Data Flow for HCC Profiler

In the beginning, data is acquired from our customers. This data may be sourced from different systems or arrive in separate batches over time. We map this data into our own data model as part of our mapping and pipeline processing phases. Once data is completed processing, it is persisted as Apixio Patient Objects and sent to our Event Extraction process. The Event Extraction processes utilizes the models and algorithms devised by the science team to convert the Patient Objects into events. These events form the atomic form of patient data and include structured data (like problem codes and RAPS return data), narrative events (like appearance of a certain text string in a document) and even user annotations.

These event are then bundled based off of bundling parameters configured in Project settings and provided parameters that scope the data set and whether to include previous annotations. The bundling parameters basically outline which filtering criteria should be used on the NARRATIVE and CMS_KNOWN events and the results of bundling these collections of events into an opportunity.

The collection of events for a patient that could create opportunities are all of the NARRATIVE events that survive bundling parameters that are “hoisted” from APXCAT codes and dates of service to HCC codes (see section about hoisting APXCAT). These are grouped by HCCDescriptor and filtered to be only for the ones in the bundling parameters (basically sweep filtering). The resultant collections are opportunities (patient, hcc_code, hcc_labelset, hcc_sweep, hcc_paymentyear). This collection is then filtered to be only ones without claims (CMS_KNOWN). Again this is done in hoisted space (see section about hoisting ICD codes). Once this is all done, the opportunities have multiple findings (the grouping of NARRATIVE events (evidence) for a document), encrypted patient demographics, and summaries are attached to the opportunity and it is persisted in elasticsearch. At the end of this process USER_ANNOTATIONS advance the status of opportunities, if this process is enabled in the bundling parameters.

Our story begins with a coder being presented with an opportunity in the application. This is a served opportunity. It came randomly from the pool of opportunities in elasticsearch (or semi-randomly if energy-routing is on) based upon the rules the coder has (see section about Outbound Rules). As a served opportunity, it might not contain all of the evidence that supported the creation of the originating opportunity. The actual thing presented to this coder is an HCC, description, and a list of documents that may or may not have evidence to support that HCC. There can be some other metadata supplied, but this list is continually in flux.

Our valiant coder, starts the lengthy task of reading these 100+ page documents looking for evidence supporting the HCC. They start from a page we, Apixio, suggest, but alas if it isn’t there, they comb through the entire document looking for evidence. Whenever our coder spots

a likely nugget, they go through the process of recalling the coding guidelines for this project (dates of service and anything else special) before annotating the finding. If our coder is particularly lucky, this will be an Accept and they will proceed to fill in date of service, pages, ICD9, and provider, otherwise they reject with a reject reason. If anything is confusing, our coder leaves a comment (only enabled through a “flag for review” button).

In this whole process, the coder generates annotations (USER_ANNOTATION). They could potentially create an infinite number of these, but ideally there is an equal number of annotations as number of presented documents. These annotations then flow up into the cloud and magic happens. This magic takes the form of opportunities progressing in the bundler and output report accruing knowledge.

In the bundler when an annotation comes in, it starts to go through the gauntlet.

1) It is pushed on a collection of other annotations waiting for flush()
2) All projectIds are extracted and verified from the annotations.
3) Each annotation then has a runtime tag (different than presentedTag) set with the orgId.
4) The annotations are grouped by this tag (orgId) and persisted as AnnotationDocuments in elasticsearch.
5) For each annotation in the above list do the following steps:
6) Infer an APXCAT code from presented* fields if no code exists on the annotation.
7) Record the annotation in the correct energy profile.
8) Hoist the annotation (without “bifurcation”)
    a) If a “reject” annotation filter this hoisted set to only be for presented Hcc.
    b) If something not “accept” or “reject” just put this on presented Hcc.
9) Find all the opportunities for the annotated finding.
10) If this is an accept that is outside dates of service (presentedHCC is not in the set of hoisted HCCs) “accept_sweep_modified”.
    a) Push a new annotation of “accept_sweep_modified” result back into queue to process next time.
    b) Look for any opportunities in elasticsearch that this accept hoists to, but don’t have the annotated finding.
    c) Clone the finding onto the opportunities in 10b. and add them to the opportunities in 9.
11) Select only the opportunities that have an HCC in the HCCs that resulted from hoisting in 8.
12) For each opportunity in 11.
13) Create a UserActionDocument.
14) Add the annotation to the correct finding if the annotation isn’t already on the finding. Also while at it update the total summary statistics, user summary statistics, and rollup winner. Rollup winner is the latest accept with the highest tag, and if no tag then the latest reject with the highest tag.
15) Set the tag on the opportunity to the presentedTag on the annotation.

