Declan AE Costello, and Irene M Moroz. Exploiting nonlinear recurrence and fractal scaling properties for voice disorder detection. *BioMedical Engineering OnLine*, 6(1):23, 2007.

[107] Chang Liu, Liehuang Zhu, Longyijia Li, and Yu'an Tan. Fuzzy keyword search on encrypted cloud storage data with small index. In *Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on*, pages 269–273. IEEE, 2011.

[108] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. 2017.

[109] Hans Löhr, Ahmad-Reza Sadeghi, and Marcel Winandy. Securing the e-health cloud. In *Proceedings of the 1st ACM International Health Informatics Symposium*, pages 220–229. ACM, 2010.

[110] Gary H Lyman and Harold L Moses. Biomarker tests for molecularly targeted therapies–the key to unlocking precision medicine. *The New England journal of medicine*, 375(1):4–6, 2016.

[111] Laura B Madsen. *Data-Driven healthcare: how analytics and BI are transforming the industry*. John Wiley & Sons, 2014.

[112] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3, 2010.

[113] Riccardo Miotto, Li Li, Brian A Kidd, and Joel T Dudley. Deep patient: an unsupervised representation to predict the future of patients from the electronic health records. *Scientific reports*, 6:26094, 2016.

[114] Riccardo Miotto, Li Li, Brian A Kidd, and Joel T Dudley. Deep patient: An unsupervised representation to predict the future of patients from the electronic health records. *Scientific reports*, 6:26094, 2016.

[115] Riccardo Miotto and Chunhua Weng. Case-based reasoning using electronic health records efficiently identifies eligible patients for clinical trials. *Journal of the American Medical Informatics Association*, 22(e1):e141–e150, 2015.

[116] Debahuti Mishra, AK Das, M Mishra, and S Mishra. Predictive data mining: Promising future and applications. *Int. J. of Computer and Communication Technology*, 2(1):20–28, 2010.

[117] Debahuti Mishra, AK Das, M Mishra, and S Mishra. Predictive data mining: Promising future and applications. *Int. J. of Computer and Communication Technology*, 2(1):20–28, 2010.

[118] W James Murdoch, Peter J Liu, and Bin Yu. Beyond word importance: Contextual decomposition to extract interactions from lstms. *arXiv preprint arXiv:1801.05453*, 2018.

[119] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016.

[120] Liqiang Nie, Meng Wang, Luming Zhang, Shuicheng Yan, Bo Zhang, and Tat-Seng Chua. Disease inference from health-related questions via sparse deep learning. *IEEE Transactions on knowledge and Data Engineering*, 27(8):2107–2119, 2015.

[121] Zhi Nie, Pinghua Gong, and Jieping Ye. Predict risk of relapse for patients with multiple stages of treatment of depression. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1795–1804. ACM, 2016.

[122] Zhi Nie, Pinghua Gong, and Jieping Ye. Predict risk of relapse for patients with multiple stages of treatment of depression. In *Proceedings of the 22nd ACM SIGKDD*

International Conference on Knowledge Discovery and Data Mining, pages 1795–1804. ACM, 2016.

[123] N. Papernot, P. McDaniel, A. Swami, and R. Harang. Crafting adversarial input sequences for rnns. *arXiv preprint arXiv:1604.08275*, 2016.

[124] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against deep learning systems using adversarial examples. *arXiv preprint*, 2016.

[125] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against deep learning systems using adversarial examples. *arXiv preprint arXiv:1602.02697*, 2016.

[126] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 582–597. IEEE, 2016.

[127] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[128] Mooi Choo Chuah Qinghan Xue. Incentive design for high quality disease prediction model using crowdsourced clinical data. *Smart Health*, 2017.

[129] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[130] Adnan Siraj Rakin, Jinfeng Yi, Boqing Gong, and Deliang Fan. Defend deep neural networks against adversarial examples via fixed anddynamic quantized activation functions. *arXiv preprint arXiv:1807.06714*, 2018.

[131] Matthew Richardson, Christopher JC Burges, and Erin Renshaw. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the*

2013 Conference on Empirical Methods in Natural Language Processing, pages 193–203, 2013.

[132] John F Roddick, Peter Fule, and Warwick J Graco. Exploratory medical knowledge discovery: Experiences and issues. *ACM SIGKDD Explorations Newsletter*, 5(1):94–99, 2003.

