HEALTHCARE INFORMATION PLATFORM IN AI ERA

by

Yanke Hu

APPROVED BY SUPERVISORY COMMITTEE:

Weili Wu, Chair

Farokh B. Bastani

Latifur Khan

Bhavani Thuraisingham

Dedicated to my family.

HEALTHCARE INFORMATION PLATFORM IN AI ERA

by

YANKE HU, BS, MS

DISSERTATION

Presented to the Faculty of

The University of Texas at Dallas

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY IN

COMPUTER ENGINEERING

THE UNIVERSITY OF TEXAS AT DALLAS

May 2021

ACKNOWLEDGMENTS

I would like to take this opportunity to thank many people I have met along the way.

First of all, I would like to express my sincere thankfulness to my advisor, Dr. Weili Wu, for her supervision, help, and care over my PhD study. My special thanks are also extended to Dr. Ding-Zhu Du, who always gives me in-time feedback on my research. I thank them for providing us with a comfortable and creative research environment and plenty of academic opportunities.

My special thanks go to Dr. Kang Zhang, who brought me to UT Dallas 12 years ago. He provided me financial support and supervised my master's study.

Thanks to Dr. Farokh B. Bastani, Dr. Latifur Khan, and Dr. Bhavani Thuraisingham for serving on my dissertation supervisory committee. Thanks to Dr. William Anderson for serving on my dissertation examining committee. Thanks to Allison Nepomnick for her careful reviewing and help to make my dissertation of the highest quality.

Special gratefulness goes to my PhD classmates 12 years ago, Dr. Dong Wang, Dr. Ling Ding, Dr. Jiaofei Zhong, Dr. Zhiming Chen, and Yifeng Shao for the great time when we were studying together. I am late, but I won't miss it.

Thanks to many great friends and coworkers that I've learned so much from over the industry practice of these years, Dr. Xiaochuan Niu (TeleNav), Manoj Punjabi, Shawn Simon, Omar Hernandez, Shridhar Damle, Kirk Saathoff, Matthew Yates, Jia Xu, Anthony Le, Lalith Maddali, Sukhbir Singh, Johann Verghese, Peggy Xue, Dr. Jiantao Huang, Haixiao Yu, Liren Ding, Changjie Yang, Karen He, Nara Bayarsaikhan, Kevin DeLoach, Travis Powell, Dr. Yunyao Li (Tealeaf, IBM), Xiahong Gao, Gaohao Zhong, Jing Yang, Yongyan Liu, Dr. Bo Geng, Zezhong Hu, Linlin Deng (Fintopia), Dr. Ying Zheng, Ming Li, Wangpeng An, Marcel Zieba, Dr. Juan Terven, Alexsandar Abas, Yang Gu, Jing Tong (AiFi), Dr. Yandan Yang, Dr. Xingde Jiang, Dr. Wenhua Guan, Yongjia Song, Lei Ren, Suresh Venkatesan, Surendar

Thiyagarajah, Bharat Kancharla, Viswanathan Renganathan, Suresh Siva, Brandon Banks, Dr. Molu Shi, Bryant Moscon, and Becky Grosh(Humana). It has been a great pleasure to work with all of you.

Special thanks go to Raj Subramanian, my manager at Humana, who has given me so much freedom and support for conducting research and conference traveling.

It's my great pleasure to meet old friends when I came back to Dallas in 2018, Dr. Jia Lu and Dr. Ying Jia. It's also nice to know so many new friends in the recent 2 years, Chenyang Zhang, Colin Halicki, Lixia Chen, Dr. Jianxiong Guo, Dr. Yi Li, Dr. Shuyang Gu, Dr. Smita Ghosh, Xiao Li, Tiantian Chen, Dr. Chuanwen Luo, Dr. Qiufen Ni, Dr. Siwen Liu, Xingjian Ding, Guoyao Rao, Dr. Liya Xu, Yapu Zhang, Mengzhu Fu, and Mengyu Fu. And thanks for those fun and happy nights of role-playing and reasoning games in this graduating semester, Yihan Liu, Xin Yao, Wenxiong Lu, Xinyuan Lai, Runtian Huang, and Jianyun Lu.

Last but not least, I would like to thank my parents and all my family who are always with me. Thanks for their unconditional support and endless love, which makes me fearless of any adventures. To them, I dedicate this dissertation.

October 2020

HEALTHCARE INFORMATION PLATFORM IN AI ERA

Yanke Hu, PhD
The University of Texas at Dallas, 2021

Supervising Professor: Weili Wu, Chair

Healthcare analytics has attracted increasing research interests as electronic health records (EHR) and medical image data have skyrocketed over the past decade [1]. EHR and lab reports contain rich text, visual, and time series information such as a patient's medical and diagnosis history, radiology images, etc which is the major source for managing and predicting a patient's health status. Meanwhile, *Deep Learning* [2] has greatly pushed forward the research frontier of computer vision, speech recognition, and natural language processing, since its big success in ImageNet 2012 competition [3]. There is an increasing interest in applying state-of-the-art deep learning techniques to the healthcare industry from the combined effort of industry and academia. IBM [4, 5], Amazon [6, 7], and Google [77, 9, 10, 11] all have pushed out their healthcare information services that can provide early symptoms warning, diagnostic support, and help make clinical decisions. Medical schools and healthcare institutes also have conducted extensive research on illness detection, physiological signals classification, mortality early warning detection, Intensive Care Unit(ICU) length of stay prediction, etc with deep learning models [12, 13, 14, 15, 16, 17]. In this dissertation, we will present two use cases of applying the recent progress of *Deep Learning* to the healthcare domain: (1) Faster Healthcare Time Series Classification with Convolutional Feature Engineering, and (2) Deep Healthcare Pre-Trained Language Models on Mobile Devices. Our work not only has

generated several top tier conference papers, but will also lay the foundation for the next generation healthcare information platform development in the US.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Healthcare analytics has attracted increasing research interests as electronic health records (EHR) and medical image data have skyrocketed over the past decade [1]. EHR and lab reports contain rich text, visual, and time series information such as a patient's medical and diagnosis history, radiology images, etc which is the major source for managing and predicting a patient's health status. Meanwhile, deep learning has greatly pushed forward the research frontier of computer vision, speech recognition, and natural language processing, since its big success in ImageNet 2012 competition [3]. There is an increasing interest in applying state-of-the-art deep learning techniques to healthcare industry from the combined effort of industry and academia. IBM [4, 5], Amazon [6, 7], and Google [77, 9, 10, 11] all have pushed out their healthcare information services that can provide early symptoms warning, diagnostic support, and help make clinical decisions. Medical schools and healthcare institutes also have conducted extensive research on illness detection, physiological signals classification, mortality early warning detection, Intensive Care Unit(ICU) length of stay prediction, etc with deep learning models [12, 13, 14, 15, 16, 17].

Despite that, the recent COVID-19 crisis has revealed the big incompetence of the US healthcare information system in respect of efficiently collecting high-risk population data, monitoring epidemic spread trends, and providing healthcare education and help via internet and mobile. The general approach for defending the epidemics is early detection, social distancing education, and tracking the trend, but it's well believed now that the true number of COVID-19 cases in the US is 10 times higher than reported [18, 19, 20, 21], and we still see many people don't take social distancing seriously [22, 23, 24].

There is an urgent need across healthcare companies, medical schools, and hospitals to build a functional and reliable healthcare information platform that can help people better understand their health status and COVID-19 risk, answer their health-related questions, and

connect them with hospitals and doctors, thus eventually defend the ongoing COVID-19 and future epidemics. Our research on faster convolutional feature engineering method of time series data, and new deep learning model compression techniques will not only contribute to the research progress in this field, but will also bring practical values to the development of a better and more competent healthcare information platform that helps better healthcare education, epidemic risk predicting and tracking, and patients care and assistance.

CHAPTER 2

FASTER CLINICAL TIME SERIES CLASSIFICATION WITH FILTER BASED FEATURE ENGINEERING TREE BOOSTING METHODS[1]

2.1 Introduction

Electronic Health Record (EHR) adoption in the US has increased from 13% to more than 90% in the past decade since 2009's Health Information Technology for Economic and Clinical Health (HITECH) Act [1]. EHR contains rich text, visual, and time series information such as a patient's medical and diagnosis history, radiology images, etc which is the major source for managing and predicting a patient's health status. Meanwhile, `"Deep Learning"` has been a buzzword since its big success in ImageNet 2012 competition [3], which has greatly pushed forward the research frontier of computer vision, speech recognition, and natural language processing since then. There is an increasing interest in applying state-of-the-art deep learning techniques to the healthcare industry, especially EHR, from the combined effort of industry and academia. In 2012, IBM started to apply their Watson DeepQA technology to the diagnostic support from a patient's EHR and claim data, after this technology beat the highest-ranked human players in the open-domain question answering show - Jeopardy! [4]. In 2018, Amazon unleashed its HIPAA-compliant language processing service called Comprehend Medical, which can help make clinical decisions, identify a patient's symptoms, and reduce cost from unstructured medical data [6]. In 2019, the United States Patent Office published a patent application from Google [77], which unveiled its intention to build predictive medicare service with EHR data.

A big portion of these EHR applications belongs to the category of multivariate time series classification, such as Intensive Care Unit (ICU) mortality prediction, physiologic

[1]In proceedings of AAAI 2020 Health Intelligence Workshop, Explainable AI in Healthcare and Medicine, Studies in Computational Intelligence, permission granted by Springer Nature Switzerland AG 2021

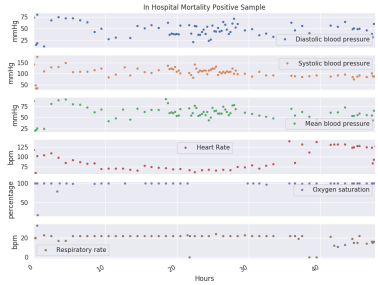
decompensation prediction, ICU length of stay prediction, phenotype classification, and so on. Traditional approaches for clinical time series classification tasks involve feature engineering on timestamp attributes and then applying task-specific classification or regression models [51]. Later, Recurrent Neural Network (RNN) based approaches such as Long Short-Term Memory (LSTM) [37] demonstrated to be powerful even when being trained on raw time series data without feature engineering [38]. Most recently, as RNN architectures being criticized as less effective in parallel computing and time consuming, certain attention based modeling architectures were proposed and evaluated to achieve state-of-the-art performance [39].

Despite the more sophisticated model architectures being applied, the accuracy gain of these deep learning approaches is very slight. Moreover, the speed performance analysis of these deep learning approaches is missing. Practically, if a model costs too much time in training and inference but only gives a little gain in the accuracy improvement, it may not be considered a good option for the machine learning system. In this paper, we propose a Filter based Feature Engineering method and a two-phase auto hyperparameter optimization method, which fits very well for the clinical time series scenario. Combined with two widely used tree boosting methods: XGBoost [44] and LightGBM [40], we demonstrated that our approach achieved the state-of-the-art results with more than 100X speed acceleration compared with RNNs such as LSTM and Gated Recurrent Unit (GRU) [41] on two MIMIC-III benchmark tasks: In-Hospital Mortality Prediction and 25-Phenotype Classification [42]. The major contributions of this work are summarized as the following:

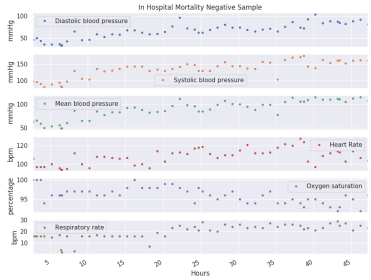
- We proposed an efficient Filter based Feature Engineering method and a two-phase auto hyperparameter optimization method, which fits very well with the clinical time series scenario.

- Combined with two widely used tree boosting methods: XGBoost and LightGBM, We demonstrated that our approach achieved the state-of-the-art results on two MIMIC-III benchmark tasks: In-Hospital Mortality Prediction and 25-Phenotype Classification.

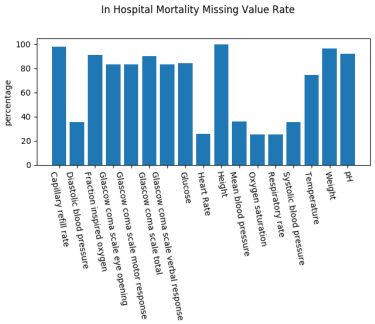
- We conducted detailed speed performance analysis with LSTM, GRU, and our proposed approach on the two MIMIC-III benchmark tasks and demonstrated that our approach achieves more than 100X speed acceleration in training and inference, which means doctors could make the right diagnosis and treatment prognosis in shorter invaluable time.



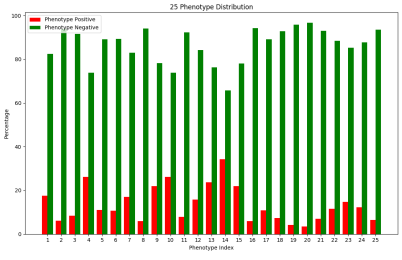
(a) In-Hospital Mortality Positive Sample



(b) In-Hospital Mortality Negative Sample



(c) In-Hospital Mortality Missing Value Rate



(d) 25-Phenotype Distribution

Figure 2.1. MIMIC III Data

2.2 Related Work

Clinical time series classification is challenging due to irregular distribution of the sampling, missing values, and wrong timestamp measurements. Recent year research shows that deep learning methods nearly always outperform traditional machine learning methods such as logistic regression, MultiLayer Perceptron (MLP), etc. The seminal work of applying LSTM to clinical time series data was by Lipton et al. [38], which demonstrated that a simple LSTM network with additional training strategies can outperform several strong baselines. With the growing interest and need for reproducibility of published methods to EHR data, Medical Information Mart for Intensive Care(MIMIC-III) database [43] became the widely accepted public dataset for evaluating competing methods, due to its large size of de-identified clinical data of patients admitted to a single-center Intensive Care Unit (ICU) such as vital signs, medications, laboratory measurements, imaging reports, and more. Based on MIMIC-III dataset, [42] proposed four clinical time series analysis tasks including in-hospital mortality prediction, physiological decompensation prediction, length of stay prediction, and 25-phenotype classification. They benchmarked and demonstrated the performance advantages of LSTM on these four tasks toward the traditional methods such as logistic regression, and joint training LSTM on the four tasks will further improve the performance. Most recently, [39] proposed the first solely attention based sequence modeling architecture for multivariate time series classification, and demonstrated its performance advantages on the four MIMIC-III benchmark tasks.

2.3 MIMIC-III Benchmark Task

In this study, we used the MIMIC-III v1.4, which was released in September 2016. The database contains a cohort of 46520 unique patients with a total of 58976 admissions. We followed [42] to transform the data from the original format to time series format. The sample

input data for these two tasks are quite similar, which is 17 vital signs in time sequence order. **Figure 2.1 (a)** shows a positive sample of in-hospital mortality, with obvious vital signs out of normal range. **Figure 2.1 (b)** shows a negative sample of in-hospital mortality, with normal vital signs. **Figure 2.1 (c)** shows that 11 vital signs have a huge percentage of missing values for the in-hospital mortality prediction task, which is a similar case for 25-phenotype classification task.

2.3.1 In-Hospital Mortality Prediction

This benchmark task is to predict the in-hospital mortality from clinical time series variables recorded in the first 48 hours of the ICU admission. ICU mortality rates are the highest among hospital units(10% to 29%). This is a binary classification task, with the ground truth label indicate whether the patient died before the hospital discharge. *In-hospital-mortality* dataset is generated with the root cohort with further excluding all ICU stays whose *Length-of-Stay* is unknown or less than 48 hours. Our training dataset contains 17939 samples, the validation dataset contains 3222 samples and the test dataset contains 3236 samples. The ground truth label is determined by checking if the patient's date of death is between the ICU admission and discharge time. The overall mortality rate in the dataset is 11.60% (2830 of 24397 ICU stays). Since it's a very imbalanced labeled dataset, we use 3 metrics for the evaluation : (i) Area Under Receiver Operator Curve($AUROC$), (ii) Area Under Precision-Recall Curve($AUPRC$), and (iii) Minimum of Precision and Sensitivity($Min(Se, P+)$).

2.3.2 25 Acute Care Phenotype Classification

This benchmark task is to detect if the patient has any of 25 conditions that are common in adult ICU from clinical time series variables recorded in a single ICU stay episode. These 25 conditions contain 12 critical conditions, such as respiratory failure, 8 chronic conditions,

such as diabetes, and 5 mixed conditions since they are chronic with a periodic acute episode. Because more than 99% of patients in the benchmark dataset have more than one diagnosis, this task is formulated as multi-label binary classification. The phenotype labels are the codes in MIMIC-III ICD-9 diagnosis table, and we only consider the 25 categories matching their HCUP CCS categories. Since MIMIC-III ICD-9 codes are associated with hospital visits, not ICU stays, this benchmark task excludes the hospital admissions with multiple ICU stays for reducing the ambiguity samples: We only consider the samples that have only one ICU stay per hospital admission. Our training dataset contains 36020 samples, the validation dataset contains 6371 samples and the test dataset contains 6281 samples. As Figure 1(d) shows, this is also a very imbalanced labeled dataset, so we use 3 metrics for the evaluation: (i) micro-averaged Area Under the ROC Curve($AUROC$), which computes single $AUROC$ irrespective of the categories, (ii) macro-averaged $AUROC$, which averages $AUROC$ per label, (iii) weighted $AUROC$, which considers each disease prevalence.

2.4 Methods

Figure 2.2 describes the experiment design in this work. Each sample input data is 17 clinical variables in the time sequence order. These 17 clinical variables will be further processed into a one-dimension vector of length 76 after the categorical variables are encoded using a one-hot vector, and the numeric variables are standardized by subtracting the mean and dividing by the standard deviation. The final raw data sample input now can be treated as a two-dimension vector. Deep learning methods are very good at processing the raw data input and producing the features with their internal structure. Tree boosting methods can also take raw data samples as the input with necessary data reshaping, but to get better accuracy, appropriate feature engineering strategies have to be employed. In-hospital mortality prediction has the binary classification ground truth for each sample input, while

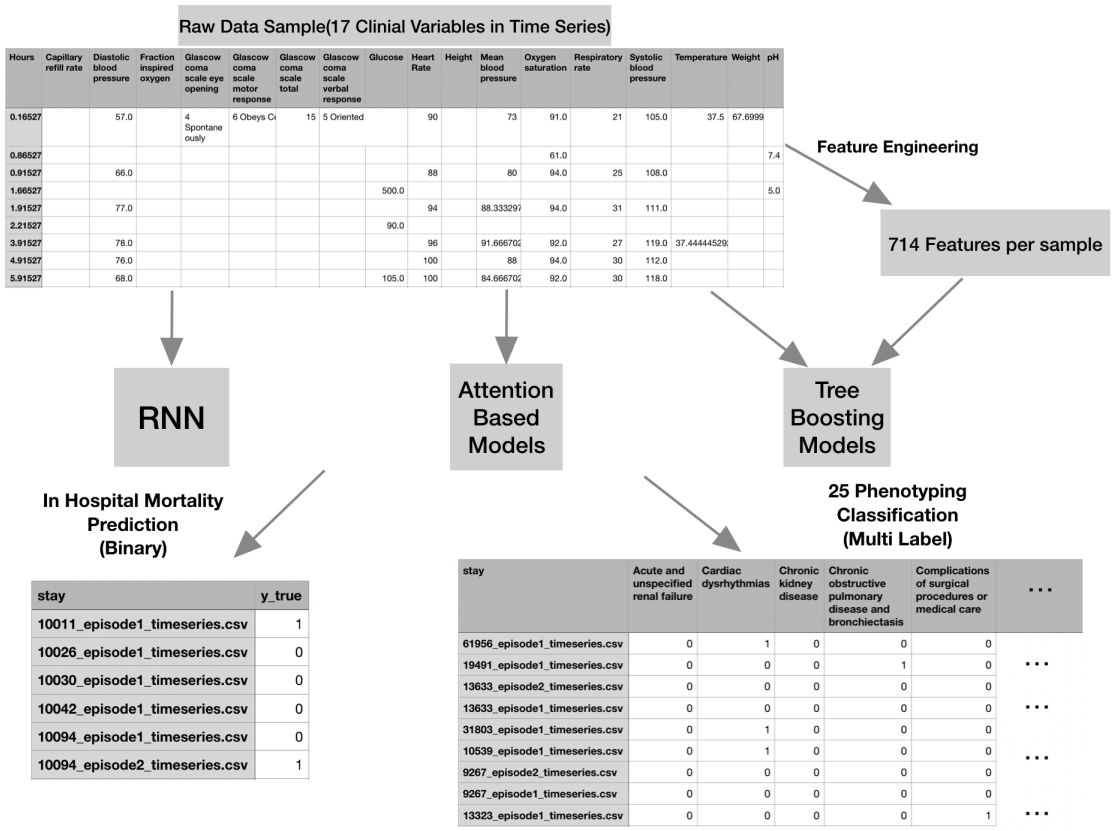


Figure 2.2. MIMIC-III benchmark tasks with different Models

25-phenotype classification has the multiple label binary classification ground truth for each sample input.

