Choosing Health: Making Big Data Work for Families

205 Final Project

Ashton Chevallier, Andrew Lam, Eric Tsai

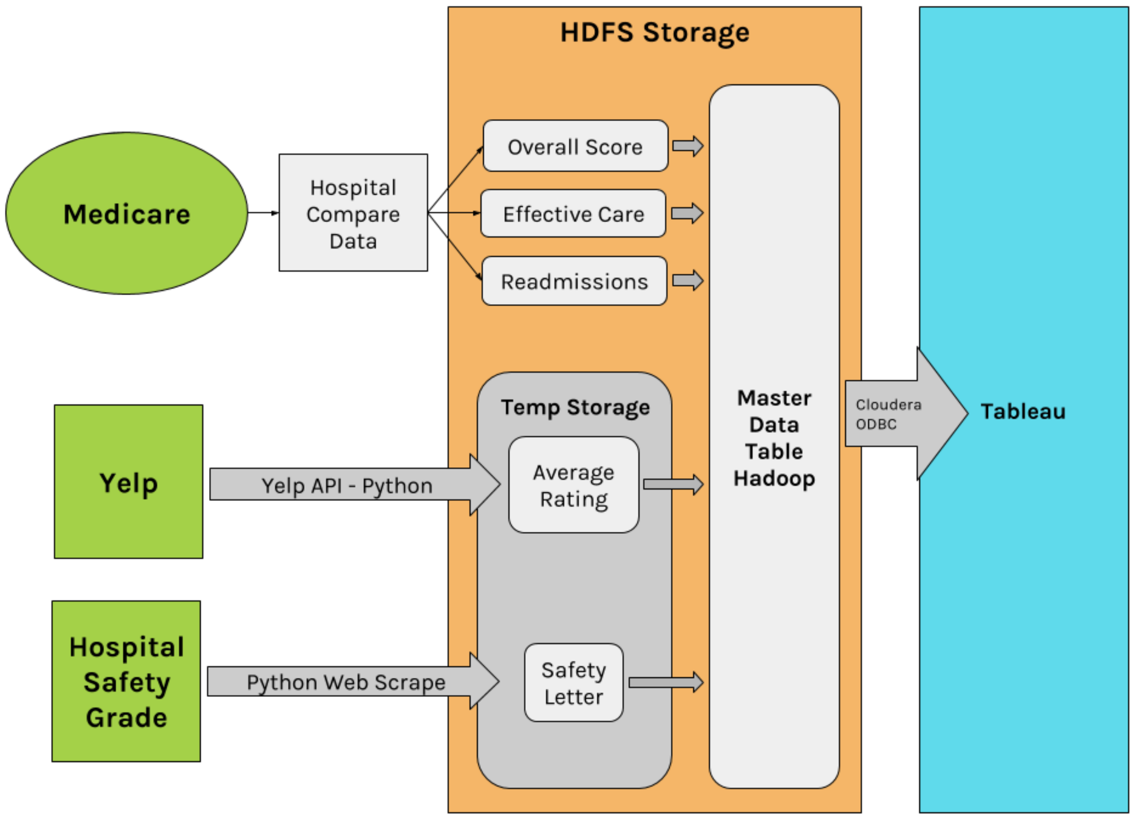
December 15, 2016

Introduction

One of the biggest problems with US healthcare system is the lack of transparency around general hospital information. Prices are unknown, doctors reputations are only gathered by experience and when you actually have a data you want, it's in different websites, in different formats. Using the data in aggregate, where it's most useful, is almost impossible.

Our system is designed to make that part of health care research that much easier. By cleaning, aggregating and visualizing multiple healthcare datasets, we can offer consumers a chance to make better health choices with data not normally available to them.

Architecture



Our general guiding principle was to get things operating. We needed to access and download 3 different datasets, with different keys, sets and formats and join them together in a meaningful way. Our initial schema incorporated more HDFS tables, but in the end we didn't need the extra storage and retrieval capability that hadoop offered. Once we dove into the data we found the scale to be in the 'medium' range, even with the entire US covered. HDFS wasn't adding any advantage over a standard filesystem. However, we did chose Hive to manage our serving layer because it allowed for simple query building for the master tables and compatibility with Tableau.

Yelp Extraction

Yelp is where consumers around the world rate various business with their experiences. Potentially, a business's average yelp star rating is correlated highly with the overall experience and quality of a hospital. We felt this additional consumer insight would be a useful addition when combined with the Medicare and Safety data.

The Yelp extraction script is relatively simple, by utilizing the Yelp Fusion API and the 'requests' python package we are able to pull data direction from yelp into our machines for processing. (Note: the secret code and client id are hard coded to our project). If altering or copying code, pay close attention to the syntax on the token, client, setup. The API is finicky and must be used in precise manner. The online documentation was lacking, making this output more difficult than it should have been.