[133] John F Roddick, Peter Fule, and Warwick J Graco. Exploratory medical knowledge discovery: Experiences and issues. *ACM SIGKDD Explorations Newsletter*, 5(1):94–99, 2003.

[134] Philip Sedgwick et al. Pearsons correlation coefficient. *BMJ*, 345:e4483, 2012.

[135] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 373–374. ACM, 2014.

[136] Elaine Shi, John Bethencourt, TH Hubert Chan, Dawn Song, and Adrian Perrig. Multi-dimensional range query over encrypted data. In *Security and Privacy, 2007. SP'07. IEEE Symposium on*, pages 350–364. IEEE, 2007.

[137] Sanasam Ranbir Singh, Hema A Murthy, and Timothy A Gonsalves. Feature selection for text classification based on gini coefficient of inequality. *Fsdm*, 10:76–85, 2010.

[138] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[139] Dawn Xiaoding Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on*, pages 44–55. IEEE, 2000.

[140] Jyoti Soni, Ujma Ansari, Dipesh Sharma, and Sunita Soni. Predictive data mining for medical diagnosis: An overview of heart disease prediction. *International Journal of Computer Applications*, 17(8):43–48, 2011.

[141] Emil Stefanov, Charalampos Papamanthou, and Elaine Shi. Practical dynamic searchable encryption with small leakage. In *NDSS*, volume 71, pages 72–75, 2014.

[142] Jimeng Sun, Fei Wang, Jianying Hu, and Shahram Edabollahi. Supervised patient similarity measure of heterogeneous patient records. *ACM SIGKDD Explorations Newsletter*, 14(1):16–24, 2012.

[143] Mengying Sun, Fengyi Tang, Jinfeng Yi, Fei Wang, and Jiayu Zhou. Identify susceptible locations in medical records via adversarial attacks on deep predictive models. *arXiv preprint arXiv:1802.04822*, 2018.

[144] Wenhai Sun, Bing Wang, Ning Cao, Ming Li, Wenjing Lou, Y Thomas Hou, and Hui Li. Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking. In *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*, pages 71–82. ACM, 2013.

[145] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, et al. Going deeper with convolutions. Cvpr, 2015.

[146] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[147] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[148] Fengyi Tang, Cao Xiao, Fei Wang, and Jiayu Zhou. Predictive modeling in urgent care: a comparative study of machine learning approaches. *JAMIA Open*, 2018.

[149] Nicholas P Tatonetti, P Ye Patrick, Roxana Daneshjou, and Russ B Altman. Data-driven prediction of drug effects and interactions. *Science translational medicine*, 4(125):125ra31–125ra31, 2012.

[150] Athanasios Tsanas, Max A Little, Patrick E McSharry, and Lorraine O Ramig. Accurate telemonitoring of parkinson's disease progression by noninvasive speech tests. *IEEE transactions on Biomedical Engineering*, 57(4):884–893, 2010.

[151] Bo Wang, Aziz M Mezlini, Feyyaz Demir, Marc Fiume, Zhuowen Tu, Michael Brudno, Benjamin Haibe-Kains, and Anna Goldenberg. Similarity network fusion for aggregating data types on a genomic scale. *Nature methods*, 11(3):333, 2014.

[152] Boyang Wang, Yantian Hou, Ming Li, Haitao Wang, and Hui Li. Maple: scalable multi-dimensional range search over encrypted cloud data with tree-based index. In *Proceedings of the 9th ACM symposium on Information, computer and communications security*, pages 111–122. ACM, 2014.

[153] Cong Wang, Kui Ren, Shucheng Yu, and Karthik Mahendra Raje Urs. Achieving usable and privacy-assured similarity search over outsourced cloud data. In *INFOCOM, 2012 Proceedings IEEE*, pages 451–459. IEEE, 2012.

[154] Cong Wang, Bingsheng Zhang, Kui Ren, Janet M Roveda, Chang Wen Chen, and Zhen Xu. A privacy-aware cloud-assisted healthcare monitoring system via compressive sensing. In *INFOCOM, 2014 Proceedings IEEE*, pages 2130–2138. IEEE, 2014.