2.4.1 RNN on Raw Data

Given a series of observations $\{x_t\}_{t\geq 1}^{T}$, an LSTM model learns to generate the prediction $\hat{y}$ of the ground truth y. Here t stands for a timestamp, and T stands for the length of the series.

$$
(1)\begin{cases}
u_t = \sigma(x_t W^{(xu)} + h_{t-1} W^{(hu)}), \\
f_t = \sigma(x_t W^{(xf)} + h_{t-1} W^{(hf)}), \\
o_t = \sigma(x_t W^{(xo)} + h_{t-1} W^{(ho)} + b^{(o)}), \\
c_t = f_t \odot c_{t-1} + \\
\quad u_t \odot tanh(x_t W^{(xc)} + h_{t-1} W^{(hc)} + b^{(c)}), \\
h_t = o_t \odot \sigma(c_t),
\end{cases}
$$

Here $h_0 = 0$, and the σ (sigmoid) and $tanh$ are element-wise functions. As the formula (1) shows, an LSTM unit has three gates: update gate, forget gate, and output gate. Based on the hidden states $\{h_t\}_{t\geq 1}^{T}$, we add a task-specific layer.

$$
(2) \quad \hat{y} = \sigma(w^{(y)} h_t + b^{(y)})
$$

GRU model is using the following formula for its forward pass procedure:

$$
(3)\begin{cases}
u_t = \sigma(x_t W^{(xu)} + c_{t-1} W^{(hu)}), \\
r_t = \sigma(x_t W^{(xr)} + c_{t-1} W^{(hr)}), \\
c_t = (1 - u_t) \odot c_{t-1} + \\
\quad u_t \odot tanh(x_t W^{(xc)} + r_t \odot c_{t-1} W^{(hc)} + b^{(c)}), \\
h_t = o_t \odot \sigma(c_t),
\end{cases}
$$

As we can see, GRU is not using separate memory cells, and it also uses fewer gates (only update gate and reset gate). In practice, GRU can achieve the LSTM comparable accuracy on many use cases with faster speed (Chung et al., 2014).

2.4.2 Filter Based Feature Engineering (FBFE)

RNN is good at capturing the long-range dependencies of time series data, but it also brings one drawback: data has to be processed time stamp by time stamp both in the training phase and inference phase, which negatively affect the speed performance. In the clinical

time series scenario, missing observations are very common, so the input sample matrix is very sparse. The past study [42] utilized the subsequence and sub-timeframe based feature engineering method with the logistic regression. For any given time series input sample, they computed six different sample statistic features (minimum, maximum, mean, standard deviation, skew, and number of measurements) on seven different subsequences(the full time series, the first 10% of the full time series, the first 25% of the full time series, the first 50% of the full time series, the last 50% of the full time series, the last 25% of the full time series, and the last 10% of the full time series). Thus each time series input sample will generate $17 \times 7 \times 6$ features. This method mainly captures the statistic attribute of the data, but not the sequence dependency, so its accuracy performance is always outperformed by deep learning approaches, moreover, its running time is non-trivial. Here we propose a Filter based Feature Engineering method from the inspiration of the filters in Convolutional Neural Networks(CNN). We take the time series input sample after the one-hot encoding and standardization processing, which is essentially a $T \times 76$ matrix, then we apply convolution operation with an $M \times N$ filter matrix to the input sample, and reshape the output matrix into a one dimension feature vector. Experiments show that our approach can generate more fine-grained features with faster speed.

2.4.3 Tree Boosting Methods with FBFE

Tree Boosting algorithms work on generated features from each sample. After the feature engineering of one time series sample $\{x_t\}_{t\geq 1}^{T}$, it turns into $\{x_i'\}(x_i' \in R^m)$, where m stands for the number of the features. A tree boosting model uses K additive functions to predict the output.

$$(4) \quad \hat{y}_i = \phi(x_i') = \sum_{k=1}^{K} f_k(x_i'), f_k \in F$$

where F is the space of the regression tree. For any given sample $\{x_i'\}(x_i' \in R^m)$, the

decision rules in ϕ classify it into the leaves and summing up all the scores in the corresponding leaves into a final prediction. The model training is to minimize the following regularized objective.

$$(5) \quad L(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

Here l is the loss function that measures that difference between the prediction $\hat{y}_i$ and the ground truth y_i. The function Ω penalizes the complexity of the whole model, which helps smooth the learned weights to prevent over-fitting.

Two tree boosting implementations we used in this study are XGBoost and LightGBM, of which the major difference is their ways of splitting tree nodes. XGboost's original implementation was based on the pre-sorted algorithm, which is to find the best split node based on its pre-sorted feature values. LightGBM adopted a histogram-based algorithm, which is to bucket continuous feature values into discrete bins and use them to construct feature histograms during training.

2.4.4 Two-Phase Auto Hyperparameter Optimization

The performance of most machine learning algorithms hinges on their hyperparameters, which are used to control the training process and set by data scientists before training. For example, RNN methods are sensitive to hyperparameters like learning rate, number of depths, dropout rates, batch size, etc. The performance of tree boosting methods mainly depends on hyperparameters like learning rate, maximum depth, number of leaves in one tree, the maximum number of trees ,etc.

Given a machine learning algorithm A, which has hyperparameters $\lambda_1, ..., \lambda_n$ with respective domains $\Lambda_1, ..., \Lambda_n$. We define its hyperparameter space as $\mathbf{\Lambda} = \Lambda_1 \times \cdots \times \Lambda_n$. We use A_λ to denote the learning algorithm A using the hyperparameter setting $\lambda \in \mathbf{\Lambda}$, and

$l(\lambda) = \mathcal{L}(A_\lambda, \mathcal{D}_{train}, \mathcal{D}_{valid}))$ to denote the validation loss that A_λ achieves on data $\mathcal{D}_{valid}$ after trained on data $\mathcal{D}_{train}$. Our goal is to find the $\lambda \in \mathbf{\Lambda}$ that can minimize $l(\lambda)$.

The Most commonly used auto hyperparameter optimization methods include grid search, random search [45] and bayesian optimization [46]. In this study, we divide the tree boosting auto hyperparameter optimization process into two phases as described in Algorithm 1: In the first phase, we use random search to evaluate the best range of each hyperparameter; In the second phase, we use grid search on a fine-grained scale.

Algorithm 1 Two-Phase Auto Hyperparameter Optimization for Tree Boosting Methods

Input: Target function l; limit T_1, T_2; hyperparameter space $\mathbf{\Lambda}$;
Result: Best hyperparameter configuration in this process λ^*

for $i = 1$ to T_1 **do**
  Random Search λ_i
  $\hat{\lambda_i} \leftarrow$ *Evaluate* $l(\lambda_i)$
end for
for $j = 1$ to T_2 **do**
  Grid Search λ_j *from* $\hat{\mathbf{\Lambda}}$
  $\hat{\lambda_j} \leftarrow$ *Evaluate* $l(\lambda_j)$
end for
return $\lambda^* \in arg\,min_{\lambda_j \in \{\lambda_1, \ldots, \lambda_{T2}\}}$

2.5 Experiments

In this section, we compare the accuracy and speed performance between deep learning methods and our proposed approach on two MIMIC-III benchmark tasks: In-Hospital Mortality Prediction and 25-Phenotype Classification. Our experiments were conducted on a server with hardware configuration: CPU Intel® Core™ i7-8700K CPU @ 3.70GHz x 12 , memory: 32 Gb , GPU: GeForce GTX 1080 Ti/PCIe/SSE2.

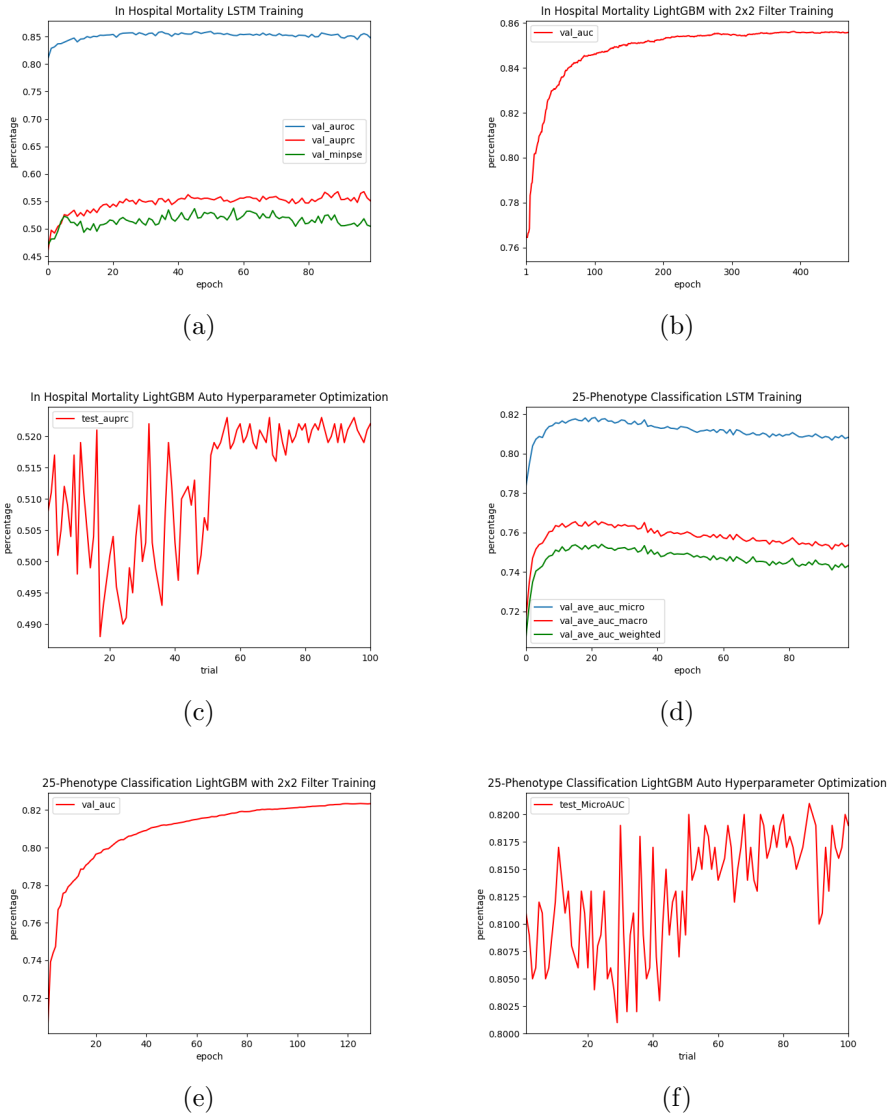


Figure 2.3. Experiments

2.5.1 In-Hospital Mortality Prediction

Figure 2.3(a) shows training a one-layer LSTM model for 100 epochs, with an input dimension of 16, a dropout rate of 0.3, a batch size of 8. The model converges at the 25th epoch. For this trial, it takes around 2125 seconds to find the best model. **Figure 2.3(b)** shows training a LightGBM model with a 2×2 dimension filter, with the early stopping round of 80, the number of leaves of 11, learning rate of 0.07, number of estimators of 10000.

The model converges at the 390th epoch. For this trial, it takes around 61 seconds to find the best model. **Figure 2.3(c)** shows the auto hyperparameter optimization process for LightGBM. In the first 50 trials, it does a random search according to the $AUPRC$ of the test set and estimates the trial range of early stopping round as $[50, 100]$, the trial range of the number of leaves as $[5, 15]$, the trial range of learning rate as $[0.01, 0.1]$, and the trial range of the number of estimators as $[1000, 40000]$. In the second 50 trials, it does the grid search and find several best hyperparameter configurations that can make the model achieve $AUPRC$ of **0.523** on the test set, beating the previous state-of-the-art $AUPRC$ of **0.518**. One of these configurations with the early stopping round as 80, number of leaves as 11, learning rate as 0.07, number of estimators as 10000 can make the model achieve $min(Se, P+)$ of **0.508** on the test set, beating the previous state-of-the-art $min(Se, P+)$ of **0.500**.

Table 2.1 shows the accuracy performance of our methods and previously published works. **Table 2.2** shows averaged training epoch time and averaged test inference time for the single-layer LSTM model, the single-layer GRU model, XGBoost model and LightGBM model on this use case. As we can see, for the MIMIC-III in-hospital-mortality prediction benchmark task, tree boosting methods are more than 100X faster than RNN methods on both training speed and inference speed. Though tree boosting methods may spend more epochs to reach the best model, the overall training time of that is still around $\frac{1}{30}$ of RNN methods.

Table 2.1. Accuracy Comparison for MIMIC-III In-Hospital Mortality Prediction

Metrics	AUROC	AUPRC	min(Se, P+)
LSTM [42]	0.854	0.516	0.491
SAndD [39]	**0.857**	0.518	0.5
FBFE_XGBoost(Ours)	0.856	0.517	0.487
FBFE_LightGBM(Ours)	0.850	**0.523**	**0.508**

Table 2.2. Speed Comparison for MIMIC-III In-Hospital Mortality Prediction

Metrics	Training Epoch (s)	Inference on Test (s)
LSTM_1	85	4.292
GRU_1	65	3.443
FBFE_XGBoost	0.422	0.048
FBFE_LightGBM	**0.156**	**0.006**

2.5.2 25-Phenotype Classification

Figure 2.3(d) shows training a one-layer LSTM model for 100 epochs, with an input dimension of 256, a dropout rate of 0.3, a batch size of 8. The model reached the best accuracy at the 15th epoch and then started to overfit on the training set. For this trial, it takes around 25200 seconds to find the best model. **Figure 2.3(e)** shows training a LightGBM model with a 2×2 dimension filter, with the early stopping round of 10, the number of leaves of 12, learning rate of 0.08, number of estimators of 10000. The model converges at the 119th epoch. For this trial, it takes around 134 seconds to find the best model. **Figure 2.3(f)** shows the auto hyperparameter optimization process for LightGBM. In the first 50 trials, it does a random search according to the $MicroAUC$ of the test set and estimates the trial range of early stopping round as $[5, 30]$, the trial range of the number of leavs as $[10, 20]$, the trial range of learning rate as $[0.05, 0.1]$, and the trial range of the number of estimators as $[1000, 30000]$. In the second 50 trials, it does the grid search and find one hyperparameter configuration that can train the model to tie the previous state-of-the-art on Micro-Averaged $AUROC$ of $\mathbf{0.821}$ and Macro-Averaged $AUROC$ of $\mathbf{0.770}$ on the test set, and beat the previous state-of-the-art on Weighted $AUROC$ of $\mathbf{0.757}$ by a little gap with Weighted $AUROC$ of $\mathbf{0.760}$

Table 2.3. Accuracy Comparison for MIMIC-III 25-Phenotype Classification

Metrics	Micro AUC	Macro AUC	Weighted AUC
LSTM [42]	**0.821**	**0.770**	0.757
SAndD [39]	0.816	0.766	0.754
FBFE_XGBoost(Ours)	0.819	0.768	0.758
FBFE_LightGBM(Ours)	**0.821**	**0.770**	**0.760**

Table 2.4. Speed Comparison for MIMIC-III 25-Phenotype Classification

Metrics	Training Epoch (s)	Inference on Test (s)
LSTM_1	1680	17.466
GRU_1	1330	14.139
FBFE_XGBoost	6.160	0.919
FBFE_LightGBM	**1.124**	**0.154**

Table 2.3 shows the accuracy performance of our methods and previously published works. **Table 2.4** shows averaged training epoch time and averaged test inference time for the single-layer LSTM model, the single-layer GRU model, XGBoost model and LightGBM model on this use case. As we can see, for the MIMIC-III 25-phenotype classification benchmark task, tree boosting methods are more than 100X faster than RNN methods on both training speed and inference speed. Though tree boosting methods may spend more epochs to reach the best model, the overall training time of that is still around $\frac{1}{180}$ of RNN methods.

2.6 Conclusion

Although recent studies have shown that deep learning approaches have achieved state-of-the-art results on several clinical time series tasks in the aspect of accuracy performance, yet their speed performance analysis is missing. Practically, if a model costs too much time in training and inference but only gives a little gain in the accuracy improvement, it may not be considered a good option for the machine learning system. In this work, we developed an efficient Filter based Feature Engineering method and a two-phase auto hyperparameter optimization method, which fits very well with the clinical time series scenario. Combined with two widely used tree boosting methods: XGBoost and LightGBM, we demonstrated that our approach achieved the state-of-the-art results with more than 100X faster speed compared with RNN methods on two MIMIC-III benchmark tasks: In-Hospital Mortality Prediction and 25-Phenotype Classification. Due to its superior accuracy and faster speed advantages, our approach has broad clinical application prospects, especially assisting doctors to make the right diagnosis and treatment prognosis in shorter invaluable time. In the future,

we will continue improving this approach and apply it to broader clinical time series use cases.

CHAPTER 3

FASTER HEALTHCARE TIME SERIES CLASSIFICATION FOR BOOSTING MORTALITY EARLY WARNING SYSTEM[1]

3.1 Introduction

Electronic Health Record (EHR) and healthcare claim data have skyrocketed over the past decade, due to the prevalent adoption of internet and mobile technologies from hospitals and healthcare insurance companies. EHR contains rich text, visual, and time series information such as a patient's medical and diagnosis history, radiology images, etc which is the major source for managing a patient's health status. Healthcare claim data is the insurance claims that patients filed based on their health plans. EHR data normally contains more complete clinical information for patients, but one drawback is that different hospitals or medical systems may have different EHR formats, which leads to challenges of data integration. Compared to EHR, Healthcare claim data contains longitudinal information from all different parties in a patient-centered fashion. In the past, EHR and healthcare claim data are mainly used for patients' health status administration. Recently, there is an increasing interest in predictive analysis with EHR and healthcare claim data.

One important scenario in healthcare applications is multivariate time series classification. EHR contains rich short-term time-stamped nurse-verified physiological measurements for patients admitted to the Intensive Care Unit (ICU), that can be utilized for in-hospital mortality prediction, physiologic decompensation prediction, ICU length of stay prediction, and so on. Healthcare claim data, on the other hand, can be used for longer-term prediction such as 12-month mortality prediction for palliative care.

[1]© 2020 IEEE. Reprinted, with permission, from Yanke Hu, Raj Subramanian, Wangpeng An, Na Zhao, Weili Wu, Faster Healthcare Time Series Classification for Boosting Mortality Early Warning System, International Conference on Intelligent Robots and Systems(IROS), October 2020, Las Vegas, US

Traditional approaches for healthcare time series classification tasks heavily rely on feature engineering on timestamp attributes and then appending with task-specific classification or regression models [51]. Later, Recurrent Neural Network (RNN) approaches such as Long Short-Term Memory (LSTM) [37] was proven to be effective even when being trained with raw time series data without the need for feature engineering [38]. More recently, as RNN architectures being complained as less effective in parallel computing and slow, specific attention based modeling architectures were developed and evaluated to achieve the state-of-the-art result [39].