The core of the code is looping through the zip codes you want analyzed, requesting yelp data for each one. (IMPORTANT: Yelp limits your requests to 25k a day, so you can't get all of the US in one shot) Pay attention to the params object, as it defines how we are doing the yelp searching. We specifically wanted hospitals, so we used the params: 'terms' of hospitals and 'categories' of medcenters. This doesn't give us a precise match to our medicare data, but should give us a great chance of getting significant matches.

Once we're done looping through the desired zip, we create a pandas data.frame and save it to csv for further processing. Future releases may do this processing directly in HDFS if our data gets bigger, rather dealing in transferring flat files back and forth.

Hospital Safety Grade Web Scrape

As the name suggests, the Hospital Safety Grade is a measure that scores hospitals on a scale of A to F, and reflects how safe the hospital keeps patients from errors, injuries, accidents, and infections.

Hospital Safety Grade information can be acquired by running the "Hospital_Safety_Grade.py" file. This script takes the argument 1, 2, or 3 at the command line to process the information in three different parts. This is achieved through the PyQt4 package, which provides the ability to pull Javascript from a website. Processing each part takes about 5 minutes and is outputted to a CSV file:

1. "Hospital Safety Grade - Part 1.csv"
2. "Hospital Safety Grade - Part 2.csv"
3. "Hospital Safety Grade - Part 3.csv"

Medicare.gov Data Cleaning

The Medicare data is loaded into HDFS, then loaded into Hive with a DDL file. The columns of interest are then extracted and a the mean of the score values for effective care and readmissions is calculated across all measures performed at each healthcare provider. The columns are then consolidated in a single table with the overall rating score, the data on how each hospital compares to the national average across seven categories and the average effective care and readmission scores. This table is written to file to be merged with the other data sources.

Data Merge

Python was used to match the three datasets. At a high level, the script ("data_merge.py") is divided into four sections:

1. Matches across all three datasets
2. Matches between the Medicare and Hospital Safety Grade datasets
3. Matches between the Medicare and Yelp datasets
4. Medicare data that does not match the Hospital Safety Grade or Yelp datasets

Left joins are applied on the Zip Code field to speed up processing. Then, the Levenshtein package is used to match hospitals with similar names and addresses. Finally all four parts are appended and outputted as a CSV file ("Master_Data_Set.csv").

Storage and Serving

Once the data has been merged into a single table, the data is loaded into Hive. A DDL is applied and a Hive server is started to serve as a public data interface for Tableau.

Visualizations

To show off how we collected and combined all the data, we felt Tableau 10.1 was the best choice. Going with the latest version was a bit of a hedge against future usage over compatiblity. The visualizations chosen show off how this data can be combined and analyzed together revealing further insights than could be achieved by any data set alone.

To get the vis working you must install both Tableau 10.1 and Cloudera Hadoop ODBC Drivers. Once both are installed on your machine, open the Tableau 205_Final_Dashboard.twb workbook. You may have to configure the data source, if so use the IP address of your machine running the Hive server and chose Hive Server 2 as type and root login...no password should be required.

We did some additional calculations on our dataset once it arrived, most importantly our Meta Analysis metrics. These were created by recoding some of the nominal variables to numeric so they could be aggregated in a meaningful way. The MetaAnalysis gives consumers a way to understand all the datasets simultaneously to make the best hospital choices for them.

GitHub Directory

Name	Description
docs	
● w205FinalProjectDocumentation.pdf	This file
● W205 Project Presentation.pdf	MS1: Initial Project Presentation
● W205 Project Proposal - Eric, Andrew, Ashton.pdf	MS2: Project Proposal
● W205ProjectProgressReport-EricAndrewAshton.pdf	MS3: Progress Report

● w205 Final Project - Andrew, Ashton, Eric.pdf	MS4: Final Project Presentation
sql	
● effective_care_ddl.sql	Extracts columns of interest from effective care Medicare table
● hive_base_ddl.sql	The data definitions for the downloaded Medicare files
● master_ddl.sql	The data definition file for the final master data set
● medicare_merge.sql	Performs mean operation on effective care and readmission data and merges it with the hospital overall ratings
● readmissions_ddl.sql	Extracts columns of interest from readmission Medicare table
.gitignore	Used to ignore files in the repo
205_Final_Dashboard.twb	Final Tableau dashboard
Hospital Safety Grade - Part 1.csv	Part 1 of Safety Grade data
Hospital Safety Grade - Part 2.csv	Part 2 of Safety Grade data
Hospital Safety Grade - Part 3.csv	Part 3 of Safety Grade data
Hospital_Safety_Grade.py	Python script to scrape Safety Grade data
Master_Data_Set.csv	Master data set for Hive
README.md	README file
data_merge.py	Python script to create master data set CSV
load_medicare_data.sh	Downloads data from the Medicare repositories, loads them into HDFS, and consolidates the data of interest
master.sh	Uploads the master database to Hive and starts the Hive server

medicare.csv	Medicare data
setup.sh	Runs the end-to-end data loading and processing and starts the database server
update_python.sh	Optional script to install many of the packages for the Python scripts.
yelp_final.csv	Yelp data output from yelp_pull.py
yelp_pull.py	Python script to pull Yelp data