16) Run the inbound rules engine and update opportunity's status and persist the opportunity.

Running the inbound rules engine is really a gauntlet in itself, so here are the steps.

1) Save the starting state, and if ever starting state is equal to current state or if current state has ever been seen before, then stop.
2) Traverse the inbound rules and find the first that the opportunity satisfies. (see: Current Inbound Rules)
3) Apply the transition to the status.
4) Repeat 2-4 until exit criteria in 1.

For the sake of completeness, the operation taken on each annotation in the output report is:

1) Lookup the reportable code for the presentedHCC of the annotation (create new if not there).
2) Deduplicate the annotation by eventId, and continue if it is new annotation.
3) If the annotation is an "accept" get all hoisted codes (see hoisting ICD) and corresponding reportable codes in elasticsearch (create new if not there).
4) Find out if this annotation is overriding an accept on the presented reportable code.
5) If it is, get all the reportable codes that previous accept hoists to.
6) If this annotation was a reject, propagate it to all reportable codes that are in set(5 - 3).
7) If this annotation is an accept, propagate a faked "reject" to set(5 - 3).
8) Propagate the annotation on all of set(3).
9) Calculate the winner (or lack-thereof) and persist all touched reportable codes.

Source Data

The source data from our customers arrives in a variety of formats. It includes:

- Patient Details
    - Demographics including Name, DOB and Gender
    - Primary and Alternate IDs including Member ID and HICN
    - Health Plan Information
- Clinical Data
    - Structured Documents
        - The include CCD/CCR or other XML formats which include codified medical history of the patient
        - All information about the patient and document are encapsulated into the document itself.
        - These are rendered into a textual document via XSLT
    - Unstructured Documents
        - Document Metadata
            - Patient ID - links to Patient Details
            - Document Title - if unavailable, we will use a unique identifier

                - Document Date - if unavailable, we will try to “extract” the date or otherwise use an “assumed date”
                - Encounter Information including Provider
            - Textual Documents - including plain text, RTF, or text included in PDFs
            - Image Data - TIF/PDF or other image data which must be OCR’d to extract text information.
- Subtraction Data
    - Includes information used to eliminate opportunities before they are sent to users.
    - RAPS
        - supports ICD9 or ICD10
        - Keyed by HICN
    - General Claims Data
        - CPT Filtering is applied pre-ingest (although it should be moved to the event extractor in the future) to limit ICD9 codes to those associated with a F2F Procedure code.
        -
- Provider Data
    - Used to populate the drop down list in the application. This requires an NPI list, so source providers with no NPI or one not in our mapping won’t be included.

Pipeline Processing

Most data follows the same path of Parsing, OCR (if necessary), Persistence, and then Event creation. This will be built out more once we add Virtual Document segmentation.

OCR Job

Since many of our projects are PDF projects, OCR is critical. We have several configurable settings in OCR;

- Default DPI: only used if source DPI >=200.
- Text skip limit - the number of text characters which triggers the skipping OCR for a page
- Timeout - how long to spend OCR-ing a single page before skipping it

Event Extraction

Before the Source Data is usable for The HCC Profiler, it must first be converted into Patient Events. Patient Data is converted into Events as part of the standard processing pipeline, but it can also be “re-run” on subsets of patient data with different settings.