Despite the more sophisticated model architectures emerging, the accuracy gain of these deep learning approaches is very slight. Moreover, the training and inference time of these sophisticated deep learning models is non-trivial. In this paper, we provide a different angle of solving healthcare multivariate time series classification by turning it into a computer vision problem. We propose a Convolutional Feature Engineering (CFE) methodology, that can effectively extract long sequence dependency time series features. Combined with LightGBM [40], a widely used Gradient Boosting Decision Tree (GBDT) method, it can achieve the state-of-the-art results with 35X speed acceleration compared with LSTM based approach on MIMIC-III In-Hospital Mortality benchmark task [42]. We deploy CFE based LightGBM into our Mortality Early Warning System at Humana and train it on one million member samples. The offline metrics show that this new approach generates better-quality predictions than the previous LSTM based approach, and meanwhile, greatly decrease the training and inference time. The major contributions of this work are summarized as the following:

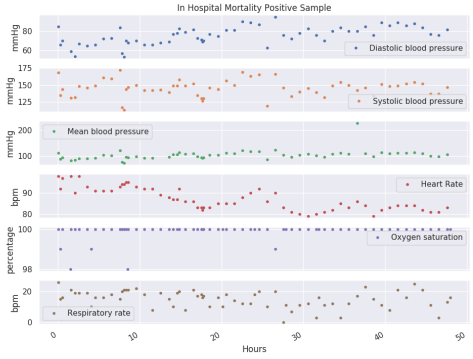
- We propose a different perspective of dealing with healthcare multivariate time series classification problem by encoding the vital signs into a $< 0, 1 >$ vector and aligning these vectors by time series. This will turn each time series sample into a 2-dimension image that can be applied with Convolutional Neural Networks (CNN) image classification models, which is much faster than RNN models.

- We propose a Convolutional Feature Engineering (CFE) methodology, that can effectively extract long sequence dependency time series features. Combined with LightGBM, we demonstrated that this approach can achieve state-of-the-art results with 35X speed acceleration compared with LSTM based approach on MIMIC-III In-Hospital Mortality benchmark task.

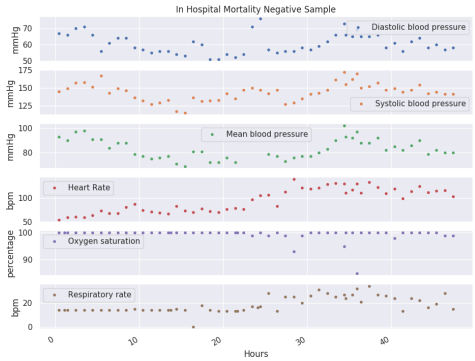
- We deploy CFE based LightGBM into our Mortality Early Warning System at Humana and train it on one million member samples. The offline metrics show that this new approach generates better-quality predictions than the previous LSTM based approach, and meanwhile, decrease the training time from 33 hours to 1 hour.

3.2 Related Work

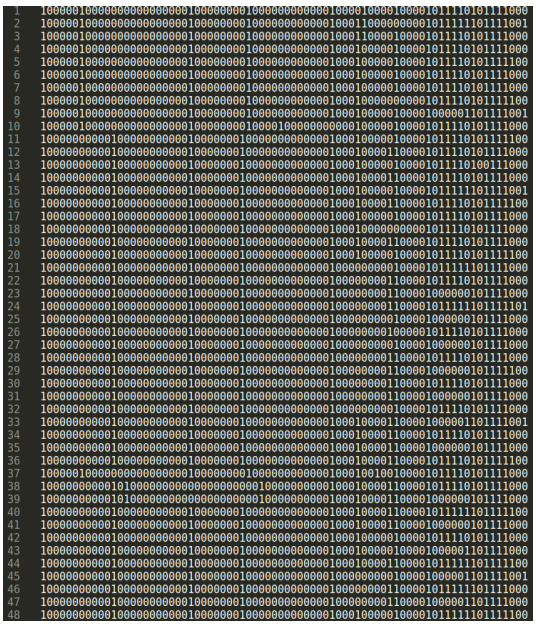
The traditional approach for mortality predictive analytics in ICU heavily involves the formulation of hand-crafted clinical decision rules (CDR) [50], which suffers from questions of limitations of analytics insights, small preselected rules, and constrained usability. Meanwhile, palliative care plays a more important role for old weak, ill or disabled, and automatic screening and notification will greatly help the palliative team proactively approaching the patients rather than relying on referrals from family physicians. The above two needs can be formulated as multivariate time series classification problems such as short-term ICU mortality prediction based on EHR data and longer-term mortality prediction based on longitudinal healthcare claim data from a data science perspective. Healthcare time series classification is very challenging because of the irregular distribution of the sampling, wrong timestamp measurements, and missing values. Recent year research shows that deep learning methods outperform traditional machine learning methods most of the time. Lipton et al. [38] demonstrated that a simple LSTM network with additional training strategies can outperform several strong baselines in 2015. With the growing need and interest of reproducing published



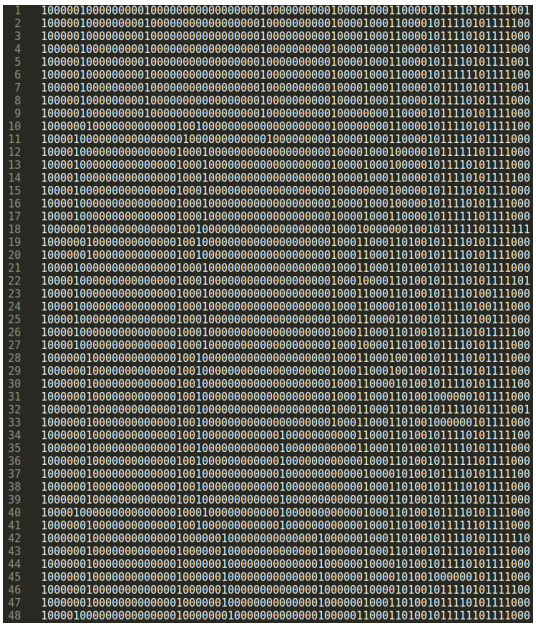
(a) Positive Sample by Vital Sign Measures and Time Series



(b) Negative Sample by Vital Sign Measures and Time Series



(c) Positive Sample after one-hot encoding



(d) Negative Sample after one-hot encoding

Figure 3.1. MIMIC III In-Hospital Mortality Data Sample

methods to EHR, Medical Information Mart for Intensive Care (MIMIC-III) database [43] has established its reputation for evaluating different methods, because of its large size of de-identified clinical data of patients. From MIMIC-III dataset, Harutyunyan et al. proposed four clinical time series analysis tasks containing in-hospital mortality prediction, physiological decompensation prediction, ICU length of stay prediction, and 25-phenotype classification

in 2017 [42]. They demonstrated the performance advantages of LSTM on these four tasks compared to traditional methods such as logistic regression, and joint training with LSTM on the four tasks will improve the performance further. In 2018, Song et al. proposed the sequence modeling architecture solely based on attention mechanism for multivariate time series classification. They demonstrated its performance improvement on the four MIMIC-III benchmark tasks [39].

3.3 Problem Formulation

3.3.1 Multivariate Time Series Classification

We denote a multivariate time series as F variables of length T

$X = (x_1, x_2, ..., x_T) \in \mathbb{R}^{T \times F}$

For each time stamp $t \in \{1, 2, ..., T\}$, $x_t \in \mathbb{R}^F$ represents the t-th measure of F variables, and x_t^f denotes the f-th variable of x_t. Both of the problems in this paper can be formulated as binary classification tasks, where we predict the label $l_i \in \{0, 1\}$ given the time series data $\mathbb{D}$, where $\mathbb{D} = \{(X_i)\}_{i=1}^N$, and $X_i = [x_1^{(i)}, ..., x_{T_i}^{(i)}]$

3.3.2 MIMIC-III In-Hospital Mortality Prediction

This task is to predict the mortality risk from clinical time series variables recorded in the first 48 hours after the ICU admission. Here we use the MIMIC-III v1.4, which was released in September 2016. The database contains a cohort of 46520 unique patients from a total of 58976 admissions. We followed [42] to transform the data from the original format into time series format. Each sample contains 48 timestamps of 17 vital signs including Capillary Refill Rate, Diastolic Blood Pressure, Fraction Inspired Oxygen, Glasgow Coma Scale Eye Opening, Glasgow Coma Scale Motor Response, Glasgow Coma Scale Total, Glasgow Coma Scale Verbal Response, Glucose, Heart Rate, Height, Mean Blood Pressure, Oxygen Saturation,

Respiratory Rate, Systolic Blood Pressure, Temperature, Weight, pH. Our training dataset contains 17939 samples, validation dataset contains 3222 samples and test dataset contains 3236 samples. The ground truth label is determined by checking if the patient's date of death is between the ICU admission and discharge time. The overall mortality rate in the dataset is 11.60% (2830 of 24397 ICU stays). Since it's a very imbalanced labeled dataset, we use 3 metrics for the evaluation : (i) Area Under Receiver Operator Curve($AUROC$), (ii) Area Under Precision-Recall Curve($AUPRC$), and (iii) Minimum of Precision and Sensitivity($Min(Se, P+)$).

3.3.3 Humana Healthcare Claim One Year Mortality Prediction

This task is to predict the mortality risk of that patient within 12 months, given the Humana Claim data of that patient over the past 3 years. Here we use our Humana Mortality Early Warning benchmark dataset, which contains a cohort of 1 million unique patients. We standardized each patient's claim report into a bi-monthly measure, so each sample contains $(3 \times 12 \times 2)$ timestamps of 100 selected vital symptom ICD9 codes including categories like Infectious Diseases, Neoplasms, Blood Organs, Mental Disorder, Nervous System, Circulatory System, Respiratory System, Digestive System, etc. We have additional 10K patient samples as a validation dataset and additional 10k patient samples as a test dataset. The overall mortality rate in the dataset is 17.70%, we use $AUROC$ for the evaluation.

3.4 Convolutional Feature Engineering

3.4.1 Existing Approaches

The past studies mainly utilize the Sub-Timeframe based feature engineering method with the logistic regression. In [42], for any given time series input sample, they compute six different sample statistic features (minimum, maximum, mean, standard deviation, skew and

number of measurements) on seven different subsequences (the full time series, the first 10% of the full time series, the first 25% of the full time series, the first 50% of the full time series, the last 50% of the full time series, the last 25% of the full time series, and the last 10% of the full time series). Thus each time series input sample will generate $17 \times 7 \times 6$ features. One obvious problem of this method is that it mainly captures the statistic attribute of the data, but not the sequence dependency, so its accuracy performance is always outperformed by RNN approaches. Moreover, its running time is non-trivial. [42] then benchmarked a single-layer 16 units LSTM model on the same 4 tasks and demonstrated that it can achieve a much better accuracy without the need for any feature engineering. RNN based approaches are good at capturing the long-range dependencies of time series data, but they have their problems: data has to be processed timestamp by timestamp both in the training phase and inference phase, which negatively affect the speed performance.

3.4.2 Convolutional Neural Networks Approach

Here we are trying to look at this time series problem from a different angle. In the MIMIC-III In-Hospital Mortality prediction case, we apply one-hot encoding of these 17 vital signs into a 76 length vector $\{x_i\}$ ($x_i \in \{0, 1\}$) (for non-categorical vital signs, we will first define degree levels and then transform them into degree categorical values). We align these vital sign vectors by the timestamp order, then each sample will turn into a two-dimension $\{0, 1\}$ array. Now our original problem formulation will turn into:

Given the time series data $\mathbb{D}'$, where $\mathbb{D}' = \{(Y_i)\}_{i=1}^{N}$, $Y_i = [y_1^{(i)}, ..., y_T^{(i)}]$ and $y_t^f(i) \in \{0, 1\}$ denotes the f-th variable of y_t, the task is to assign the label $l_i \in \{0, 1\}$ to Y_i.

Figure 3.1 (a) shows a positive MIMIC-III In-Hospital Mortality data sample in a vital sign time series format. **Figure 3.1 (c)** shows a positive MIMIC-III In-Hospital Mortality data sample in $\{0, 1\}$ transformed format. **Figure 3.1 (b)** shows a negative MIMIC-III In-Hospital Mortality data sample in vital sign time series format. **Figure 3.1 (d)** shows a negative MIMIC-III In-Hospital Mortality data sample in $\{0, 1\}$ transformed format.

We then treat these two dimension arrays as images and apply CNN image classification models. We tested 3 lightweight CNN models: Cifar_10 [47], SqueezeNet [49], MobileNetV2 [48]. The result shows that CNN based approaches can achieve close to state-of-the-art result with 10 times faster speed than a single-layer 16 units LSTM model, which we will elaborate more in the Experiment section.

3.4.3 Convolutional Feature Engineering

Since CNN approaches don't achieve the state-of-the-art result, here we apply with one more enhancement. We use CNN purely for feature engineering and remove the last fully connected layer, which is responsible for classification. We then feed these feature vectors into Gradient Boosting Decision Tree (GBDT) models, which are normally more suitable for structured data. Experiments showed that even a single convolution layer can reserve the sequence dependency attribute for not too large samples (e.g. MIMIC-III in-hospital mortality prediction samples), and greatly speed up the GBDT training process since it shrinks the feature vector length. Moreover, it's at least 200 times faster than the traditional Sub-Timeframe feature engineering method on GPU servers.

3.5 Experiments

3.5.1 MIMIC-III in-hospital Mortality Prediction

For MIMIC-III in-hospital Mortality Prediction, our experiments were conducted on an on-premises server with hardware configuration: CPU Intel® Core™ i7-8700K CPU @ 3.70GHz × 12 , memory: 32 Gb , GPU: GeForce GTX 1080 Ti/PCIe/SSE2 .

Table 3.1 shows the speed comparison of different models on MIMIC-III In-Hospital Mortality Prediction task. A single-layer 16 units RNN model is generally 10 times slower than the three light CNN models. Gradient Boosting Decision Tree (GBDT) methods are

even faster than the three light CNN models. Especially LightGBM is 2 times faster than XGBoost [44] on this task. The Sub-Timeframe feature engineering will generally speed up these two GBDT methods 3 times faster, while the 2×2 filter feature engineering will speed up these two GBDT methods 4 times faster, but Sub-Timeframe feature engineering is more than 200 times slower than 2×2 filter feature engineering.

Table 3.2 shows the best accuracy comparison of different models on MIMIC-III In-Hospital Mortality Prediction task. These three light CNN models achieve close to RNN accuracy. An interesting observation is that even MobileNetV2 [48] is a more sophisticated network, it doesn't achieve better accuracy than classical Cifar_10 [47] and SqueezeNet [49]. 2×2 filter feature engineering not only speed up the GBDT methods, it also helps XGboost achieve the state-of-the-art $AUROC$, and helps LightGBM achieve the state-of-the-art $AUPRC$ and $min(Se, P+)$

Figure 3.2 shows the training process of different models on MIMIC-III In-Hospital Mortality Prediction task. **Figure 3.2 (a)** shows training a single-layer 16 units LSTM model for 100 epochs with a dropout rate of 0.3 and a batch size of 32. The model converges at the 25th epoch, so it takes around $25 \times 85 = 2125$ seconds to find the best model. **Figure 3.2 (b)** shows training classical Cifar_10 CNN network for 100 epochs with a batch size of 32. The model converges at the 6th epoch, so it takes around $6 \times 2.87 = 17.22$ seconds to find the best model. **Figure 3.2 (c)** shows training XGboost with Sub-Timeframe feature engineering with a learning rate of 0.07, a number of estimators of 10000, max depth of 3 and early stopping rounds of 40. The model converges at the 230th epoch, so it takes around $230 \times 0.517 + 112 = 230.91$ seconds to find the best model. **Figure 3.2 (d)** shows training LightGBM with Sub-Timeframe feature engineering with a learning rate of 0.07, a number of estimators of 10000, a number of leaves of 11, and the early stopping round of 80. The model converges at the 120th epoch, so it takes around $120 \times 0.172 + 112 = 132.64$ seconds to find the best model. **Figure 3.2 (e)** shows training XGboost with 2×2 filter feature

engineering with a learning rate of 0.07, a number of estimators of 10000, a max depth of 3 and early stopping rounds of 40. The model converges at the 250th epoch, so it takes around $250 \times 0.422 + 0.37 = 105.87$ seconds to find the best model. **Figure 3.2 (f)** shows training LightGBM with 2×2 filter feature engineering with learning rate of 0.07, number of estimators of 10000, number of leaves of 11, and the early stopping round of 80. The model converges at the 390th epoch, so it takes around $390 \times 0.156 + 0.37 = 61.21$ seconds to find the best model. Compared to the original LSTM model, LightGBM with 2×2 filter feature engineering can achieve the state-of-the-art result with around 35 times faster speed.

3.5.2 Humana Healthcare Claim One Year Mortality Prediction

For Humana Healthcare Claim One Year Mortality Prediction, our experiments were conducted on Microsoft Azure Standard_NC6s_v3 virtual machine. We trained 1 million samples with LightGBM with 2×2 filter feature engineering. The feature engineering time on this training dataset is around 72 seconds, and one training epoch takes around 11.2 seconds. The model converges at the 340th epoch, with $AUROC$ of 0.812 on the test dataset. so it takes around $340 \times 11.2 + 72 = 3880$ seconds ≈ 1.08 hours to find the best model. Our previous LSTM based model normally takes around 33 hours to find the best model, with $AUROC$ of 0.798 on the test dataset.

Table 3.1. Speed Comparison for MIMIC-III In-Hospital Mortality Prediction

Metrics	Training Epoch (s)	Inference on Test (s)	Feature Engineering (s)
LSTM(16 units, 1 layer)	85	4.292	0
GRU(16 units, 1 layer)	65	3.443	0
Cifar_10	2.87	0.355	0
SqueezeNet	3.98	0.782	0
MobileNetV2	8.65	1.002	0
XGBoost	1.899	0.212	0
LightGBM	0.781	0.3	0
XGBoost(Sub-Timeframe)	0.517	0.064	112
LightGBM(Sub-Timeframe)	0.172	0.09	112
XGBoost(2x2 filter)	0.422	0.048	**0.37**
LightGBM(2x2 filter)	**0.156**	**0.006**	**0.37**

Table 3.2. Accuracy Comparison for MIMIC-III In-Hospital Mortality Prediction

Metrics	AUROC	AUPRC	min(Se, P+)
LSTM(16 units , 1 layer)	0.854	0.516	0.491
GRU(16 units , 1 layer)	0.851	0.500	0.482
Cifar_10	0.834	0.469	0.456
SqueezeNet	0.845	0.479	0.461
MobileNetV2	0.841	0.472	0.466
XGBoost	0.848	0.501	0.483
LightGBM	0.846	0.506	0.486
XGBoost (Sub-Timeframe)	0.847	0.478	0.471
LightGBM (Sub-Timeframe)	0.844	0.481	0.469
XGBoost (2x2 filter)	**0.856**	0.517	0.487
LightGBM (2x2 filter)	0.850	**0.523**	**0.508**

3.6 Conclusion

Although recent studies have shown that RNN based approaches are powerful in various time series use cases, yet its drawback of slow processing is easily neglected. In this work, we propose a different perspective of dealing with healthcare multivariate time series classification problem by encoding the vital signs into a $< 0, 1 >$ vector and thus turning it into a computer vision problem. We then propose a Convolutional Feature Engineering methodology, that can effectively reserve long sequence dependency time series features. Combined with LightGBM, it can achieve the state-of-the-art results with 35X speed acceleration compared with LSTM based approach on MIMIC-III In-Hospital Mortality benchmark task. We deploy CFE based LightGBM into our Mortality Early Warning System at Humana and train it on one million member samples. The offline metrics show that this new approach generates better-quality predictions than the previous LSTM based approach, and meanwhile, greatly decrease the training and inference time. In the future, we will continue improving this approach and apply it to broader time series use cases for better palliative care.

