Patient Health Information in the Cloud

Elizabeth Mathew and Varun Venkatesh
Cloud Computing EN.601.419 Spring 2019

ABSTRACT

Analysis of medical data has been done for a very long time. Being able to make predictions about future patients given ones that have already been seen is a valuable tool in the healthcare industry. In recent years, the use of electronic health data has increased drastically. This poses the risk of data breaches and other safety issues regarding patient data. So, data de-identification is being used to protect sensitive information. We make an effort to combine data de-identification with analysis of medical data through a cloud-based application. Our app is a Django application in Python that uses the Google Healthcare API in order to de-identify the data in FHIR standard formatting that is submitted by the user [6]. It uses the BigQuery API to process the data and a logistic regression in Python to make predictions given the current data. The user enters data on our home screen and can submit any number of entries into the database. At any point the user can view an chart displaying relevant information, or use entered data to make a prediction. The app largely works as planned, but there are shortcomings of data and time. We cannot use real data with the app, to stay in accordance with HIPAA, and the app takes about 10 seconds to display the plots and predictions. Ideally, the app would allow for more capability, handle real datasets, and be more efficient.

INTRODUCTION

Data has increasingly been shifted from paper to digital form, especially in the medical and health care fields. With this comes new challenges and potential problems. Mainly there is the issue of patients' personal health data being stored in digital forms, sometimes online. This data can include sensitive information such as the patients' name, age, date of birth, even Social Security number and others. Thus, the HIPAA Privacy Rule was established. HIPAA specifies

over 19 data fields which are considered patient identifiers, and must be removed in order for data to be deidentified. To solve these problems, data de-identification, which is the process of altering data records to remove this sensitive information, has been used. This way, data can be accessed by those who have permissions, but otherwise the data will not contain patient information. In addition to data de-identification, there is an increasing demand for medical data processing. Since the medical and healthcare fields require large quantities of resources, it is important to be able to analyze patient data to make some important predictions.

BACKGROUND

The prior work in this area is focused mostly on data de-identification alone. The system we have built conducts data processing as well as the de-identification, and is thus different from the systems discussed in the Friedlin paper [1] or the deidentify system [2]. Rather than taking medical reports with sensitive information, removing patient information, and then producing a report sans this information, our system allows the user to input data, store it in a database, and de-identify the data to be used for computations which we show back to the user. Thus, our system combines a user input aspect, de-identification, and data processing all in one cloud-based application. There are some existing systems that use cloud services to store healthcare data, such as the one outlined by Kuo et al [3]. However, these systems usually involve data mining and sharing the data between multiple physical locations. The paper by Kuo et al proposes “a cloud-based data mining platform for researchers in three different geographic locations to share research data/results through the Internet while remaining cost effective, flexible, secure and privacy-preserved.”[3] Our solution just involves storing data entered into a cloud-based database. In addition, the system outlined by Kuo el al does not involve data processing as part of its capability.

DESIGN METHOD

It is important to note that we are not using any real patient data: if anyone wishes to use our patient app to store real patient data, more steps must be taken to ensure the app is not

violating any HIPAA policies. For example, you may need to accept the Google's HIPAA Business Associate Amendment, have patient approval for use of their data, have proper approvals for those who can view the datasets, etc. Thus, this app is suitable for test data only.

First, we set up a basic Django framework. The fundamental component of our application is that it is able to take in patient input and save it to a database. This was accomplished using an FHIR store, part of the Google Healthcare API. This is a set framework that is widely used, and is structured, with specific formatting of parameters and certain data types. We set up a dataset in the HealthCare API, and set up an FHIR store within the dataset according to the online documentation. Within the FHIR store, there are different types of set resources to describe information. For our purposes, we only wanted to store some patient information which included gender, procedure, procedure outcome, and reason for procedure. Therefore, only two FHIR resources were utilized: Patient and Procedure. The Patient Resource contains the gender of the patients while Procedure Resource stores the type of procedure, the outcome of the procedure, etc. After we store patient information into our FHIR store by creating two new resources per patient, we can then perform data calculations.

The first step to ensure that we are not sending any protect health information from our dataset is to de-identify the HealthCare API dataset which contains our FHIR store; the dataset can be deidentified easily with the Healthcare API by creating a new de-identified dataset from the original dataset. Once de-identified we can export the FHIR store in the de-identified dataset to a BigQuery dataset. For our project, two new tables, Patient and Procedure, are created in the BigQuery dataset, and each table has a list of all the patients entered via the user-input form. Using the BigQuery API, we can access the tables from our BigQuery dataset and run SQL commands and data cleaning in order to render the calculations and visuals with Python. Then we display the plot to the users, and create another page so that users can input test data for a logistic regression and make a prediction for the success rate of a procedure based on features they select.

