Name:	Hang Su
Affiliation:	Georgia Institute of Technology
Email:	hangsu@gatech.edu
Presentation type	Poster

Scalable Cohort Construction for Patient-level Predictive Modeling

Hang Su, BS[1], Sherry Yan, Ph.D[2], Walter (Buzz) F. Stewart, PhD, MPH[2], Jimeng Sun, Ph.D[1]
[1]Georgia Institute of Technology, Atlanta, GA, USA, [2]Sutter Health, Walnut Creek, CA, USA

Abstract *Cohort construction, which aims at finding suitable subjects for a study, is the essential first step for clinical predictive modeling. Existing tools that leverage SQL and relational databases are suffering from significant performance issues especially when the underlying data volume is large. There are two main challenges in cohort construction: 1) flexible and intuitive programming interface to describe complex criteria for the cohort, and 2) efficient computation for extracting the cohort from observational data. To address these challenges, we proposed a flexible domain specific language (DSL) for defining cohorts and developed a simple and efficient intermediate patient representation for supporting parallel cohort construction. We demonstrated the expressive power of the DSL using cohort construction for epilepsy refractory patient prediction as an example.*

Introduction Rich health delivery and patient health profile contained in electronic healthcare records (EHR) brings new opportunities for secondary usage of healthcare data and make data-driven prediction possible. Predictive modeling based on those observational data also benefits from the large EHR data. However, challenges also come along with increasing data volume and heterogeneity natural of EHR data, and efficiency becomes one of key components to evaluate predictive modeling. Cohort construction[1], as the first step of predictive modeling process, consumes all available data and output eligible patients. The standard methodology for cohort construction often requires researchers to interact intensively with patient data stored in relational databases such as one using OMOP CDM by using query tools like OHDSI Circe[2] to extract the relevant patients and extra processes to identify outcome, and index date[1] (a date when corresponding prediction is made for a given patient). Traditional method suffers huge inefficiency due to following reasons: 1) relational data model by design is not suitable to succinctly describe complex cohort definitions[3], which leads to complicated and unreadable SQL expressions; 2) observational data is longitudinal in nature but most relational database management system (RDBMS) lacks efficient query support for temporal operations; 3) cohort definition, which is often iteratively updated as the study progress, demands an efficient computation for extracting corresponding cohort from various patients.

In this work, we proposed to use an intermediate patient centric data model that combines idea from data warehouse (DW) and NoSQL database. We encode all the patient information (e.g., diagnoses, medications and procedures) into *events* and group all events by patients and ordered by time. The proposed data format can be easily compressed and stored in parallel storage system such as HDFS. Because the proposed model is quite simple, we can easily convert EHR data into the proposed format. Based on proposed data model, we designed a general strategy to specify eligibility criteria and identify prediction outcome, index. Instead of using SQL to define cohort, we developed succinct and flexible domain specific language (DSL) with great expressive power. By storing the proposed data model on HDFS, we implement cohort construction using Apache Spark to improve the computation efficiency.

Method The atomic component of our data model is event, which has format *e := {patient-id, concept, begin-time (BT), end-time (ET), properties}*, where *properties* are optional key-value pairs for extended information. Only *patient-id* and *concept* fields are required. *Concept* comes from OMOP's vocabularies. The *BT* and *ET* are not required as some concepts do not have a timestamp associated with such as gender and ethics. With timestamp, one can either encode event at a single time point by specifying the BT only (such as diagnosis event), or specify a time duration with both BT and ET (e.g., hospital stay from *BT* to *ET*). As such, we store one patient as a series of *event*s ordered increasingly in *BT* (if exists). Physical storage of *event* on disk may be raw file on HDFS or on NoSQL database such as MongoDB. New events can be appended to the existing data.

Given the above data model, we defined four transformations of *event* sequence, which are building blocks for specifying prediction outcome, index date and eligibility criteria: 1) **filter** events using concept, time and properties. For example, extract events about v 250.* ICD9 code; 2) **project** event to a new value. For example, convert lab values into low, normal and high based on their normal ranges; 3) **Subset** original event sequence. For example, take events within a given time window; 4) **group** one or more events from original sequence to form new event.