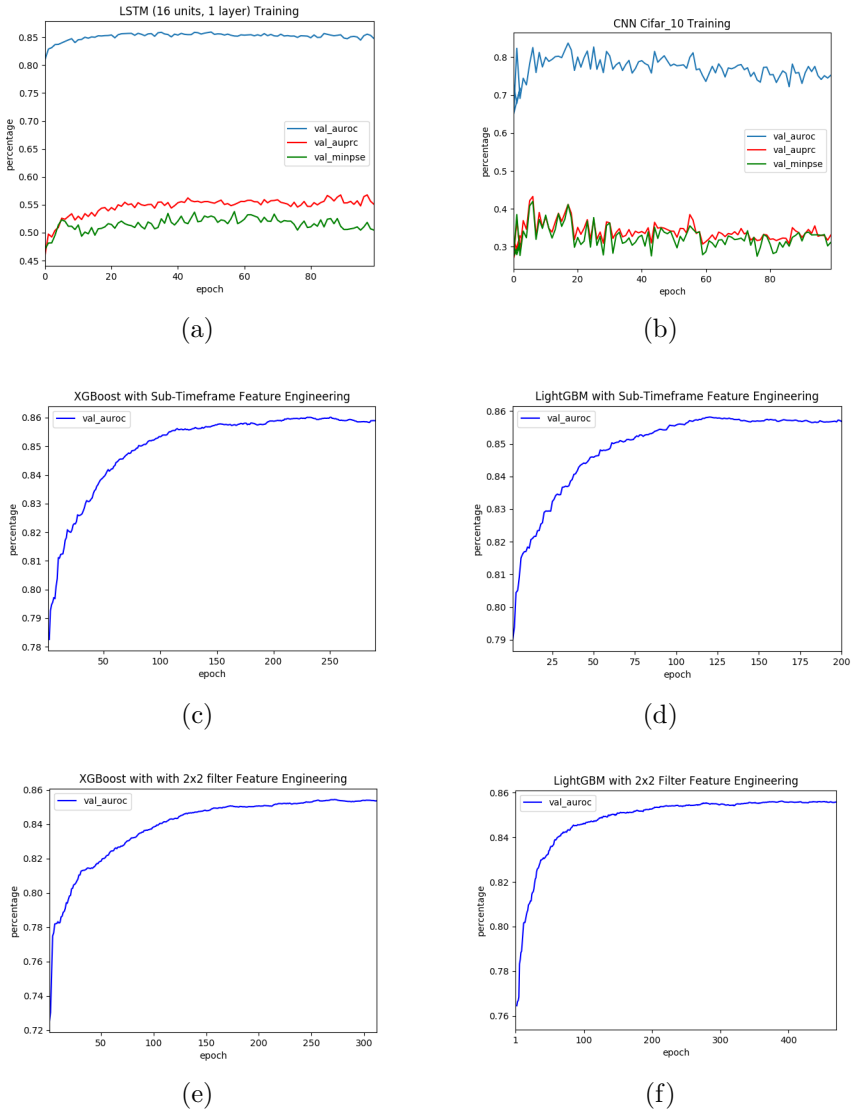


Figure 3.2. MIMIC-III In-Hospital Mortality Training

CHAPTER 4

SQUEEZEBIOBERT: BIOBERT DISTILLATION FOR HEALTHCARE NATURAL LANGUAGE PROCESSING[1]

4.1 Introduction

Healthcare text mining attracts increasing research interest as electronic health records (EHR) and healthcare claim data have skyrocketed over the past decade. Recently, deep pre-trained language models, such as BERT[53] and GPT[54], have improved many natural language processing tasks significantly. However, it won't give satisfactory results by directly applying those deep pre-trained language models to healthcare text mining, because those models are trained from generic domain corpora, which contains a word distribution shift from healthcare corpora. Moreover, deep pre-trained language models are generally heavy and slow, which makes them very difficult to use on resource-restricted devices. Embedded models that can directly inference on mobile are important for healthcare-related apps in the US because (1) it can provide better user experience at poor cell phone signal locations, and (2) it doesn't require users to upload their health sensitive information onto the cloud. In the US, health-related data are only allowed to upload to the cloud by mobile apps being developed by certified institutes, which greatly suppresses the enthusiasm of developing healthcare mobile apps from individual developers. There are some model compression techniques developed recently for generic BERT [57, 58, 59], but there doesn't exist a small and efficient enough pre-trained language model in the healthcare domain. In this work, we developed SqueezeBioBERT. SqueezeBioBERT has 3 transformer layers, and inference much faster while being accurate on healthcare natural language processing tasks. Our contributions are summarized as below:

[1]In proceedings of International Conference on Computational Data and Social Networks(CSoNet) 2020, permission granted by Springer Nature Switzerland AG 2021

- We designed a novel knowledge distillation method, which is very effective for compressing Transformer-based models without losing accuracy.
- We applied this knowledge distillation method to BioBERT[56], and experiments show that knowledge encoded in the large BioBERT can be effectively transferred to a compressed version of SqueezeBioBERT.
- We evaluated SqueezeBioBERT on three healthcare text mining tasks: name entity recognition, relation extraction, and question answering. The result shows that SqueezeBioBERT achieves more than 95% of the performance of teacher BioBERT on these three tasks while being 4.2X smaller.

4.2 Transformer Layer

As the foundation of modern pre-trained language models[53, 54, 55], the transformer layer [52] can capture long-term dependencies of the input tokens with the attention mechanism. A typical transformer layer contains two major components: *multi-head attention* (MHA) and *feed-forward network*(FFN).

4.2.1 Multi-Head Attention

Practically, we calculate the attention function on a query set $\mathbf{Q}$, with key set $\mathbf{K}$ and value set $\mathbf{V}$. The attention function can be defined as below:

$$\mathbf{A} = \frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}} \tag{4.1}$$

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax(\mathbf{A})\mathbf{V} \tag{4.2}$$

where d_k denotes the dimension of $\mathbf{K}$.

Multi-head attention will jointly train the model from different representation subspaces. It is denoted as below:

$$MultiHead(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = Concat(head_1, ..., head_h)\mathbf{W} \tag{4.3}$$

where h denotes attention head number, $head_i$ is computed by **Equation (2)**, and W is the linear parameter weight.

4.2.2 Feed-Forward Network

After multi-head attention, a fully connected feed-forward network will follow, which is denoted as below:

$$FFN(x) = max(0, x\mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2 \tag{4.4}$$

4.3 Knowledge Distillation

A very common way to boost the performance of a machine learning algorithm is to train several models, and then ensemble. Deep learning models are generally heavy neural networks, so it's normally considered too computationally expensive and inefficient to deploy the ensemble of deep neural networks in the production environment. [60] first proposed *Knowledge Distillation* and showed the possibility of compressing the function learned from a large complex model into a much smaller and faster model without significant accuracy loss [61]. As deep learning models are becoming more and more complex, knowledge distillation has shown its power of transferring the knowledge from a group of specialist networks to a single model [61, 62, 63].

Formally, the *Knowledge Distillation* process can be defined as the process of minimizing the loss function between a large teacher network $\mathbf{T}$ and a small student network $\mathbf{S}$ as below:

$$\mathcal{L}_{KD} = \sum_{x \in X} L(f^T(x), f^S(x)) \tag{4.5}$$

where L denotes the loss function to evaluate the difference between $\mathbf{T}$ and $\mathbf{S}$, x is the token input, X is the training set, f^T denotes the output of the teacher network $\mathbf{T}$ and f^S denotes the output of the student network $\mathbf{S}$.

4.4 BioBERT

Deep pre-trained language models, such as BERT[53] and GPT[54], have improved many natural language processing tasks significantly. However, it won't give satisfactory results by directly applying those deep pre-trained language models to healthcare text mining, because those models are trained from generic domain corpora, which contains a word distribution shift from healthcare corpora. BioBERT[56], with almost the same structure as BERT and pre-trained on biomedical domain corpora such as PubMed Abstracts and PMC full-text articles, can significantly outperform BERT on biomedical text mining tasks.

BioBERT has been fine-tuned on the following three tasks: Named Entity Recognition (NER), Relation Extraction (RE), and Question Answering(QA). NER is to recognize domain-specific nouns in a corpus, and precision, recall, and F1 score are used for evaluation on the datasets listed in **Table 4.1**. RE is to classify the relationships of named entities, and precision, recall, and F1 score are used for evaluation on the datasets listed in **Table 4.2**. QA is to answer a specific question in a given text passage, and strict accuracy, lenient accuracy and mean reciprocal rank are used for evaluation on BioASQ factoid dataset[75].

4.5 BioBERT Distillation

In this section, we developed a novel distillation method for BioBERT. Experiments show that knowledge encoded in the large BioBERT can be effectively transferred to the compressed version of SqueezeBioBERT.

Figure 4.1 shows an overview of the proposed knowledge distillation method. Supposing that the teacher BioBERT has M transformer layers and the student SqueezeBioBERT has

Table 4.1. BioBERT Named Entity Recognition Evaluation Datasets

Dataset	Entity Type
NCBI Disease [64]	Disease
2010 i2b2/VA [65]	Disease
BC5CDR [66]	Disease/Drug
BC4CHEMD [67]	Drug
BC2GM [68]	Gene
JNLPBA [69]	Gene
LINNAEUS [70]	Species
Species-800 [71]	Species

Table 4.2. BioBERT Relation Extraction Evaluation Datasets

Dataset	Entity Type
GAD [72]	Gene/Disease
EU-ADR [73]	Gene/Disease
CHEMPROT [74]	Protein

N transformer layers, we distilled BioBERT both on transformer layers and task-specific layers.

Transformer layer distillation consists of multi-head attention distillation and feed-forward network distillation. For multi-head attention distillation, we combine **Equations (4.2)**,**(4.3)**, and **(4.5)**, and use the mean squared error(MSE) as the loss function since it's more suitable for regression tasks. Thus, the multi-head attention distillation process is denoted as below:

$$\mathcal{L}_{MHA} = \frac{1}{h}\sum_{i=1}^{h} MSE(M_i^T, M_i^S) \tag{4.6}$$

where h denotes the number of attention heads, M_i^S denotes the output of i-th student attention head, and M_i^T denotes the output of i-th teacher attention head.

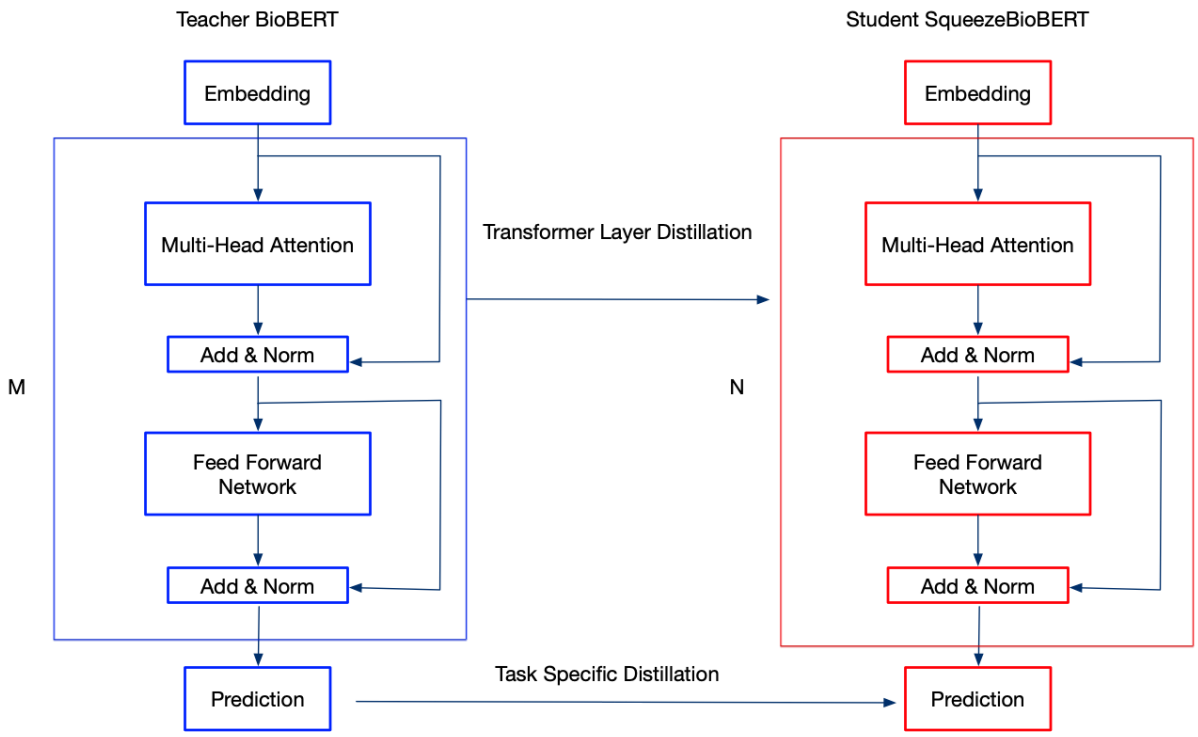


Figure 4.1. The Overview of Distillation from BioBERT to SqueezeBioBERT

For feed-forward network distillation, we can use a single linear transformation W_{FFN} to transform the output of the teacher network into the student network. Thus, the feed-forward network distillation process is denoted as below:

$$\mathcal{L}_{FFN} = MSE(O^T_{MHA}W_{FFN}, O^S_{MHA}) \tag{4.7}$$

For task-specific prediction layer distillation, we use *softmax cross-entropy* as the loss function, since it's more suitable for classification tasks. Thus, the task-specific prediction layer distillation is denoted as below:

$$\mathcal{L}_{pred} = -softmax(O^T_{FFN})log(softmax(O^S_{FFN})) \tag{4.8}$$

Table 4.3. Named Entity Recognition Metrics Comparison

Dataset	metrics	BioBERT-Base v1.1	SqueezeBioBERT v1.0
NCBI Disease [64]	Precision	88.22	86.19
	Recall	91.25	88.42
	F1	89.71	87.74
2010 i2b2/VA [65]	Precision	86.93	83.97
	Recall	86.53	83.85
	F1	86.73	85.26
BC5CDR [66]	Precision	86.47	82.84
	Recall	87.84	84.94
	F1	87.15	85.23
BC4CHEMD [67]	Precision	92.80	89.83
	Recall	91.92	87.78
	F1	92.36	90.33
BC2GM [68]	Precision	84.32	82.46
	Recall	85.12	83.16
	F1	84.72	82.11
JNLPBA [69]	Precision	72.24	69.93
	Recall	83.56	81.67
	F1	77.49	75.09
LINNAEUS [70]	Precision	90.77	89.41
	Recall	85.83	84.29
	F1	88.24	85.15
Species-800 [71]	Precision	72.80	70.47
	Recall	75.36	74.38
	F1	74.06	72.73

In summary, **Equations (4.6)**,**(4.7)** and **(4.8)** describes the overall procedure of the BioBERT distillation process.

4.6 Experiments

We use BioBERT-Base v1.1[76] as our source model, and distilled it to SqueezeBioBERT on the same three healthcare NLP tasks. BioBERT-Base v1.1 has 12 transformer layers and 109M weights. SqueezeBioBERT has 3 transformer layers and 26M weights.

Table 4.4. Relation Extraction Metrics Comparison

Dataset	metrics	BioBERT-Base v1.1	SqueezeBioBERT v1.0
GAD [72]	Precision	77.32	74.69
	Recall	82.68	81.61
	F1	79.83	77.04
EU-ADR [73]	Precision	77.86	75.37
	Recall	83.55	80.54
	F1	79.74	77.19
CHEMPROT [74]	Precision	77.02	75.79
	Recall	75.90	72.41
	F1	76.46	74.01

Table 4.5. Question Answering Metrics Comparison

Dataset	metrics	BioBERT-Base v1.1	SqueezeBioBERT v1.0
BioASQ 4b [75]	Strict Accuracy	27.95	27.31
	Lenient Accuracy	44.10	42.12
	Mean Reciprocal Rank	34.72	33.26
BioASQ 5b [75]	Strict Accuracy	46.00	43.58
	Lenient Accuracy	60.00	58.08
	Mean Reciprocal Rank	51.64	49.94
BioASQ 6b [75]	Strict Accuracy	42.86	41.83
	Lenient Accuracy	57.77	56.48
	Mean Reciprocal Rank	48.43	46.83

NER results are shown in **Table 4.3**, RE results are shown in **Table 4.4**, and QA results are shown in **Table 4.5**. From the results, we can see that SqueezeBioBERT is 4.2X smaller than BioBERT, but still achieves more than 95% accuracy performance of the teacher BioBERT on the three NLP tasks. This proves the efficiency of the proposed method of transferring knowledge encoded in the large BioBERT to the compressed version of SqueezeBioBERT.

4.7 Conclusion

Although recent deep pre-trained language models have greatly improved many natural language processing tasks, they are generally heavy and slow, which makes them very difficult to use on resource-restricted mobile or IoT devices. Embedded models that can directly inference on mobile is important for healthcare-related apps in the US because: (1) it can provide a better user experience at poor cell phone signal locations, and (2) it doesn't require users to upload their health sensitive information onto the cloud. In this paper, we designed a novel knowledge distillation method, which is very effective for compressing Transformer-based models without losing accuracy. We applied this knowledge distillation method to BioBERT, and experiments show that knowledge encoded in the large BioBERT can be effectively transferred to a compressed version of SqueezeBioBERT. We evaluated SqueezeBioBERT on three healthcare text mining tasks: name entity recognition, relation extraction, and question answering. The result shows that SqueezeBioBERT achieves more than 95% of the performance of teacher BioBERT on these three tasks while being 4.2X smaller.

CHAPTER 5

HEALTHREAD: HEALTH NEWS RECOMMENDATION BASED ON YOUR HEALTH STATUS

5.1 Introduction

Mobile news reading has changed people's traditional habits of reading physical newspapers due to its abundant sources and great convenience. Mobile News apps, like Google News [77] and Toutiao [78], collect news from trillions of sources around the world, and provide a user friendly aggregation of the refreshing news. The key challenge here is to help users find the news that most interests them from the overwhelming volumes of the information.

There are mainly two strategies for building recommendation systems: Content-based recommendation and collaborative filtering. The content-based method recommends information based on user profiles built from user interests or activities, while the collaborative filtering method cares more about the opinions of peer users so that it captures the general news trend. The mainstream news recommendation apps nowadays are all trying to find a good balance of these two strategies since individual interests are easily influenced by the general trend[79, 80].

There are numerous ways to build user profiles in news reading apps. Some of them is to ask users to manually fill a survey at the beginning. Some are to adjust certain dimensions of user profiles by measuring users' comments or likes after they read an article. The user-profiles will keep evolving with their explicit and implicit activities of reading. In another word, the news reading apps keep tracking users' reading habits continuously and then generate personalized news recommendations.

In this research work, we are interested in building a healthcare mobile app that can combine personalized healthcare information recommendations and health status tracking. There have been many successful recommendation engine based apps like Google News,

Youtube, Amazon that can accommodate people's needs of reading, watching, and shopping based on users' interests and other attributes like location, but the market lacks an app that provides targeted and personalized healthcare information and meanwhile tracks users' health status. Moreover, many health status monitoring apps on the market (e.g. Humana's Go365 mobile app) track users' health status heavily based on survey information, but users' subjective survey inputs are not always reflecting the real facts (e.g. a claimed early sleeper has many reading logs in the deep night).

The major contributions of this work are summarized as the following:

- We define a user profile that jointly utilizes the user's reading habits and health status. The user profile will be used to recommend personalized healthcare news and articles, thus provide better health guidance. On the other hand, users' reading habits reflect users' health status more accurately in several specific domains.

- We developed a news recommendation mechanism based on the similarity of word embeddings. The mechanism is effective and universal to other domains.

- We developed the software architecture of the healthcare information recommendation engine in Azure, and based on that published an iOS app called "HealthRead" in Apple App Store.

5.2 User Profile Construction

The user profile is the representation of users' reading habits and health status, both of which are changing over time. Reading habits consist of users' sequential click history of news items, reading time and duration, reading comments or judges (e.g. like or report), etc. Health status information, gained from mobile OS API (e.g. iOS HealthKit), contains users' physical conditions (e.g. height, weight), vital signs (e.g. heart rate, blood pressure), exercise amount

(e.g. daily steps, stand hours), sleep quality (e.g. deep sleep duration, overall sleep duration), etc. The raw information of the above two types is used to generate the healthcare word embeddings with the pre-trained BioWordVec [81]

To start simple, we consider two dimensions of the user profile: news entity and peer users of similar properties. Each user profile can be formulated into a two-attribute tuple $P = < E, U >$, where E represents top-k news entity word embedding list generated from reading habits and health status $\{< e_1, w_1 >, < e_2, w_2 >, ..., < e_k, w_k >\}$, and U represents the list of users that have the similar properties $\{u_1, u_2, ...\}$

5.3 News Recommendation

The newly-published healthcare news and articles are crawled in a pre-defined setting (time interval, information sources, etc) and then stored in Elastic Search [82]. Then BioWordVec is employed to extract word embeddings of each article, and then assign top-k label embeddings $A = \{< a_1, w_1 >, < a_2, w_2 >, ..., < a_k, w_k >\}$ to each article.