[155] Jingyi Wang, Jun Sun, Peixin Zhang, and Xinyu Wang. Detecting adversarial samples for deep neural networks through mutation testing. *arXiv preprint arXiv:1805.05010*, 2018.

[156] Q. Wang, W. Guo, K. Zhang, A. G. Ororbia II, X. Xing, C. L. Giles, and X. Liu. Adversary resistant deep neural networks with an application to malware detection. *arXiv preprint arXiv:1610.01239*, 2017.

[157] Shu-Lin Wang, Jin Li, and Jianwen Fang. Predicting progression of als disease with random frog and support vector regression method. In *International Conference on Intelligent Computing*, pages 160–170. Springer, 2016.

[158] Yajuan Wang, Kenney Ng, Roy J Byrd, Jianying Hu, Shahram Ebadollahi, Zahra Daar, Steven R Steinhubl, Walter F Stewart, et al. Early detection of heart failure with varying prediction windows by structured and unstructured data in electronic health records. In *Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE*, pages 2530–2533. IEEE, 2015.

[159] Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*, 2015.

[160] Shuang Wu, Guoqi Li, Feng Chen, and Luping Shi. Training and inference with integers in deep neural networks. *arXiv preprint arXiv:1802.04680*, 2018.

[161] Chen Xu, Jianqiang Yao, Zhouchen Lin, Wenwu Ou, Yuanbin Cao, Zhirong Wang, and Hongbin Zha. Alternating multi-bit quantization for recurrent neural networks. *arXiv preprint arXiv:1802.00150*, 2018.

[162] Ke Xu, Cui Wen, Qiong Yuan, Xiangzhu He, and Jun Tie. A mapreduce based parallel svm for email classification. *Journal of Networks*, 9(6):1640, 2014.

[163] Rong Xu, Li Li, and QuanQiu Wang. driskkb: a large-scale disease-disease risk relationship knowledge base constructed from biomedical text. *BMC bioinformatics*, 15(1):105, 2014.

[164] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017.

[165] Yuhui Xu, Yongzhuang Wang, Aojun Zhou, Weiyao Lin, and Hongkai Xiong. Deep neural network compression with single and multiple level quantization. *arXiv preprint arXiv:1803.03289*, 2018.

[166] Qinghan Xue, Mooi Choo Chuah, and Yingying Chen. Privacy preserving disease treatment & complication prediction system (pdtcps). In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pages 841–852. ACM, 2016.

[167] Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. Question answering using enhanced lexical semantic models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1744–1753, 2013.

[168] Chun-Nam Yu, Russell Greiner, Hsiu-Chin Lin, and Vickie Baracos. Learning patient-specific cancer survival distributions as a sequence of dependent regressors. In *Advances in Neural Information Processing Systems*, pages 1845–1853, 2011.

[169] Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. Deep learning for answer sentence selection. *arXiv preprint arXiv:1412.1632*, 2014.

[170] Ting Wang Yujie Ji, Xinyang Zhang. Backdor attacks against learning systems. In *IEEE Conference on Communications and Network Security*, 2017.

[171] Valentina Zantedeschi, Maria-Irina Nicolae, and Ambrish Rawat. Efficient defenses against adversarial attacks. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 39–49. ACM, 2017.

[172] Valentina Zantedeschi, Maria-Irina Nicolae, and Ambrish Rawat. Efficient defenses against adversarial attacks. *arXiv preprint arXiv:1707.06728*, 2017.

[173] Mingwu Zhang and Tsuyoshi Takagi. Geoenc: Geometric area based keys and policies in functional encryption systems. In *Australasian Conference on Information Security and Privacy*, pages 241–258. Springer, 2011.

[174] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*, 2016.