DESIGN CHOICES

We decided to use the Django app due to easy integration with most APIs. The challenge with using Django was the difficulty to learn it with no prior UI or frontend experience. We came across Flask which is more user friendly, but it would have been difficult to switch to Flask while in the middle of our app development process. For the de-identification of patient data, we first chose Very Good Security[5] to store secure data and redact information. We ran into problems with it, however, because it proved extremely difficult to work with to integrate with our app. After some time and experimentation, we moved on to a solution that was easier to integrate with the app and offered more documentation for use, which was the Google Data Loss Prevention API. This API had a method for de-identifying information that seemed optimal for our use. We came across some problems with the integration for our particular datasets, but noticed that the new beta release of Google's Healthcare API was perfect for our work. The Healthcare API offers HIPAA compliance, de-identification for PHI, and FHIR formats. In addition, we were able to use the datasets accessible via the HealthCare API rather than CloudSQL which was our original choice for storing data. The HealthCare API dataset offered is actually a container in our GCP project holding "modality-specific healthcare data"[4]. We decided that this structured dataset was ideal for our project since it maintains the FHIR standards.

To ensure a timely completion of the created app, we limited input to certain features and stored this small set of features in our dataset. Therefore, we could only take a small subset of these features, gender and reason for procedure, for our prediction calculations. We also attempted to use age but ran into problems with storing age in the FHIR store with correct formatting. We chose to export to BigQuery since it was an efficient way to transform FHIR stores to tables on which we could do SQL queries. Using the client libraries for BigQuery we could also directly create a python dataframe from the SQL query. The plot we display to the user is a plot of the rate of success for a procedure based on different reasons for having the procedure. We thought this plot was a good way for users to visualize how many people have successful procedures in the dataset. The prediction we display is a Logistic Regression which uses gender and the reason for procedure from our dataset as the "training" data to predict the success rate of a procedure given a "test" sample of user-inputted features.

A major problem with testing our app was that the Google APIs did not produce useful error messages, so understanding them and working with the platform was very difficult. Running the app locally was completed successfully but there were problems with deployment. Images can not be saved to the static folder during deployment, so we cannot dynamically change the plot image on the deployed site to reflect new patient entries. However, prediction still works perfectly.

EVALUATION

The app is largely working as designed, with a user interface to enter data and the ability to produce predictions based on some features. The user interface is simple and intuitive. As it is a basic application, it is easy to use. An assessment of the quality of the predictions can be done if there were significant testing and training data sets. However, we are limited by the data that we have, which is not real data.

Time is also a shortcoming of the application. The app runs slowly, taking nearly 10 seconds to display a plot or display a prediction as shown in the table below. It would be difficult for a user to use this app over an extended period of time.

Type of Display	Average Time (sec)	Standard Deviation (sec)
Plot	7.1919782876968	1.5598146872267
Prediction	9.2735009670258	1.8648272094715

LIMITATIONS

We are limited by type of data we obtain. Since we cannot use real patient data, calculating the accuracy of our prediction models is not indicative of how the model may perform for real patient procedures.

Another limitation is that our app takes in input by entering data into a form manually for each patient so it is difficult to create a huge sample size for testing.

CONCLUSION

Our app provides a strong first step towards combining de-identification and data processing with cloud based computing. The cloud application works well to connect all of the used datasets and functions. There is room to improve how efficiently our app runs by restructuring the templates so that we do not perform repeated operations. Although we are sometimes limited by the APIs which we chose, they ultimately provide an efficient way to integrate cloud storage with healthcare data standards. As the resources available to us continue to develop, stronger implementations for this app can be achieved.

FUTURE WORK

Our app currently takes in every new entry from the user input page as a new patient. In the future, including a method to enable users to change existing data based on a patient's name and date of birth would be useful for users. Exploring more visualizations and predictions will also help to create more useful application. Also, enabling a larger selection of features for users to input will support a more accurate prediction model.

REFERENCES

[1] Friedlin FJ, McDonald CJ. A Software Tool for Removing Patient Identifying Information from Clinical Documents. J Am Med Informatics Assoc. 2008;15(5):601–610.

[2] P. Burckhardt and R. Padman, "deidentify," AMIA Annu Symp Proc, vol. 2017, 485–494, 2017.

[3] Guo, Y., Kuo, M., & Sahama, T. (2012). Cloud computing for healthcare research information sharing. In *Proceedings of the 4th IEEE International Conference on Cloud Computing Technology and Science* (CloudCom) (pp. 889-894). Taipei, Taiwan: IEEE.

[4] "Projects, Datasets, and Data Stores | Cloud Healthcare API | Google Cloud." *Google*, Google, 24 Apr. 2019, cloud.google.com/healthcare/docs/concepts/projects-datasets-data-stores.

[5] "Getting Started." *Getting Started - VGS Docs 1.0 Documentation*, www.verygoodsecurity.com/docs/getting-started.

[6] "FHIR Overview." *Overview - FHIR v4.0.0*, 27 Dec. 2018, www.hl7.org/fhir/overview.html.