We divide the news queue segment in timestamp order (e.g. each news queue segment contains 100 news). We compare the topic distributions of each news queue segment and the word embeddings of each user profile. Note that we use the same length of the dimension of user profile entity word embeddings and article feature embeddings, so here we use the cosine similarity between a user profile and an article:

(1) $Sim(P_E, A) = \frac{P_E \cdot A}{||P_E|| \cdot ||A||}$

where $|P_E| = |A| = k$, and $||P_E||, ||A||$ are Euclidean norms.

We now select the news with the similarity greater than the pre-defined threshold in the news queue segment to the candidate recommendation newsgroup. The candidate recommendation newsgroup will be pushed to the users in P_U . Note that some other

properties of the news (e.g. popularity and recency) are taken into consideration in the final ranking as adjustment factors.

5.4 Architecture

Figure 5.1 shows the software architecture of the healthcare information recommendation engine we developed in this work.

We use News-Please[83] to crawl the healthcare news articles in a pre-defined group of web domains. Data ingestion, embeddings extraction, and news recommendation are being deployed in Azure Databricks Notebooks, and orchestrated by Azure Data Factory to run in a pre-defined time interval. The news recommendation lists for each user profile are saved in Azure SQL Data Warehouse and provided as a REST API via Azure App Service. We also developed an iOS app called "HealthRead" and published it on Apple App Store.

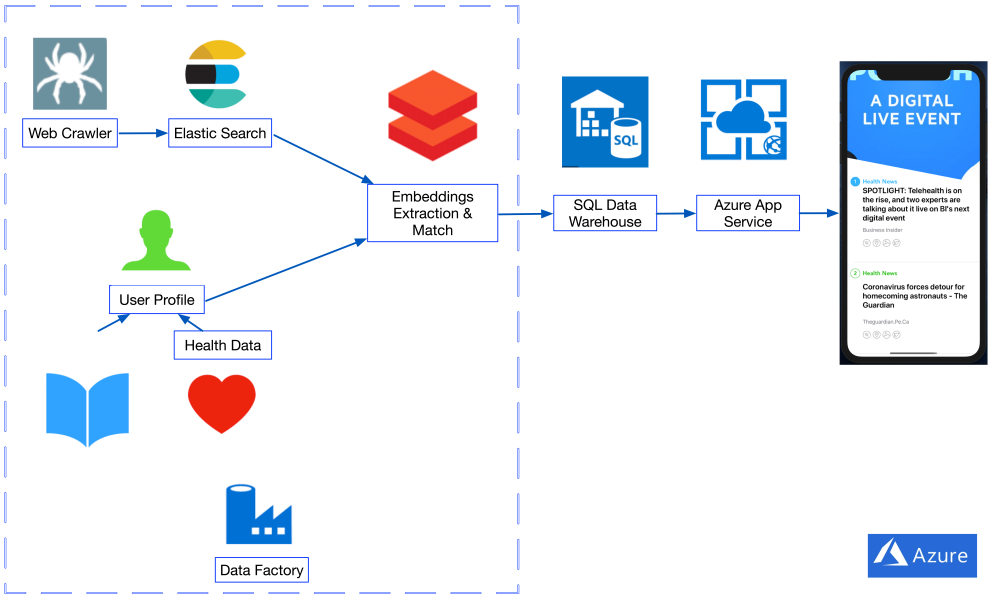


Figure 5.1. HealthRead Architecture

5.5 Conclusion

Despite many successful recommendation engine based apps accommodating people's needs of entertainment, the market lacks an app that provides targeted and personalized healthcare information and meanwhile tracks users' health status. In this paper, We define a user profile that jointly utilizes the user's reading habits and health status, and developed a news recommendation mechanism based on the similarity of word embeddings of user profile and news items. We developed the software architecture of the healthcare information recommendation engine in Azure, and based on that published an iOS app called "HealthRead" in Apple App Store. We will continue polishing and improving this App for better usability.

CHAPTER 6

EXTRACTIVE SUMMARIZATION WITH VERY DEEP PRETRAINED LANGUAGE MODEL[1]

6.1 Introduction

Document summarization is a widely investigated problem in natural language processing and is mainly classified into two categories: extractive summarization and abstractive summarization. Extractive approaches work in the way of extracting existing words or sentences from the original text and organizing into the summary, while abstractive approaches focus more on generating the inherently same summary that is closer to the way human expresses. We will focus on extractive summarization in this work.

Traditional extractive summarization approaches can be generally classified into two ways like greedy approaches[84] and graph-based approaches[85]. Recently, deep learning techniques have been proven successful in this domain. Kageback et al. 2014[86] utilized continuous vector representations in the recurrent neural network for the first time and achieved the best result on the Opinosis dataset[87]. Yin et al. 2015 [88] developed an unsupervised convolutional neural network for learning the sentence representations, and then applied a special sentence selection algorithm to balance sentence prestige and diversity. Cao et al. 2016 [89] applied the attention mechanism in a joint neural network model that can learn query relevance ranking and sentence saliency ranking simultaneously and achieved competitive performance on DUC query-focused summarization benchmark datasets. Cheng et al. 2016 [90] developed a hierarchical document encoder and an attention-based extractor and achieved results comparable to the state of the art on CNN/Daily Mail corpus [91] without linguistic annotation.

[1]In proceedings of International Journal of Artificial Intelligence & Applications(IJAIA), Mar 2019, permission granted by AIRCC Publishing Corporation

The recent development of GPT [54] and BERT [53] has proven the effectiveness of a generative pre-trained language model on a wide range of different tasks, such as text classification, question answering, textual entailment, etc, but neither of these approaches has been tested on the text summarization task.

In this paper, we introduced a two-phase encoder-decoder architecture based on BERT. We fine-tuned this model on the CNN/Daily Mail corpus for single-document summarization task. The result demonstrated that our model has state-of-the-art comparable performance by both automatic metrics (in terms of ROUGE [92]) and human assessors. To the best of our knowledge, this is the first work that applies BERT based architecture to a text summarization task.

6.2 Training Dataset

CNN/Daily Mail [91] dataset is the most widely used large scale dataset for summarization and reading comprehension. The training set contains 287226 samples. The validation set contains 13368 samples. The test set contains 11490 samples. Each sample contains an article and a referenced summary. After tokenization, an article contains 800 tokens on average, and a corresponding summary contains 56 tokens on average.

6.3 Summarization Model

In this section, we propose the summarization model that efficiently utilizes BERT [53] as the text encoder. The architecture is shown in **Figure 6.1** and consists of two main modules, BERT encoder and sentence classification.

The sentences of the original document will be feed into the BERT encoder sequentially and be classified as if or not to be included in the summary. BERT is essentially a multi-layer bidirectional Transformer [52] encoder. Each layer consists of a multi-head self-attention sub-layer and a following linear affine sub-layer with the residual connection. The BERT

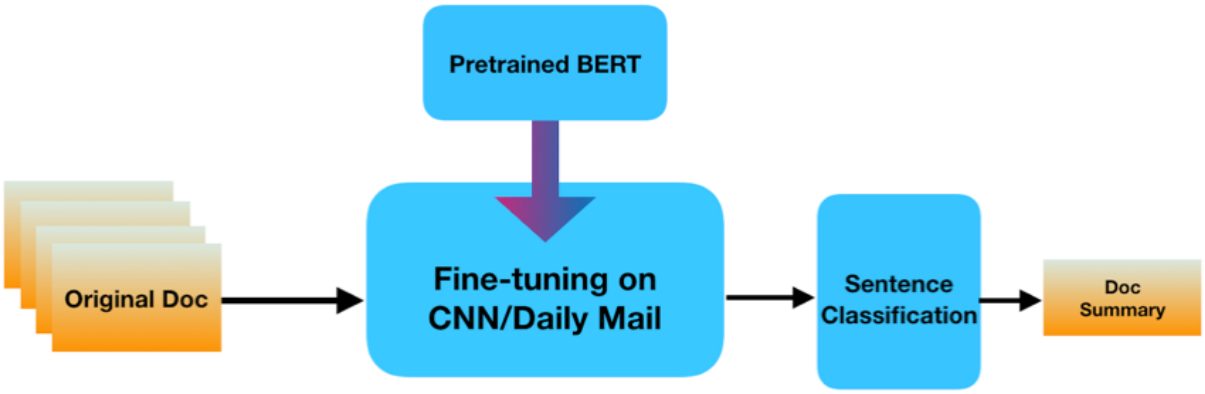


Figure 6.1. The Architecture of the Summarization Model

encoding process utilizes query matrix W^q, key matrix W^k of dimension d_k, and value matrix W^v of dimension d_v, and it can be expressed as **Eq. 6.1**.

$$BERT(W^q, W^k, W^v) = softmax(\frac{W^q(W^k)^T}{\sqrt{d_k}})W^v \tag{6.1}$$

Given the input document as $X = \{x_1, \ldots, x_m\}$ where x_i denotes one source token. The BERT encoder's output can be expressed as **Eq. 6.2**.

$$H = BERT(W^q, W^k, W^v)\{x_1, \ldots, x_m\} \tag{6.2}$$

In CNN/Daily Mail dataset, the ground truth is the referenced summaries without sentence labels, so we follow Nallapati et al. [100] 's method to convert the abstractive referenced summaries to extractive labels. The idea is to add one sentence each time incrementally to the candidate summary so that the Rouge score of the current set of selected sentences is increasing regarding the referenced summary. We stopped when the remaining candidate sentences cannot promote the Rouge score for the referenced summary. With this approach, we convert a text summarization task to a sentence classification task.

Table 6.1. Comparative evaluation of BERT Summarization with recently reported summarization systems

Models	ROUGE-1	ROUGE-2	ROUGE-L
Pointer Generator [93]	36.44	15.66	33.42
ML + Intra-Attention [94]	38.30	14.81	35.49
Saliency + Entailment reward [95]	40.43	18.00	37.10
Key information guide network [96]	38.95	17.12	35.68
Inconsistency loss [97]	40.68	17.97	37.13
Sentence Rewriting [98]	40.88	17.80	38.54
Bottom-Up Summarization [99]	41.22	18.68	38.34
BERT Summarization (Ours)	37.30	17.05	34.76

Our BERT encoder is based on Google's TensorFlow implementation (TensorFlow version $\geq$ 1.11.0). We used the BERT-Base model(uncased, 12-layer,768-hidden,12-heads, 110M parameters) and then fine-tune the model on the training set of CNN/Daily Main corpus. All inputs to the reader are padded to 384 tokens; the learning rate is set to 3×10^{-5}, and other settings are by default.

6.4 Experiment Result

We evaluate ROUGE-1, ROUGE-2, and ROUGE-L [92] of our proposed model on CNN/Daily Mail test dataset. This automatic metric measures the overlap of 1-gram (R-1), bigrams (R-2), and the longest common subsequence between the model generated summaries and the reference summaries. We also listed the previously reported ROUGE scores on CNN/Daily Mail in Table1for comparison. As **Table 6.1** shows, the ROUGE score of BERT summarization is comparable to the start-of-the-art models.

As per Schluter [101], extractive summarization performance tends to be undervalued for ROUGE standard, so we also conduct a human assessment on Amazon Mechanical Turk (MTurk) for the relevance and readability between Bottom-Up Summarization model (best-reported score on CNN/Daily Mail) and our BERT Summarization model. We selected

Table 6.2. Human assessment: pairwise comparison of relevance and readability between Bottom-Up Summarization [99] and BERT Summarization

Models	Relevance	Readability	Total
Bottom-Up Summarization [99]	42	38	80
BERT Summarization (Ours)	49	46	95
Tie	9	16	25

3 human annotators who had an approval rate over than 95% (at least 1000 HITs) and showed them 100 samples from the test dataset of CNN/Daily Mail, including the input article, the reference summary, and the outputs of the two candidate models. We asked them to choose the better one between the two candidate models' outputs, or choose "tie" if both outputs were equally good or bad. The relevance score is mainly based on if the output summary is informative and redundancy free. The Readability score is mainly based on if the output summary is coherent and grammatically correct. As shown in **Table 6.2**, our BERT Summarization model achieves higher scores than the best reported Bottom-Up Summarization model by human assessment.

6.5 Conclusion

In this paper, we present a two-phase encoder-decoder architecture based on BERT for extractive summarization task. We demonstrated that our model has state-of-the-art comparable performance on CNN/Daily Mail dataset by both automatic metrics and human assessors. To the best of our knowledge, this is the first work that applies BERT based architecture to a text summarization task. In the future, we will test employing better decoding and reinforcement learning approaches to extend it to an abstractive summarization task.

CHAPTER 7

TEALEAF CXMOBILE – REPLAYING REAL-TIME CUSTOMER EXPERIENCE[1]

7.1 Introduction

With the rapid evolvement of mobile and cloud technology [104], improving and optimizing business mobile apps' customer experience is unprecedentedly important[105]. Mobile developers are expecting a tool that can help them know how customers are interacting with their apps, and how their apps reacting to the customers' behaviors[106]. If there is any imperfection of the mobile app interface design or bugs being thrown out, they would want to know that as earlier as possible. This helps a lot to accelerate the conversion rate and avoid business losses.

There is a term "Growth Hacking" [107] that describes this kind of business strategy. Growth Hacking is mainly the strategy of pushing product growth by testable and repeatable data analysis. The traditional marketer has a very broad focus[108], and while there may be many factors that influence the business growth, it is not as necessary for a newly built product to do all around marketing channels [109]. Growth Hacking will help to find out the most important factors that truly matter to your business, positive or negative, so that the businesses can reinforce their strength and eliminate their drawbacks [110].

There are a lot of raw data analytics tools on the market to help those online businesses make improvements, like Mixpanel [111], New Relic [112], Heap Analytics [113], Google Analytics [114]. Most of those tools provide many metrics that track and measure users' engagement, page views, mobile device status. They may also provide A/B Testing [115], codeless instrumentation, or some general insights based on those data. But no one can tell

[1]© 2016 IEEE. Reprinted, with permission, from Yanke Hu, Tealeaf cxMobile – Replaying Real-Time Customer Experience, IEEE International Conference on Geoinformatics, August 2016, Galway, Ireland

businesses why and how their mobile customers are struggling in the way of what you see is what you get.

IBM Tealeaf CxMobile is the first-ever tool on the market to combine qualitative data with statistics, giving online businesses the ability to precisely see what their consumers see. This unique solution gives online businesses the ability to capture and replay and analyze every interaction in their mobile apps or on their mobile websites regardless of what device or browser was used. Data entered by users is captured and synchronized with important device event information such as orientation changes and touch screen interactions like swipe, zoom, and scroll. This data stream lets online businesses understand customer behavior and provides key insight around struggles the customers may be encountering on their mobile channel. On the server side, Tealeaf dashboard quantifies the monetary impacts of any known issue and offers the ability to use dimensional reporting to discover exactly what range of devices and browsers are affected. This insight allows companies to prioritize issues based on known quantities to eliminate the issues having the biggest impact on their business.

IBM Tealeaf CxMobile is using the Modularization SDK methodology [103], which is essentially encapsulating core function modules into SDKs and centralize the backend reporting. Compared to the traditional Ad-hoc approach [116], Modularization SDK methodology can bring several advantages including removing redundant implementation and bringing more scalabilities.

In this paper, we will explain how Tealeaf Native Mobile Replay technology works. We will also give an example of applying the Modularization SDK methodology to integrate the Geo-Location logging functions into IBM Tealeaf CxMobile. The rest of the paper is organized as the following. Section 2 describes the overview of Tealeaf CxMobile and Tealeaf Native Replay technology. Section 3 explains how Tealeaf Native Replay technology works on iOS and Android mobile platforms. Section 4 illustrates how to integrate the Geo-Location logging functions into IBM Tealeaf CxMobile. Section 5 concludes the paper.

7.2 Tealeaf CxMobile and Native Replay

Mobile experience plays a central role in the minds of consumers, and businesses know that mobile is the key channel to win and retain their customers. Customers nowadays have high expectations of businesses' mobile channel performance. A poor mobile experience can affect the overall customer relationship. So it's key for organizations to have visibility into their customers' mobile experience. Companies must be able to quickly gather and analyze mobile data in a way that allows them to understand mobile customers' experiences and ultimately improve their mobile sites and apps.

IBM Tealeaf cxMobile provides expansive visibility into the mobile customer experience. It can help discover why mobile customers succeed or fail, automatically detect customer struggles, obstacles, or issues, drill down into actual user behavior (complete with gestures), translate customer feedback into actionable improvements, correlate customer behavior with network and application data.

The key technology to make Tealeaf CxMobile unique is the Tealeaf Native Mobile Replay. Tealeaf Native Mobile Replay is the ability to replay a user's journey through a native iOS or Android app. The capture and replay of the traditional websites' experience has been the key selling point of Tealeaf products for more than 10 years. But capture and replay the native mobile apps' experience is not that easy, since we don't have a universal powerful language like Javascript on mobile platforms that can react to any callback on any browser.

In Tealeaf product, the replay of Tealeaf mobile session occurs in Tealeaf's browser-based replay (BBR) window as show in **Figure 7.1**. The user's journey through the app is represented as a series of steps that are drawn onto a representation of a mobile device screen.

The replay of Native iOS and Android applications relies on the native iOS and Android SDKs that are configured to collect data about the users' interactions. The SDKs will periodically post JSON data representing these interactions to "Tealeaf Target" URL, where

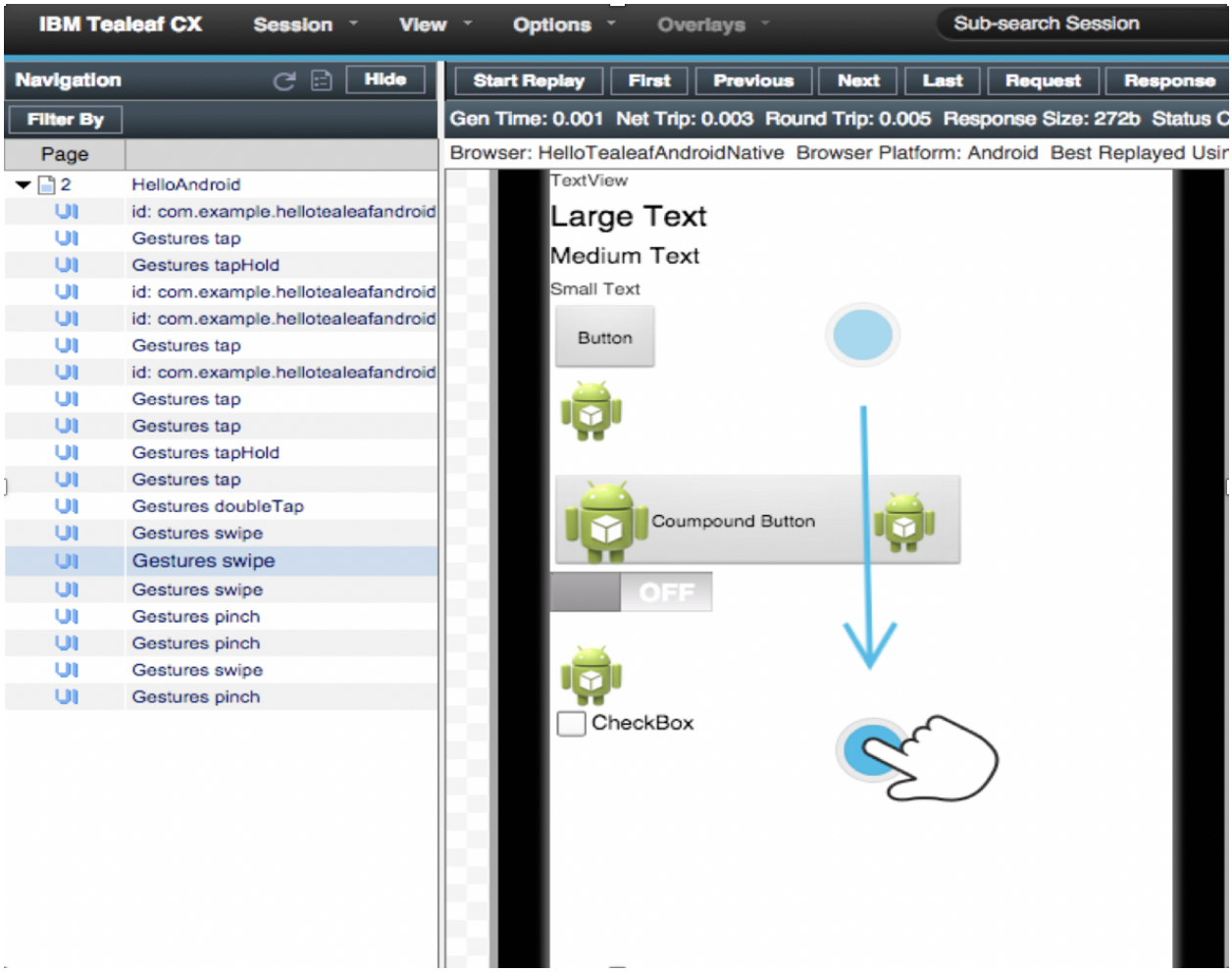


Figure 7.1. Tealeaf Browser-based Replay (BBR)

our monitoring and reporting servers are located. The JSON data posted by the SDKs consists of client environment information and an array of messages that describe the user's journey through the application. Each message has a message "type" where different message types represent information on different application events.