For one patient, prediction target was obtained by a series of chained transformations on the original event sequence, followed by identification of index date with another series of transformations. If target or index cannot be extracted, this patient may be regarded as ineligible or requires further processing. The eligibility criteria on index date are categorized into 1) **aggregation** based criterion such as total duration of hospital stay events should be more than 10 days; 2) **temporal** constraints such as at least two occurrences of v 345.* ICD9 code followed by at least anti-epilepsy drug (AED); 3) **composite** criterion that combines multiple criteria using Boolean logic.

We developed a DSL for chaining *event* transformations and composing eligibility criteria using Scala programming language. The cohort construction process runs on top of Apache Spark. Spark parallelizes operations on Resilient Distributed Dataset (RDD), a distributed array-like container. We first load data into a RDD of events then group and sort data to be RDD of patients (sequence of events). The grouping by patient is an efficient strategy as typically there is no interaction across patients during cohort construction.

Sample Use Case Here we use an example to illustrate cohort construction for predictive modeling using the methodologies proposed above. The outcome of the study is whether the epilepsy patient is refractory or not. The case patients are those who had at least 4 AED failures (changing or adding AED are considered as failures). The control patients are those who only failed once. For both case and control patient, index date was defined as the date of first failure. Patients met additional eligibility criteria at index date were selected: 1) at least one 345.* or two 780.39 ICD9 code within two years before index date followed by continuous AED refill for at least 6 months; 2) patient should be at least 16 years old at index date. Below example shows the DSL snippet for constructing the target of a case patient:

```
1 val target = (($"type" = "medication") & ($"concept" in List("GABA", "CABA", ))) |
2              Overlap(maxGap = 5 days) | /*generate new events from overlap*/
3              Merge | /* concat events into one */
4              ($"count" >= 4)
```

Line 1 filter to keep all anti-epilepsy drug events and line 2-4 are additional transformations. Patients who failed in any step (i.e. empty result) become potential control and we followed similar steps to define controls' target. In this example, we took *BT* of target event as index date. For other applications, user may need to define transformations like above. Then we checked eligibility of patients at corresponding index dates using below criteria:

```
1 val ageEvent     = ($"concept" = "DOB") | OffsetToIndex(granularity = Years)
2 val ageCriterion = ($"value" >= 16).on(ageEvent)
3
4 val diag780Event = ($"concept" = "v780.39") | WithIn(-2 years, null) | Merge | ($"count" >=2 )
5 val diag345Event = ($"concept" rmatch "^v345") & ($"type" = "diagnosis") | WithIn(-2 years, null)
6 val drugEvent    = (($"type" = "medication") & ($"concept" = "GABA")) |
7                    WithIn(-2 years, null) | Concat(maxGap = 7 days)
8
9 val criterion    = ((diag780Event < drugEvent) or (diag345Event < drugEvent)) and ageCriterion
```

We defined relevant events involved in each atomic criterion and created the final composite criterion. *WithIn* transformation is a filter to keep events of interests within two years before index date. We used Allen's interval algebra[4] and operations to define temporal relationship between events. For example, line 9 of above code block shows diagnosis should take place before medication event.

Conclusion A new cohort construction module for predictive modeling has been developed. This module takes flexible *event*s as input and chained event transformation mechanism is applied to to define prediction outcome, index date and eligibility criteria. Running on top of Apache Spark made the utility scalable to processing large healthcare observational data. Next, we are conducting experiments and will present the quantitative performance comparison to the traditional cohort builder such as Circe in the poster.

References

1. Ng K, Ghoting A, Steinhubl SR, Stewart WF, Malin B, Sun J. PARAMO: A PARAllel predictive MOdeling platform for healthcare analytic research using electronic health records. Journal of biomedical informatics. 2014 Apr 30;48:160-70.
2. Cohort definition and syntax compiler tool for OMOP CDM. https://github.com/OHDSI/Circe.
3. Kimball R, Ross M. The data warehouse toolkit: the complete guide to dimensional modeling. John Wiley & Sons; 2011 Aug 8.
4. Allen JF. Maintaining knowledge about temporal intervals. Communications of the ACM. 1983 Nov 1;26(11):832-43.