VITA

EDUCATION

Lehigh University, Bethlehem, PA, USA ***Sept 2013-2019***
Ph.D. in Computer Science, Department of Computer Science & Engineering

Nanjing University of Posts & Telecommunications, NJ, China ***Sept 2010-May 2013***
M.S. in Computer Application Technology, Department of Computer Science & Engineering

Nanjing University of Posts & Telecommunications, NJ, China ***Sept 2006-June 2010***
B.S. in Information Security, Department of Computer Science & Engineering

EXPERIENCE

Data Mining for Wearable Sensors in Health Monitoring ***March 2018-July 2018***
Data Scientist (Intern) Samsung Research America, Inc. (SRA), Mountain View, CA, 94043, USA

- **Medical Data Mining for Disease Prediction** ***May 2017-August 2017***
  Data Scientist (Intern) *Enviveus, Palo Alto, CA, 94306, USA*

- **Automatic Generation & Recommendation for API Mashups** ***Feb 2017-May 2017***
  Data Scientist (Intern) *Fujitsu Laboratories of America, Inc, Sunnyvale, CA, 94085, USA*

- **Cloud-based Music Search & Recommendation System** ***May 2016-August 2016***
  Software Developer (Intern) *Vinci, Beijing, China*

- **Computational Prediction of Gene-cancer Relationship** ***May 2016-August 2016***
  Research Scientist (Intern) *Chinese Academic of Sciences, Beijing, China*

- **CSE 160 Introduction to Data Science** ***September 2018-December 2018***
  Teaching Assistant *Lehigh University, Bethlehem, PA*

  CSE 343/443 Data Mining ***September 2017-December 2017***
  Teaching Assistant *Lehigh University, Bethlehem, PA*

  CSE 303 Operating System ***September 2016-December 2016***
  Teaching Assistant *Lehigh University, Bethlehem, PA*

PUBLICATIONS

[1] ***Qinghan Xue***, Xiaoran Wang, Samuel Meehan, Jilong Kuang, Alex Gao, Mooi Choo Chuah. "Recurrent Neural Networks based Obesity Status Prediction Using Activity Data", *The 17th IEEE International Conference on Machine Learning and Applications (ICMLA), 2018.*

[2] ***Qinghan Xue***, Mooi Choo Chuah. "New Attacks on RNN based Healthcare Learning System and Their Detections", *The Third IEEE/ACM Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE), 2018.*

[3] ***Qinghan Xue***, Mooi Choo Chuah. "Incentive Design for High Quality Disease Prediction Model using Crowdsourced Clinical Data", *Smart Health Journal, 2017.*

[4] ***Qinghan Xue***, Lei Liu, Wei-peng Chen and Mooi Choo Chuah. "Automatic Generation and Recommendation for API Mashups", *The 16th IEEE International Conference on Machine Learning and Applications (ICMLA), 2017.*

[5] ***Qinghan Xue***, Mooi Choo Chuah. "Incentivising High Quality Crowdsourcing Clinical Data For Disease Prediction", *The Second IEEE/ACM Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE), 2017.*

[6] Xin Li, ***Qinghan Xue***, Mooi Choo Chuah. "CASHEIRS: Cloud Assisted Scalable Hierarchical Encrypted Based Image Retrieval System", *IEEE International Conference on Computer Communications (INFOCOM), 2017.*

[7] ***Qinghan Xue***, Mooi Choo Chuah, Yingying Chen. "Privacy Preserving Disease Treatment & Complication Prediction System (PDTCPS)", *ACM Symposium on Information, Computer and Communications Security (AsiaCCS), 2016.*

[8] ***Qinghan Xue***, Mooi Choo Chuah. "Cuckoo-Filter Based Privacy-Aware Search over Encrypted Cloud Data", *11th Mobile Ad hoc and Sensor Network (MSN), 2015.*

[9] Danwei Chen, Hanbing Yang, ***Qinghan Xue***, Yong Zhou. "Live Migration of Virtual Machines Based on DPDT", *6th China Conference on Wireless Sensor Network (CWSN), 2012.*

[10] Danwei Chen, ***Qinghan Xue***, Yun Zhang. "Research of RDP authentication system based on ECC", *Nanjing Youdian Daxue Xuebao (Ziran Kexue Ban)/ Journal of Nanjing University of Posts and Telecommunications, 2012.*

PATENTS

[1] ***Qinghan Xue***, Lei Liu, Wei-peng Chen. "Application Program Interface Mashup Generation", *Fujitsu Laboratories of America, Inc. F1423.10362US01, 2017.*