Type 10 and Type 4 messages are probably the most important message types for Native Mobile Replay. Type 10 messages describe the screen state at a moment in real-time and will be converted to HTML for replay in the Browser-Based Replay (BBR) window. Type 4 messages describe UI events that occurred while a mobile view was loaded, e.g. the user clicked on a button, or entered a text box, or the app goes to background. The HTML

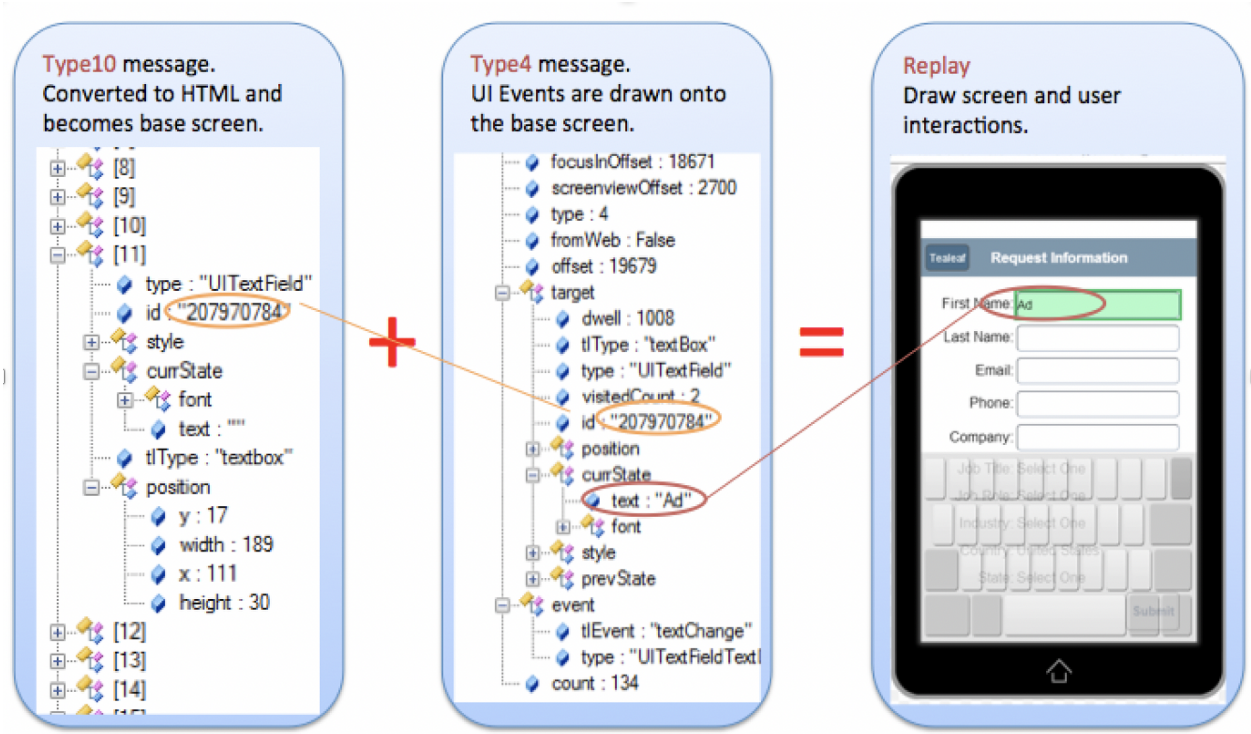


Figure 7.2. Native Replay messages

generated in BBR from the Type 10 message serves as a canvas that the Type 4 events are drawn on. The example of a Type 10 message and a Type 4 message is in **Figure 7.2**.

Type 10 JSON messages will be converted into HTML to make them viewable. The conversion from JSON data to HTML is controlled by Tealeaf Template Engine. It is essentially a set of template scripts. Tealeaf Template Engine translates the root of each Type 10 message into a template (Type10.html). In this process, the root template will hand off sub JSON elements to sub-templates, each of which builds HTML fragments and then combine them together. In general, templates will combine static HTML text with JSON values.

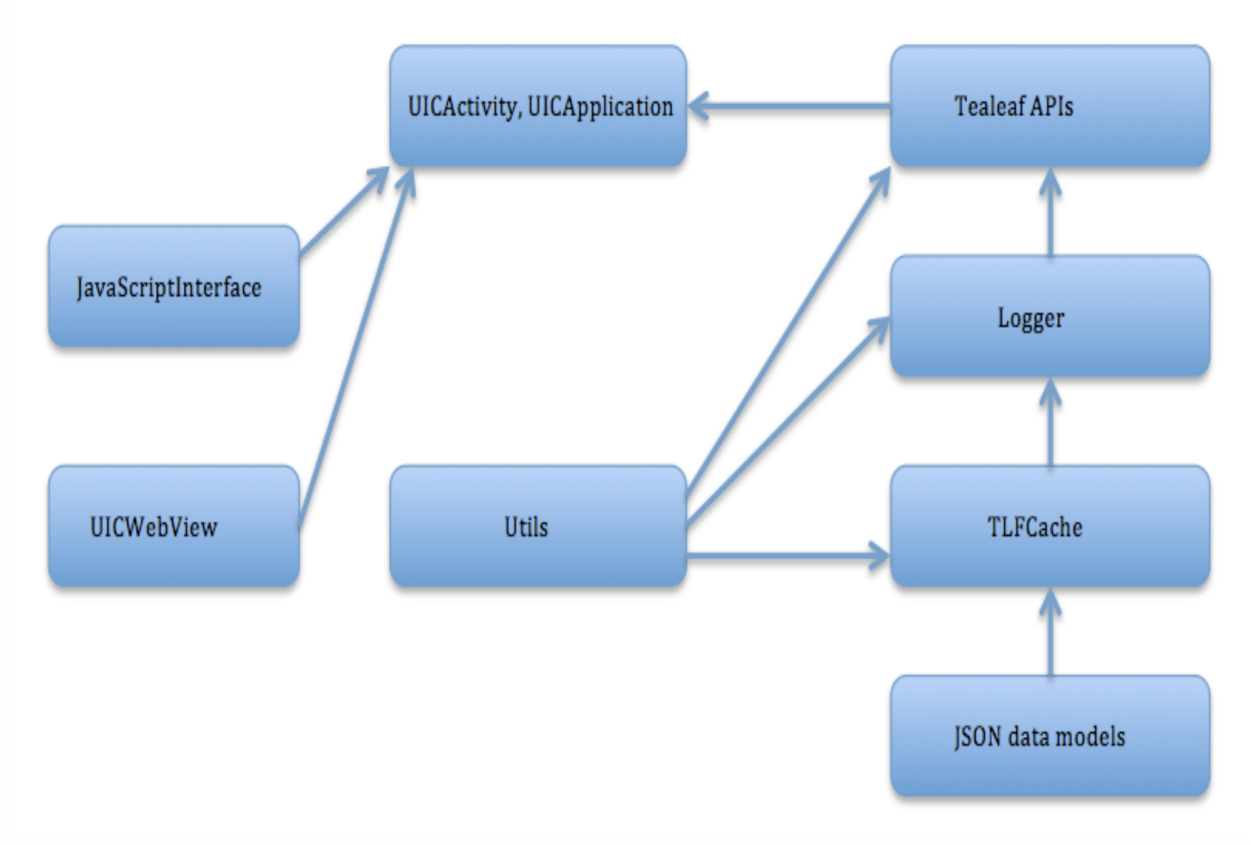


Figure 7.3. TealeafCxMobile SDK architecture

7.3 Native Replay iOS and Android

IBM Tealeaf CxMobile currently offers SDKs on iOS and Android platforms. The architecture of Tealeaf CxMobile SDKs is shown in **Figure 7.3**

As **Figure 7.3** shows, UICApplication provides entrance to SDK functions. Tealeaf APIs provide developer-oriented function calls. Logger is the internal implementation of most logging functions. TLFCache manages JSON data queue. JSON data models provide JSON formats representing each component. Utils provide specific function classes used by Tealeaf APIs, Logger, and TLFCache.

Due to the different lifecycles between the iOS and Android platforms, the instrumentation varies a little bit.

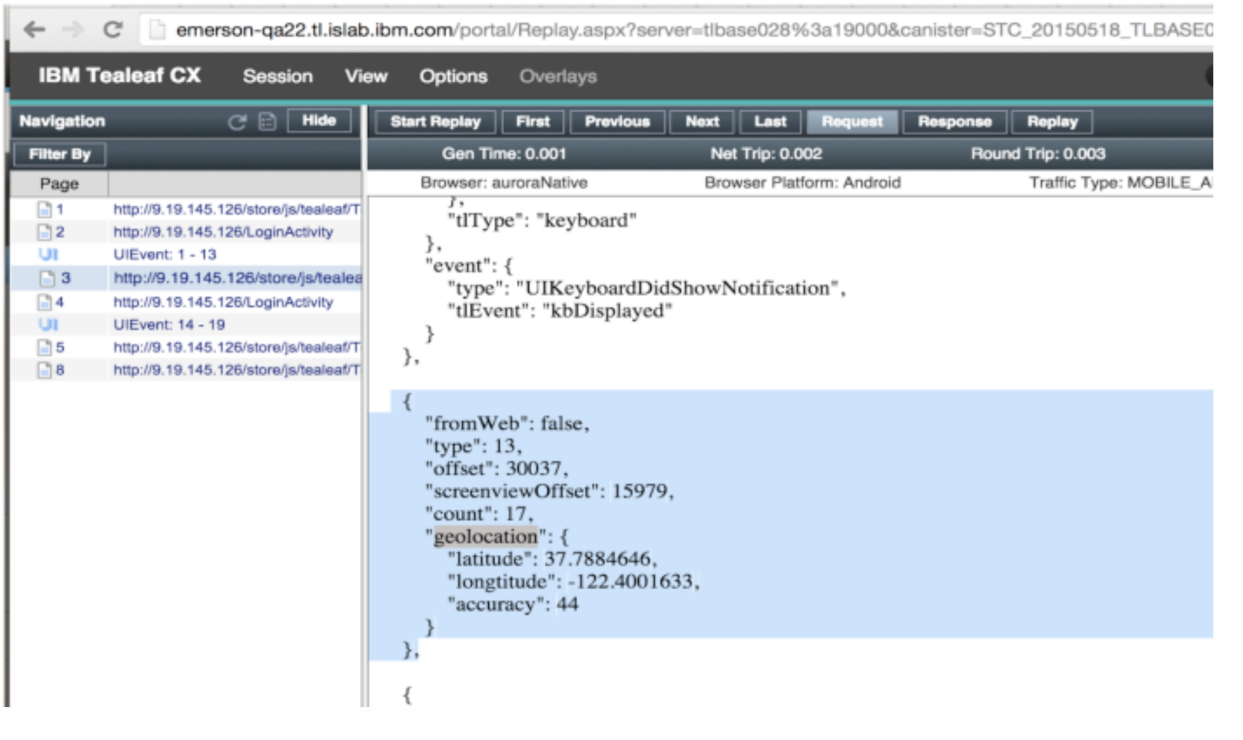


Figure 7.4. Geo-Location JSON Data

On both iOS and Android platforms, UICApplication needs to be set up at the app's entrance to monitor the global interactions. In iOS, a View can be considered a page that is displayed on a mobile device. By default, we would like to record a View being displayed. We can do that by placing the API logScreenLayout inside a ViewController's viewDidAppear function. In Android, an Activity can be considered a page that is displayed on a mobile device. To record an activity being displayed, we can place the API logScreenLayout inside an Activity's OnCreate method. Since iOS provides a mechanism called Swizzling, which essentially meaning that Objective-C runtime allows changing binding between selector (method) declarations and implementations. In iOS SDK, we provide an auto instrumentation option that the developer does not need to inject logScreenLayout call to every ViewController, since we replaced the iOS original methods with our logging functions.

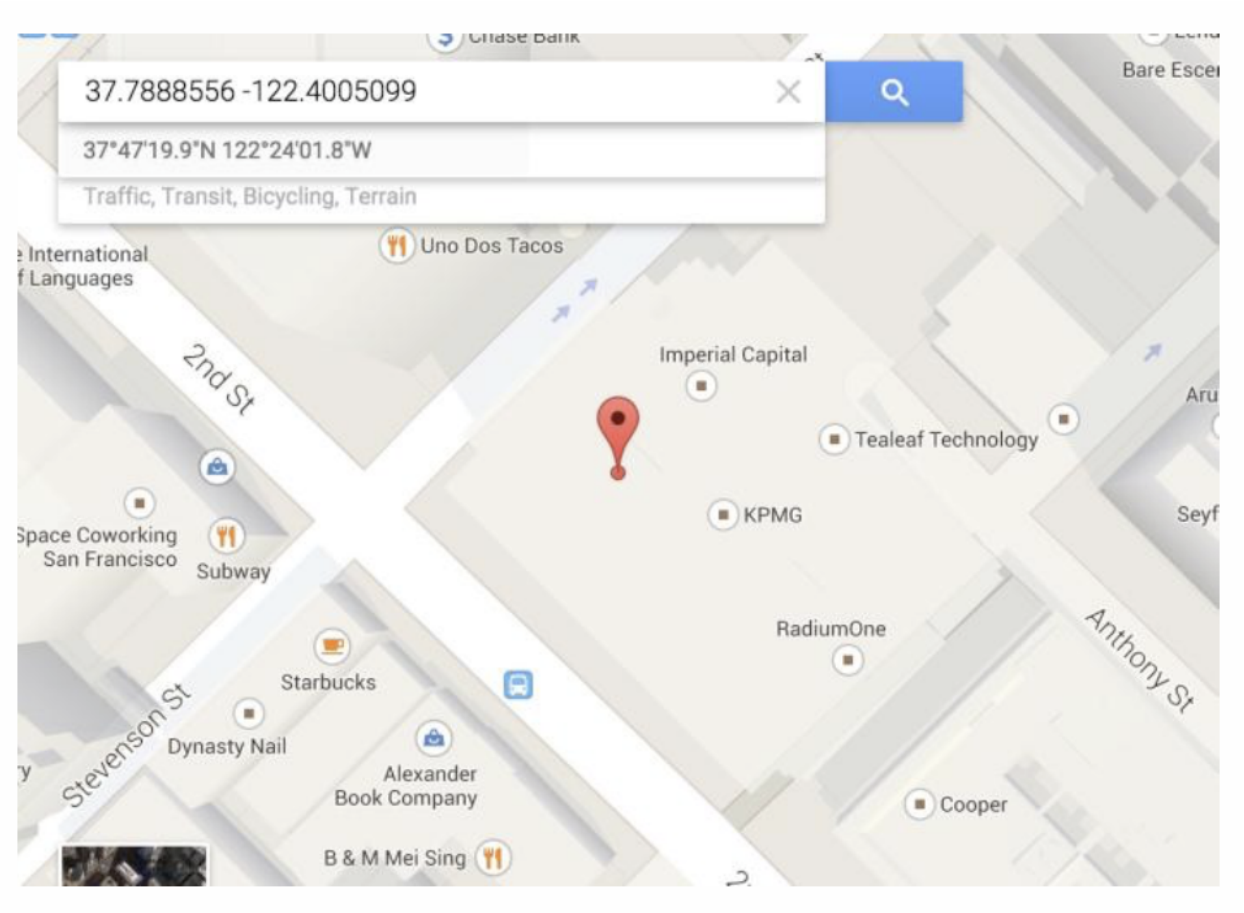


Figure 7.5. Tealeaf Office West Corner

7.4 Geo-Location Logging Implementation

As location-based services have been increased with the mobile trend, customer attitude is shifting towards being more comfortable with sharing locations. So it is increasingly important to deliver excellent mobile experiences with Location and contextual awareness. In this session, we will explain how to add Geo-Location Logging module to Tealeaf CxMobile.

First, let's consider several scenarios that Geo-Location is being considered important:

1. Airline companies want to know if users access the application when they are in the airport because they could enable an easier way to access flight status.

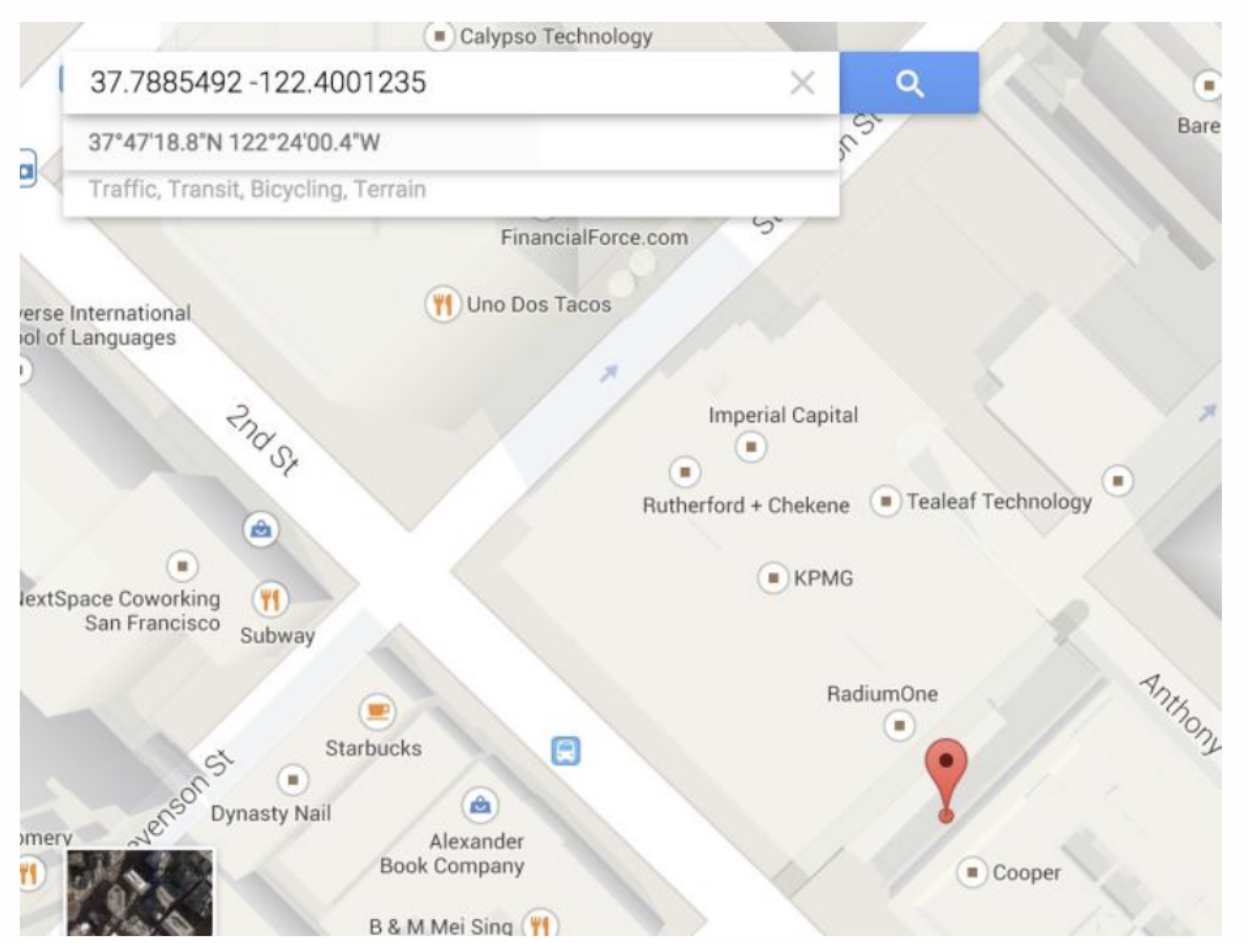


Figure 7.6. Tealeaf Office East Corner

2. Hotel companies want to know whether users reserved a hotel room from their home, or the airport, or somewhere else.

3. Banks want to know how many customers use the application when they are in the bank as opposed to when they are not.

4. E-Commerce companies want to know where customers are when they access the application and complete their transactions.

