Nic Dobbins
BIME 550
Project 1: Build a simple ontology /knowledge base with RDF

My research interests include creating systems to process natural language questions to dynamically query clinical databases of arbitrary data models. A contrived example of the use of a hypothetical system could begin with the request, "Find patients with a [past] diagnosis of asthma", where in response the system is able to infer something akin to:

- `Asthma` *is a* `physiological condition` ->
- `ICD9/10 codes` *represent* `physiological conditions` ->
- `<database>` *has* `<table><column>` which *encodes* `ICD9/10` ->
- A `query` *can be written using* `<database>` to *find humans* with `Asthma` *using* `code(s) <x>`

As such, I chose to attempt to model the metadata of the MIMIC-III[1] database (specifically, an instance of it on my home computer). In order to create such a system, I sought to ensure that 1) linkage of the database schema to the clinical and biological concepts their data represent is done by `OWL Object Properties` such as *Encodes*, *HasColumn*, and *Represents*, and 2) database schema data (e.g., `Column` *DataType* and *Name*) are captured in sufficient detail to be able to construct a SQL query on MIMIC-III based on the ontology. I encountered a number of interesting questions and ideas during the project.

First, I struggled with the fact that my local copy of MIMIC-III is one *instance of many*, rather than a monolithic, universal instance. The MIMIC-III database schema (as maintained by MIT) changes over time, and even its representation on my local computer in *SQL* may differ elsewhere – the MIMIC-III data could be represented by RDF triples or CSV files as well, in which case their ontology would differ. Therefore, if I needed to represent multiple instances of MIMIC-III data, for example, I found it would be critical to model their *instance-specific* representations appropriately in the ontolo(gies).

I also wrestled with questions of how best to model database column metadata. For example, the `Column` [SUBJECT_ID] appears on nearly all MIMIC-III `Tables` and represents unUse the "Insert Citation" button to add citations to this document.

ique patient identifiers (specific to MIMIC-III). Every instance of [SUBJECT_ID] is an `integer` type. Initially, I concluded that it would therefore be best to simply have a single [SUBJECT_ID] instance of class `DatabaseColumn` with each Table it was present in connected by an RDF triple, i.e., `?table :HasColumn :SUBJECT_ID`. While this representation was convenient and appeared to work for [SUBJECT_ID], however, as a general ontology pattern it simply wouldn't hold true for all other cases. For example, a column named [CATEGORY] appears in two `Tables` ([D_ITEMS] and [D_LABITEMS]), but carries a different semantic role and domain for each (i.e., categories of general clinical measurements versus categories of laboratory tests). Thus I ultimately chose to refactor all Columns into table-specific instances with the URI pattern `<TABLE_NAME.COLUMN_NAME>`, and each with an RDF *Name* `Data Property` of `<COLUMN_NAME>^^xsd:string`.

In terms of `Column` *DataType* (e.g., `integer, float, string`), I experimented with two different patterns. I began with the observation that *rdf:datatype* types mostly corresponded to the database types (e.g., */XMLSchema#int* is functionally equivalent to a SQL *int*). It therefore seemed that I could represent a given `Column DataType` as either (in RDF/XML):

[1] Johnson, A., Pollard, T., Shen, L. *et al.* MIMIC-III, a freely accessible critical care database. *Sci Data* **3,** 160035 (2016) doi:10.1038/sdata.2016.35

- `<:DataType rdf:datatype="…/XMLSchema#int"/>`, (effectively using the RDF `datatype` itself as the value)
- `<:DataType rdf:datatype="…/XMLSchema#string">int</:DataType>` (with a `string Value`)
- `<:DataType rdf:resource="#int"/>` (with an RDF `resource Value`)

Ultimately I chose the third option and created instances of `DataTypes` specific to my database, as this allowed the greatest flexibility and representative power in modeling instances of MIMIC-III, as discussed earlier.

Overall I found the exercise to be very helpful in learning RDF and OWL. I look forward to learning how to use OWL to implement first order logic and I hope I can build upon the model I've developed here.