Based on these function requirements, our solution should be able to make the following points:

1. CxMobile iOS and Android SDKs should be able to record the Geo-Location information of the user when the device is granted with geo permission: latitude, longitude, accuracy.

2. CxMobile SDKs should be able to detect when the device is disabled with Geo-Location.

3. Geo-Location functions should be configurable with some flexibility.

With the approach proposed in [103], we implemented the logGeoiocation API. It can be placed at any callback function of the user interactions. We also provide more configuration options for developers to enable or disable Geo-Location Logging and set time out. Geo-Location information is placed as a Type 13 message. When the device is granted with Geo-Location permission, the data is like **Figure 7.4**.

We tested the accuracy of the Geo-Location logging functions in our San Francisco Office. Normally the accuracy is within 50 meters. We projected the coordinates on Google Map [117]. You can see it can tell the different office corners as **Figure 7.5** and **Figure 7.6** shows.

7.5 Conclusion and Future Work

This paper introduces IBM Tealeaf CxMobile and its Native Mobile Replay technology. With our unique Native Mobile Replay technology, Tealeaf CxMobile can help customers to replay a user's journey through a native iOS or Android app in the way of what you see is what you get rather than mere data analysis. As Tealeaf CxMobile being adopted by most of the banks, insurance companies, airplane companies in North America and Europe, we proved that including visual replay functions into data analysis tools is extraordinarily important. In the paper, we illustrate the principles of Tealeaf Native Mobile Replay and how it works on iOS and Android platforms. We also give an example of applying the Modularization SDK methodology [103] to integrate Geo-Location logging functions into Tealeaf CxMobile. In the future, we still need to provide the method of building a better reporting system for Geo-Location Logging.

CHAPTER 8

A FAST APPROACH TO ENABLE MOBILE APPS WITH GEO LOCATION LOGGING AND REPORTING[1]

8.1 Introduction

With the rapid development of mobile and cloud technology, the cost of collecting and studying human behavior data is largely reduced. App vendors or researchers are interested in how the apps are being used, how much time the users stay on a certain mobile view, and if exceptions are happening during a user session, etc. The answers to all these questions can be found in the data collected in real-time when the user is running the app, and in the analytics of the data.

A mobile app can send back to the backend management portal most of sensor-related data, device status information, UI changing data, and so on depending on the developers' needs. Developers can just go ahead implementing the functions they want. This is called Ad-hoc Function Implementation. Moreover, there are plenty of free or paid analytics tools on the market. Those analytics tools are designed to be integrated with a mobile app easily and provide specialized insights. For example, Google Analytics [114] can give you insight about who uses your apps now, on what devices, and where they come from. It also provides event tracking and user flow visualization; Adobe Analytics [118] can give you insights about user engagement within your mobile app, including how frequently consumer launch the channel, whether they make purchases from it, and it provides better campaign analysis and dashboard reporting; IBM Tealeaf [102] can capture all the UI information and user gestures such as tapping, swiping, pinching, zooming, scrolling and device rotation, etc. It

[1]© 2015 IEEE. Reprinted, with permission, from Yanke Hu, Xiwang Zhang, Liangmin Hu, A Fast Approach to Enable Mobile Apps with Geo-Location Logging and Reporting, IEEE International Conference on Geoinformatics, June 2015, Wuhan, China

also provides real-time replay of what exactly users see and operate on mobile devices. This is called Vendor SDK Implementation.

Both Ad-hoc Function Implementation and Vendor SDK Implementation have some advantages, but neither is perfect. For Ad-hoc Function Implementation, developers have more freedom to customize their function modules, and they also have full control of the apps' reporting backend. But if an organization maintains several apps, and it wants to monitor the same sensor information trend on all the apps, it will be chaotic and redundant to implement this sensor data collecting function for every individual app. For the Vendor SDK Implementation, it has the benefit of easy instrumentation, but an SDK from a certain vendor may provide a lot of unnecessary functions, and it may not provide the exact function you need. You don't have the full control and customization of the backend reporting. And it may not be free.

In this paper, we propose an approach of Modularization SDK Implementation and Cloud reporting architecture, to enable Mobile apps with sensor data logging and reporting by inserting one single line of code. Compared to Ad-hoc Function Implementation and Vendor SDK Implementation, Modularization SDK Implementation encapsulates the sensor information logging component into developers' internal SDKs. It can be reused in any app being maintained by the same organization. The backend reporting is also fully under control and customizable. We will use Geo-Location logging as an example to illustrate how this approach works on iOS and Android platforms, which covers more than 95% of mobile platform shares globally according to IDC by the end of 2014 [119].

The rest of the paper is organized as the following. Section 2 illustrates how Geo-Location Modularization SDK Implementation works on iOS and Android platforms. Section 3 introduces the commonly used backend reporting architecture. Section 4 compares our proposed approach versus the traditional approach. Section 5 concludes the paper.

8.2 Geo-Location Modularization SDK Implementation on Mobile Platforms

One of the most important differences between web services and mobile apps is that the user location is changing much more frequently during a user session. The user location data can bring a lot of insights about the users' habit, interest, etc, thus it can be used in a variety of scenarios, like health monitoring, point of interest exploring, mobile social network, etc. Since the initial release of iOS and Android system, both platforms provide very easy to use location-based framework. This greatly stimulates the prevalence of Location-Based Service (LBS) in the mobile era.

Location-based framework provides the APIs for developers to get location status and device heading information. It utilizes a combination of cellular, WiFi, Bluetooth, and GPS to get your location with your permission. To do Location-Based Service analytics, the mobile app must have the module to collect the Location data containing longitude, latitude, accuracy information of the user device, and send the data back to the backend portal.

8.2.1 iOS Geo-Location Module Implementation

The Location-based framework on iOS platform is Core Location framework, which provides a suite of Objective-C interfaces to get users' location and heading information. The basic unit of an iOS app is a View Controller, which is very similar to a single web page of a web service. When an iOS app starts, the UIApplication object sets up the main loop at launch time to handle the updates of those View Controller classes. Every iOS app has a global delegate object to handle the app's state transitions, like app start, becoming active, terminated, etc. In the global delegate class, method didFinishLauchingWithOptions [120] is the place where to put the one line instrumentation code if the developer wants to log Geo-Location when the app launch. Other than logging Geo-Location when the app starts, the developer can also choose to log Geo-Location when a specific View Controller begins to show up. This can be simply done by putting the one-line instrumentation code inside

viewDidAppear method of a View Controller class [121], which is the starting point of the life cycle of a View Controller instance.

Once you have the entry point of the iOS Geo-Location Modularization SDK, the next thing you need to have is to create an instance of the CLLocationManager class, and set values for its desiredAccuracy and distanceFilter properties. To get the location updating notifications, the entry class must be assigned with a CLLocationManagerDelegate protocol. Firing the startMonitoringSignificantLocationChanges method of your CLLocationManager instance, and you will get the updated Location data from the didUpdateLocations method of the CLLocationManager instance [122]. The Location data can be sent back to the backend portal by wrapping a flushing queue inside a dispatch_async function [123].

8.2.2 Android Geo-Location Module Implementation

The Location-based framework on Android platform is the android.location package, which provides the classes that define Android location-based services. The basic unit of an Android app is an Activity [124]. Compared to the View Controller in iOS, the Activity is a more independent running instance. Though most of the Android apps have an Android Application instance watching the apps' life cycle, it's not indispensable. The simplest Android app can just have one Activity without any Application instance. All the Android Activities must be registered in the AndroidManifest.xml configuration file. The entry Activity of the app is also specified in AndroidManifest.xml by a MAIN action tag and LAUNCHER category tag. When an Android app starts, the app will first go to the onResume callback function of the entry Activity. This is where to put the one line instrumentation code if the developer wants to log Geo-Location when the app launches. Other than logging Geo-Locaion when app starts, the developer can also choose to log Geo-Location when a specific Activity launches. This can be simply done by inserting the one line instrumentation code inside the onResume callback function of that specific Activity class.

Table 8.1. iOS & Android Implementation Difference

	iOS	Android
Geo-Location Logging	CLLocationManagerDelegate, CLLocationManager	LocationManager, LocationListener
App Launch	didFinishLaunchingWithOptions	Application.onCreate
ViewAppear	UIViewController:viewDidAppear	Activity.onResume
Send Data	dispatch_async	AsyncTask

Once you have the entry point of the Android Geo-Location Modularization SDK, the next thing you need to do is to create an instance of the LocationManager class citegeo10. You don't need to define any delegate for a specific Activity class like what you have to do for an iOS View Controller, because this LocationManager instance already get the app context from Context.LOCATION_SERVICE parameter. The next step is to declare an instance of LocationListener [126], and use it as the parameter of the requestLocationUpdates call of the LocationManager instance. Now you should be able to get the updated Location data from the getLastKnownLocation call of the LocationManager instance. The Location data can be sent back to the backend portal by implementing an AsyncTask class [127].

On Android platform, using Google Location Services API can give a more powerful way of handling location providers, user movement, and location accuracy than android.location package, but in most cases, the android.location package is good enough and much easier to use. The android.location package is built in with Android SDK, and doesn't require additional library jar file.

Table 8.1 displays the different terminologies of implementing geo-location modules on iOS and Android platforms.

8.3 Cloud Reporting Architecture

The traditional Analytics tools usually provide an on-premise reporting backend. When you buy the software suite from the vendor, the vendor will have a professional support team

helping you setup and configure the on-premise server. This on-premise approach is still very important to those services that are dealing with sensitive data, need high security and require full control of the systems. But as the price of hardware and bandwidth dropping fast, the cloud reporting backend becomes more and more popular with small businesses. Cloud backend is more cost-effective when your business coverage is still small, and you don't need to worry too much to invest huge dollars in expensive hardware and IT staff. There are several popular cloud reporting server options on the market, like Amazon Web Services (AWS) [128], Google App Engine (GAE) [129], Microsoft Azure, IBM SoftLayer, etc. Among them, we will discuss the most important two – AWS and GAE, and compare their advantages and disadvantages.

8.3.1 Amazon Web Services

Amazon Web Services (AWS) is a collection of remote computing services offered by Amazon. AWS mainly offers the Infrastructure-as-a-Service (IaaS). IaaS works in such a way that organizations outsource servers, storage, networking, etc to IaaS providers, and pay as how much they use those resources like we pay electricity bill monthly. This makes it very easy for start-up or experimental projects, avoiding investing expensive in-house hardware at one time, but getting more customers and users first. IaaS providers normally provide virtual machines, networking and hardware. IaaS providers also handle system maintenance and backup. IaaS users have the freedom to configure applications, operating systems, etc.

8.3.2 Google App Engine

Google App Engine (GAE) is a Platform-as-a-Service (PaaS) platform for hosting web applications. You are running your web applications on a Google defined platform. You don't know what exact operating systems your applications are running on, and your applications normally run across multiple servers. The most appealing feature of GAE is that it scales

automatically for web applications as the number of requests increases. When the request demands grow, you don't need to manually add more machines or bandwidth, GAE platform will automatically allocate more resources for your application. The benefit of GAE is that it removes many system administration and configuration works when the request demands are continuously growing, but the restriction of GAE is also obvious. It can only run a limited variety of applications designed for GAE infrastructure. Currently, GAE mainly supports Python and JAVA, and their related frameworks. While AWS provides more freedom of the customization of the operating systems, GAE provides better scalability and reliability.

8.4 Discussion

As the technical aspects of Modularization SDK Implementation discussed in the former sessions, we want to conclude a summary of the difference and benefits of our proposed approach according to the traditional approach.

Compared to the traditional approach in **Figure 8.1**, our approach provides 3 advantages:

1. Reduce the duplicate geo-location implementations on client-side mobile apps. This makes it much easier when the vendor updating geo-location functions in multiple mobile apps.

2. Centralize the server reporting in the cloud. This brings more scalability, reliability, better reporting, and easier for starting up and testing new ideas.

3. Unifies the client server communications on different mobile platforms.

Our approach is illustrated in **Figure 8.2**.

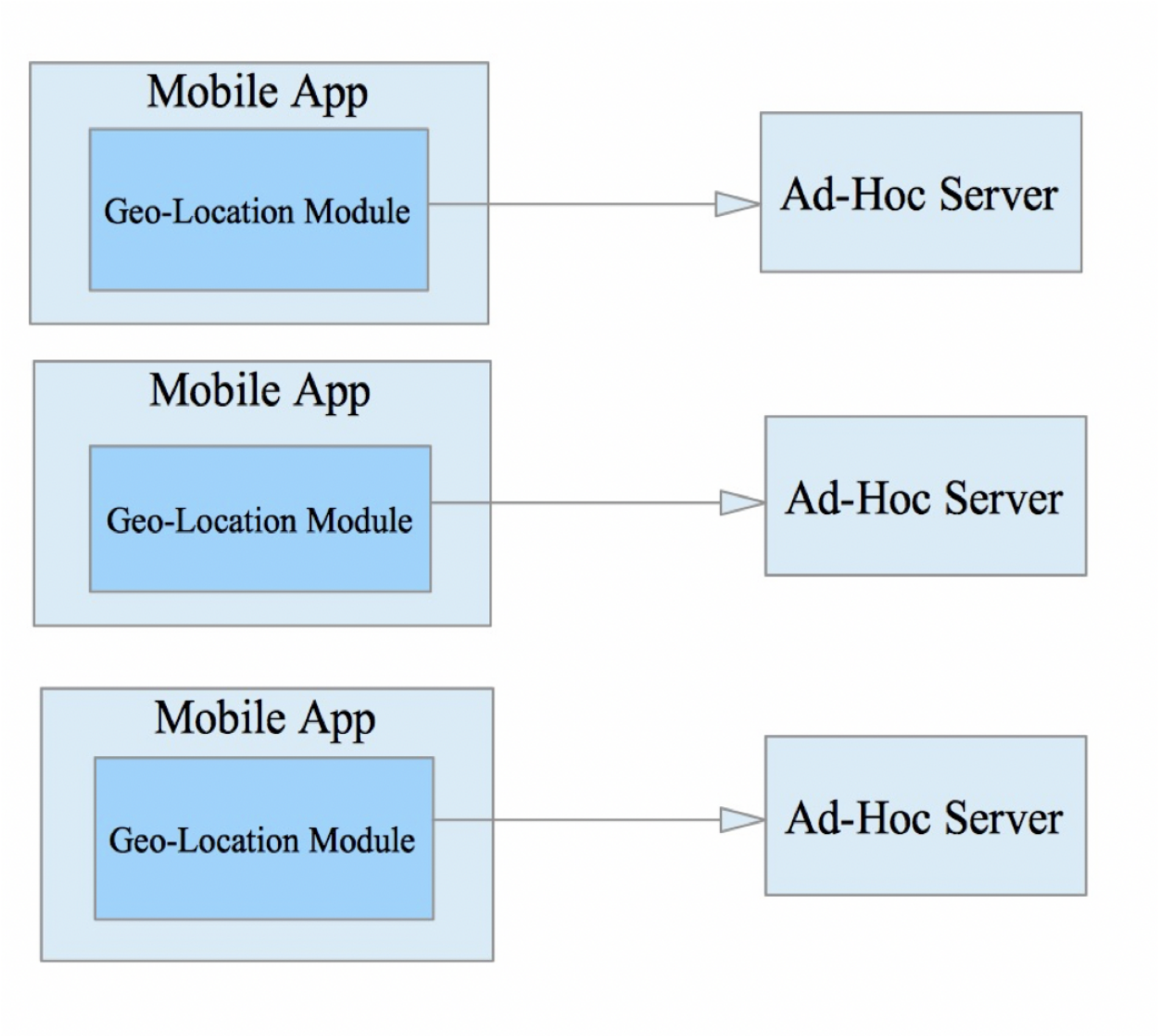


Figure 8.1. Traditional Geo-Location Logging Implementation

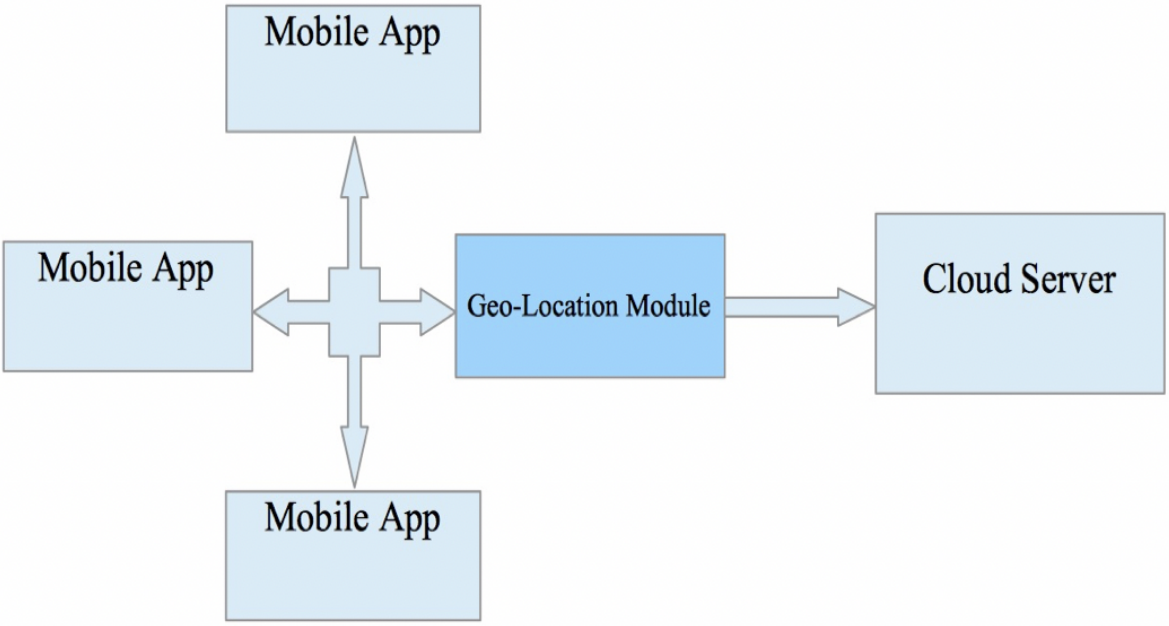


Figure 8.2. Improved Geo-Location Logging Implementation

8.5 Conclusion

This paper proposes an approach of Modularization SDK Implementation and Cloud reporting architecture, to enable Mobile apps with sensor information logging and reporting by inserting one single line of code. We also use Geo-Location logging as an example and explains how it works on iOS and Android platforms. Our approach can bring several advantages including removing redundant implementation, bringing more scalability, etc compared to the traditional approach. In the future, we will provide more implementation samples based on this approach.

CHAPTER 9

CONCLUSION

This dissertation presents the research results of several interesting problems in developing the next generation healthcare information platform, that can help people better understand their health status, educate themselves healthcare-related information, and connect them with hospitals and doctors. Correspondingly, it will help US health institutes better track ongoing COVID-19 trend and defend against future epidemics.

Despite the leading position of US fundamental medical research, the US healthcare information platform development is lagging. Our results have not only contributed to the research progress of healthcare analytics but also brought practical values to the development of the future healthcare information platform.

Our contributions are summarized as the following:

1. Physiological signals modeling and Faster convolutional feature engineering method that can greatly speed up training and inference of modeling vital signs time series data.

2. Transfer learning of deep pre-trained language models into healthcare domains and pruning and knowledge distillation methods that greatly shrink the deep learning models and make them possible or easier to be deployed on mobile devices.

3. Mobile deployment of deep learning models and Passive mobile capture techniques to monitor user health status silently and comprehensively compared to the traditional survey method.

4. Seamless adoption of the most recent research progress in deep learning and make it into practical healthcare solutions.

In the future, we will continue making efforts for developing more mature healthcare information systems and applications that can help people better understand their health status, answer their health-related questions, and connect them with hospitals and doctors, thus eventually defend the ongoing COVID-19 and future epidemics.

REFERENCES

[1] Ann Kutney-Lee, Douglas M. Sloane, Kathryn H. Bowles, Lawton R. Burns, Linda H. Aiken. Electronic Health Record Adoption and Nurse Reports of Usability and Quality of Care: The Role of Work Environment. In Applied Clinical Informatics 10(01):129-139. January, 2019

[2] Y. LeCun, Y. Bengio, and G. Hinton. Deep Learning. Nature, 521(7553):436–444, 2015.

[3] A. Krizhevsky, I. Sutskever, and G. Hinton.Imagenet classification with deep convolutional neural networks. In NIPS, 2012

[4] Ferrucci, D., Levas, A., Bagchi, S., Gondek, D.,Mueller, E. T. Watson: beyond jeopardy! Artificial Intelligence 199, 93–105, 2013

[5] IBM. 2011. IBM Watson Health

[6] Amazon. 2018. Amazon Comprehend Medical.

[7] Amazon. 2019. Amazon Care

[8] Google. System and Method for Predicting and Summarizing Medical Events from Electronic Health Records. US Patent. 2019/0034591 A1

[9] Alphabet Inc. 2008. Google Health

[10] Alphabet Inc. 2015. Verily

[11] Alphabet Inc. 2013. Calico

[12] U. R. Acharya, S. L. Oh, Y. Hagiwara, J. H. Tan, H. Adeli. Deep convolutional neural network for the automated detection and diagnosis of seizure using eeg signals. Computers in Biology and Medicine, 2017

[13] U. R. Acharya, H. Fujita, O. S. Lih, Y. Hagiwara, J. H. Tan, M. Adam. Automated detection of arrhythmias using different intervals of tachycardia ecg segments with convolutional neural network. Information sciences, 2017

[14] S. P. Shashikumar, A. J. Shah, Q. Li, G. D. Clifford, S. Nemati. A deep learning approach to monitoring and detecting atrial fibrillation using wearable technology. IEEE EMBS International Conference Biomedical & Health Informatics, 2017

[15] M. Cheng, W. J. Sori, F. Jiang, A. Khan, S. Liu. Recurrent neural network based classification of ecg signal features for obstruction of sleep apnea detection. IEEE International Conference on Computational Science and Engineering (CSE) and Embedded and Ubiquitous Computing (EUC), 2017

[16] Yanke Hu, Wangpeng An, Raj Subramanian, Na Zhao, Yang Gu, Weili Wu. Faster Clinical Time Series Classification with Filter based Feature Engineering Tree Boosting Methods. AAAI Health Intelligence Workshop, 2020

[17] Yanke Hu, Raj Subramanian, Wangpeng An, Na Zhao, Weili Wu. Faster Healthcare Time Series Classification for Boosting Mortality Early Warning System. International Conference on Intelligent Robots and Systems, 2020

[18] https://www.minnpost.com/second-opinion/2020/07/cdc-covid-19-cases-in-u-s-are-10-times-higher-than-reported-including-in-minnesota/

[19] https://mitsloan.mit.edu/ideas-made-to-matter/covid-19-cases-are-12-times-higher-reported

[20] https://khn.org/morning-breakout/true-number-of-covid-cases-could-be-10-times-higher/

[21] https://www.statnews.com/2020/07/21/cdc-study-actual-covid-19-cases/

[22] https://www.usatoday.com/story/news/nation/2020/08/10/covid-19-proms-parties-michigan-teens-students-testing-positive/3333573001/

[23] https://www.latimes.com/california/story/2020-08-17/covid-19-infections-los-angeles-party-crackdown

[24] https://www.bostonglobe.com/2020/08/17/business/deborah-birx-says-parties-are-fueling-recent-coronavirus-spread/

[25] D. Du and F. Hwang. Optimal consecutive-2-out-of-n systems. Mathematics of Operations Research, 11(1):187–191, 1986.

[26] D.-Z. Du and X.-S. Zhang. Global convergence of rosen's gradient projection method. Mathematical Programming, 44(1-3):357–366, 1989.

[27] Yanke Hu. Tealeaf CxMobile - Replaying Real-Time Customer Experience. Geoinformatics, 2016

[28] https://techcrunch.com/2012/05/02/ibm-acquires-tealeaf-to-add-customer-buying-analytics-to-smarter-commerce-products/

[29] https://www.fintopia.tech/

[30] https://www.aifi.io/

[31] https://www.apple.com/ios/health/

[32] https://www.google.com/fit/

[33] https://ouraring.com/tags-covid19-flu-cold

[34] https://med.stanford.edu/snyderlab/news/20202-covid-19-research.html

[35] https://blog.fitbit.com/early-findings-covid-19-study/

[36] https://www.frontiersin.org/articles/10.3389/fdgth.2020.00008/full

[37] S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997

[38] Lipton, Z. C.; Kale, D. C.; Elkan, C.; and Wetzell, R. Learning to diagnose with lstm recurrent neural networks.arXiv preprint arXiv:1511.03677, 2015

[39] H. Song, D. Rajan, J. J. Thiagarajan, and A. Spanias. Attend and Diagnose: Clinical Time Series Analysis using Attention Models, in AAAI 2018

[40] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. Advances in Neural Information Processing Systems, 3149 - 3157, 2017

[41] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In EMNLP 2014.

[42] Harutyunyan, H.; Khachatrian, H.; Kale, D. C.; and Galstyan, A. Multitask learning and benchmarking with clinical time series data. arXiv preprint arXiv:1703.07771, 2017

[43] Alistair E.W.Johnson, Tom J. Pollard, Lu Shen, Li-wei H. Lehman, Mengling Feng, Mohammad Ghassemi,Benjamin Moody, Peter Szolovits, Leo Anthony Celi,Roger G.Mark.Mimic-iii, a freely accessible critical care database. Sci.data 3, 2016

[44] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, pages 785–794, 2016

[45] J. Bergstra and Y. Bengio. 2012. Random search for hyperparameter optimization. JMLR, 13(1):281–305.

[46] J. Bergstra, D. Yamins, and D.D. Cox. 2013. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In Proc. of ICML, pages 115–123.

[47] https://keras.io/examples/cifar10_cnn/

[48] Mark Sandler,Andrew Howard,Menglong Zhu,Andrey Zhmoginov,Liang-Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. arXiv preprint arXiv: 1801.04381,2018

[49] Forrest N. Iandola,Song Han, Matthew W. Moskewicz, Khalid Ashraf,William J. Dally,Kurt Keutzer.SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and ¡0.5MB model size.arXiv preprint arXiv: 1602.07360,2016

[50] Taylor RA, Pare JR, Venkatesh AK, et al. 2016. Prediction of In-hospital Mortality in Emergency Department Patients With Sepsis: A Local Big Data-Driven, Machine Learning Approach. Acad Emerg Med;23:269–278.

[51] Fonarow GC, Adams KF, Abraham WT, Yancy CW, Boscardin WJ; ADHERE Scientific Advisory Committee, Study Group, and Investigators. 2005. Risk stratification for in-hospital mortality in acutely decompensated heart failure: classification and regression tree analysis.JAMA. 293:572–580. doi: 10.1001/jama.293.5.572.

[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in neural information processing systems(NeurIPS), pp. 5998–6008, 2017.

[53] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pretraining of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018

[54] Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. Improving language understanding by generative pre-training, 2018

[55] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. arXiv preprint arXiv:1906.08237, 2019.

[56] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim,Donghyeon Kim, Sunkyu Kim, Chan Ho So,and Jaewoo Kang. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. In Bioinformatics, 2019

[57] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108, 2019

[58] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu.TinyBERT: Distilling BERT for natural language understanding.arXiv preprint arXiv:1909.10351,2019

[59] Sun, Z., Yu, H.,Song, X., Liu, R., Yang, Y., and Zhou, D. MobileBERT: a compact task-agnostic BERT for resource-limited devices. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020

[60] C. Bucilua, R. Caruana, and A. Niculescu-Mizil. Model compression. In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'06, pages 535–541, New York, NY, USA, 2006. ACM.

[61] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531, 2015.

[62] Gregor Urban, Krzysztof J Geras, Samira Ebrahimi Kahou, Ozlem Aslan, Shengjie Wang, Rich Caruana, Abdelrahman Mohamed, Matthai Philipose, and Matt Richardson. Do deep convolutional nets really need to be deep (or even convolutional)? In Proc. Int. Conf. Learn. Representations, 2016.

[63] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In Proc. Advances in Neural Inf. Process. Syst., pages 2654–2662, 2014.

[64] Dogan,R.I. et al. NCBI disease corpus: a resource for disease name recognition and concept normalization. J. Biomed. Inform., 47, 1–10, 2014

[65] Uzuner,O. et al. 2010 i2b2/VA challenge on concepts, assertions, and relations in clinical text. J. Am. Med. Inform. Assoc., 18, 552–556, 2011

[66] Li,J. et al.Biocreative V CDR task corpus: a resource for chemical dis- ease relation extraction. Database, 2016.

[67] Krallinger,M. et al. The chemdner corpus of chemicals and drugs and its annotation principles. J. Cheminform., 2015

[68] Smith,L. et al. Overview of biocreative ii gene mention recognition. Genome Biol., 2008

[69] Kim,J.-D. et al. Introduction to the bio-entity recognition task at JNLPBA. In: Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP), Geneva, Switzerland. pp. 73–78. COLING. https://www.aclweb.org/anthology/W04-1213, 2004

[70] Gerner,M. et al. Linnaeus: a species name identification system for biomedical literature. BMC Bioinformatics, 11, 85, 2010

[71] Pafilis,E. et al. The species and organisms resources for fast and accurate identification of taxonomic names in text. PLoS One, 8, e65390, 2013

[72] Bravo,A. et al. Extraction of relations between genes and diseases from text and large-scale data analysis: implications for translational research. BMC Bioinformatics, 16, 55, 2015

[73] Van Mulligen, E.M. et al. The EU-ADR corpus: annotated drugs, diseases, targets, and their relationships. J. Biomed. Inform., 45, 879–884, 2012

[74] Krallinger,M. et al. Overview of the BioCreative VI chemical-protein interaction track. In: Proceedings of the BioCreative VI Workshop, Bethesda, MD, USA, pp. 141–146. https://academic.oup.com/database/art icle/doi/10.1093/database/bay073/5055578 , 2017

[75] Tsatsaronis,G. et al. An overview of the BIOASQ large-scale biomedical semantic indexing and question answering competition. BMC Bioinformatics, 16, 138, 2015

[76] https://github.com/naver/biobert-pretrained

[77] Google. 2002. https://news.google.com.

[78] ByteDance. 2012. https://www.toutiao.com

[79] Carreira, R., Crato, J. M., Gonçalves, D., Jorge, J. A. 2004.Evaluating adaptive user profiles for news classification, Proceedings of the 9th international conference on Intelligent user interfaces.

[80] Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D. and Sartin, M. 1999. Combining Content-Based and Collaborative Filters in an Online Newspaper. In Proceedings of ACM SIGIR Workshop on Recommender Systems.

[81] Yijia Zhang, Qingyu Chen, Zhihao Yang, Hongfei Lin, and Zhiyong Lu. 2019. BioWordVec, improving biomedical word embeddings with subword information and mesh. Scientific data, 6:52.

[82] Elastic Search, 2010, https://www.elastic.co/

[83] Hamborg, Felix and Meuschke, Norman and Breitinger, Corinna and Gipp, Bela.2017.news-please: A Generic News Crawler and Extractor.Proceedings of the 15th International Symposium of Information Science

[84] Carbonell, J., and Goldstein, J. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval, 335–336. ACM, 1998

[85] Radev, D., and Erkan, G. Lexrank: Graph-based lexical centrality as salience in text summarization. Journal of Artificial Intelligence Research 457–479, 2004

[86] Kageback, M.; Mogren, O.; Tahmasebi, N.; and Dubhashi, D. Extractive summarization using continuous vector space models. 31–39, 2014

[87] Ganesan, K.; Zhai, C.; and Han, J. Opinosis: a graph based approach to abstractive summarization of highly redundant opinions. In Proceedings of the 23rd international conference on computational linguistics, 340–348. Association for Computational Linguistics, 2010

[88] Yin, W., and Pei, Y. Optimizing sentence modeling and selection for document summarization. In Proceedings of the 24th International Conference on Artificial Intelligence, 1383–1389. AAAI Press, 2015

[89] Cao, Z.; Li, W.; Li, S.; and Wei, F. Attsum: Joint learning of focusing and summarization with neural attention. arXiv preprint arXiv:1604.00125, 2016

[90] Cheng, J., and Lapata, M. Neural summarization by extracting sentences and words. 54th Annual Meeting of the Association for Computational Linguistics, 2017

[91] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In Proceedings of Neural Information Processing Systems (NIPS), 2015

[92] Chin-Yew Lin. ROUGE: a package for automatic evaluation of summaries. In Proceedings of ACL Workshop on Text Summarization Branches Out, 2004

[93] Abigail See, Peter J Liu, and Christopher D Manning.Get to the point: Summarization with pointer-generator networks. arXiv preprint arXiv:1704.04368, 2017

[94] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. arXiv preprint arXiv:1705.04304, 2017

[95] Ramakanth Pasunuru and Mohit Bansal. Multireward reinforced summarization with saliency and entailment. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), volume 2, pages 646–653, 2018

[96] Chenliang Li, Weiran Xu, Si Li, and Sheng Gao. Guiding generation for abstractive text summarization based on key information guide network. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), volume 2, pages 55–60, 2018

[97] Wan-Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. A unified model for extractive and abstractive summarization using inconsistency loss. arXiv preprint arXiv:1805.06266, 2018

[98] Yen-Chun Chen and Mohit Bansal.Fast abstractive summarization with reinforce-selected sentence rewriting. arXiv preprint arXiv:1805.11080, 2018

[99] Sebastian Gehrmann, Yuntian Deng, and Alexander M Rush. Bottom-up abstractive summarization. arXiv preprint arXiv:1808.10792, 2018

[100] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents. In AAAI, pp. 3075–3081. AAAI Press, 2017.

[101] Natalie Schluter. The limits of automatic summarization according to rouge. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Short Papers. Valencia, Spain, pages 41–45, 2017

[102] https://help.goacoustic.com/hc/en-us

[103] Yanke Hu, Xiwang Zhang, Liangmin Hu, A fast approach to enable mobile apps with Geo-Location logging and reporting. Geoinformatics 2015

[104] Ohbyung Kwon, Namyeon Lee, Bongsik Shin, "Data quality management, data usage experience and acquisition intention of big data analytics". International Journal of Information Management, 2014

[105] Jefferey Spiess, Yves Tjoens, Raluca Dragnea, Peter Spencer, Laurent Philippart, "Using Big Data to Improve Customer Experience and Business Performance". Bell Labs Technical Journal, 2014

[106] Sean Kandel, Andreas Paepcke, Joseph M Hellerstein, Jeffrey Heer, "Enterprise Data Analysis and Visualization: An Interview Study", IEEE Transactions on Visualization andComputer Graphics, 2012

[107] https://en.wikipedia.org/wiki/Growth

[108] Stijn Viaene, Annabel Van Den Bunder, 'The secrets to managing business analytics projects", MIT Sloan Management Review, 2011

[109] Tammo H A Bijmolt, Peter S H Leeflang, Frank Block, Maik Eisenbeiss, "Analytics for Customer Engagement", Journal of Service Research, 2010

[110] Danyel Fisher, Robert Deline, Mary Czerwinski, Steven M Drucker, "Interactions with Big Data Analytics", Interactions, 2012

[111] https://mixpanel.com/

[112] https://newrelic.com/

[113] https://heapanalytics.com/

[114] https://analytics.google.com/analytics/web/provision/#/provision

[115] https://en.wikipedia.org/wiki/A/B_testing

[116] Hassanzadeh, O., et. al. "Helix: Online Enterprise Data Analytics". Proceedings of the 20th international conference companion on the World wide web, pages 225-228, ACM, New York, New York, 2011.

[117] https://www.google.com/maps

[118] http://www.adobe.com/marketing-c1oud/web-analytics.html

[119] http://www.idc.com/prodservIsmartphone-os-market-share.jsp

[120] https://developer.apple.com/library/prerelease/ios/documentation/UIKit/Reference/UIApplicationDelegate_Protocol/index.html

[121] https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIViewController_Class/

[122] https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/LocationAwarenessPG/CoreLocation/CoreLocation.html

[123] https://developer.apple.com/library/prerelease/ios/documentation/Performance/Reference/GCD_libdispatch_Ref/index.html

[124] http://developer.android.com/reference/android/app/Activity.html

[125] http://developer.android.com/reference/android/location/LocationManager.html

[126] http://developer.android.com/reference/android/location/LocationListener.html

[127] http://developer.android.com/reference/android/os/AsyncTask.html

[128] http://aws.amazon.com/

[129] https://cloud.google.com/appengine/

BIOGRAPHICAL SKETCH

Yanke Hu was born in Kaifeng, China. He went to Fudan University in 2003 and majored in Computer Science. He came to the US in 2008. After the completion of his master's degree in computer science from The University of Texas at Dallas(UT Dallas), he started to work in various software companies majoring in Navigation(TeleNav), Business Analytics(Tealeaf Technology, IBM), FinTech(Fintopia), Smart Retail(AiFi) and Healthcare(Humana). He came back to UT Dallas in 2018 and obtained a PhD in Computer Engineering in 2021. His research interest lies in applying the recent progress in deep learning to the healthcare industry.

CURRICULUM VITAE

Yanke Hu

August 1, 2020

Contact Information:

Department of Computer Science
The University of Texas at Dallas
800 W. Campbell Rd.
Richardson, TX 75080-3021, U.S.A.

Email: Yanke.Hu@utdallas.edu

Educational History:

BS, Computer Science, Fudan University, 2007
MS, Computer Science, The University of Texas at Dallas, 2010
PhD, Computer Engineering, The University of Texas at Dallas, 2021

Healthcare Information Platform in AI Era
PhD Dissertation
Computer Engineering, The University of Texas at Dallas
Advisors: Dr.Weili Wu

Employment History:

Senior Cognitive/Machine Learning Engineer, Humana, Irving, TX, Jan 2019 –
Senior Full Stack Engineer, AiFi, Santa Clara, CA, Oct 2017 – Sep 2018
Director of Engineering, Fintopia, Beijing, China, Dec 2015 – Sep 2017
Staff Software Engineer, Tealeaf Technology, IBM, San Francisco, CA, Mar 2011 – Dec 2015
Software Engineer, Telenav, Sunnyvale, CA, Oct 2010 – Mar 2011

Professional Recognitions and Honors:

Star Award, Humana, 2020
Teaching Assistantship, UTD, 2009-2010
Tektronix Fellowship, UTD, 2008
3rd Remin Scholarship, Fudan University, Shanghai, China, 2003-2006
National Physics Competition First Prize, Henan Region, China, 2002

Professional Memberships:

Association for the Advancement of Artificial Intelligence (AAAI), 2020
Institute of Electrical and Electronics Engineers (IEEE), 2020

Publication List:

[1] SqueezeBioBERT: BioBERT Distillation for Healthcare Natural Language Processing, Hongbin George Du, **Yanke Hu**, International Conference on Computational Data and Social Networks 2020, Dallas, US

[2] Faster Healthcare Time Series Classification for Boosting Mortality Early Warning System, **Yanke Hu**, Raj Subramanian, Wangpeng An, Na Zhao, Weili Wu, International Conference on Intelligent Robots and Systems(IROS) 2020, Las Vegas, US

[3] Faster Clinical Time Series Classification with Filter based Feature Engineering Tree Boosting Methods, **Yanke Hu**, Wangpeng An, Raj Subramanian, Na Zhao, Yang Gu, Weili Wu, AAAI 2020 Health Intelligence Workshop, New York, US

[4] Extractive Summarization with Very Deep Pre-Trained Language Model, Yang Gu, **Yanke Hu**, International Journal of Artificial Intelligence and Applications, Vol. 10, No. 2, March 2019

[5] Tealeaf CxMobile – Replaying Real-Time Customer Experience, **Yanke Hu**, IEEE International Conference on Geoinformatics, 2016, Galway, Ireland

[6] A Fast Approach to Enable Mobile Apps with Geo-Location Logging and Reporting, **Yanke Hu**, Xiwang Zhang, Liangmin Hu, IEEE International Conference on Geoinformatics, 2015, Wuhan, China