The Event Extraction job has Global and Org Level settings::

- validation

    - Values: true/false
    - Controls whether the Patient Object is validated with the APOValidator. If a Patient Object is invalid (has a code without a code system, for example) the entire thing will be discarded. This is currently the case for most (if not all) CCD and CCR files.
- spell_check
    - true/false
- spellcheck_verify_certs
    - true/false
- spellcheck_url
    - https\://spellcheck.apixio.net\:443/spell
- $modelFile
    - production_models_20150820.zip
    - Model files are versioned and contain the text to code mappings used to make predictions in dictionary and v5 extractors
- force_date_extraction
    - false
    -
- extract_dates
    - true
    - enables date extraction if the source data has metadata flag “dateAssumed” = “true” OR if the force_date_extraction flag is set to true
- extractors
    - Pipe-delimited classes
    - com.apixio.extractor.event.extractors.StructuredConditionEventExtractor
        - Bucket Type: StructuredConditions
    - com.apixio.extractor.event.extractors.DictionaryBasedEventExtractor
        - Bucket Type: dictionary.v0.1
    - com.apixio.extractor.event.extractors.MaxentV5Extractor
        - Bucket Type: NARRATIVE
- V5THRESHOLD
    - float (ex: 1)

Event Details:

- Subject - Patient UUID
- Source - Document UUID
- Fact
    - Problem Code (APXCAT, ICD9, ICD10, etc.)
    - Start and End Dates
- Attributes
    - bucketType - used to differentiate “V5” from “dictionary” (ex: dictionary.v0.1)
    - $jobId - set from the hadoop job (ex: 42422)
    - $patientUUID (ex: 7f52a5a2-27d5-4fbb-b3a5-86e39ccfdd13)

    - $orgId (ex: 372)
    - sourceType - taken from patient object (NARRATIVE)
    - encHash (nHwd11E3lGtcVTio1sbBNg==)
    - $documentId (allergy and asthma medical group4f93f758-3345-4e8e-b1db-8ce507311dd5)
    - $documentUUID (d0bab9a4-ac12-490b-b0fa-11da05b63156)
    - editType (ACTIVE)
    - editTimestamp (2014-10-08T20:41:51.867Z)
    - $workId (40516)
    - bucketName - (ex: dictionary.v0.1.ex6d_big_idf_BRDREC20141028_RETRAIN_2014_10)
    - totalPages (ex: 1)
    - $batchId (ex: 372_bundlerStressTest)
        - Either the original pipeline batch or the group of resubmitted data
        - Resubmitted data can be a list of patients or a list of documents
    - $propertyVersion - matches settings used above (ex: 1427913857741)
- Evidence
    - inferred - true/false (indicates if this is coded or predicted data)
    - Sourece - dictionary used (ex: apxcat_dict_20140918b_BR.txt)
    - Attributes (only in the case of predicted codes or annotations)
        - appliedDateExtraction
        - hasMention - whether we actually found the evidence(ex: YES)
        - title - The title of the document
        - face2face - whether this was a f2f encounter (ex: true/false)
        - f2fConfidence - a 0 to 1 confidence in the face2face prediction
        - predictionConfidence
        - junkiness - a 0 to 1 score of the ‘junkiness’ of the data
        - associatedEncounter (ex: 872b9602-16c9-4616-81b5-d9454538bdd6)
        - snippet - the string that predicted the code (ex: diabetes)
        - pageNumber - where the snippet was found
        - mentions - now the same as snippet
        - patternsFile - the patterns file used
        - extractionType - is this always plainText?

Bundling

Bundling is currently driven entirely from project settings. These filter the data set and instruct the bundler on what analysis to create opportunities for.

Bundling parameters:

- Project Settings
    - Sweep and Payment Year
        - Defines the Code Filter (what versions, sweeps and payment years):

                - 2014: V12-$sweep-2014;V22-$sweep-2014
                - 2015: V12-$sweep-2015;V22-$sweep-2015
                - 2016: V22-$sweep-2016
            - Used to default the DOS field in the CMP UI
        - DOS start and DOS end
            - NARRATIVE rules - include all in range as “ADD”
            - CMS_KNOWN rules - include all in range as “CLAIM”
        - Org Name - identifies the elastic search index
        - Project Id - opportunities are tagged with a project id when bundled (opps can be in multiple projects)
        - Doc Filters - a list of document titles to filter from inclusion during bundling
        - Patient List - a list of patient to include events from sequence store
- API Parameters
    - Batch Ids - default: all Event batches
        - Limits Patient Events to those with a $batchId in this set
    - User Annotations - default: All
        - Accept - include just accepts
        - Reject - include just rejects
        - None - include no user annotations

How bundling actually works: TODO

Hoisting

There are two parts minimum (a third part optional) to Hoisting.

Part 1:
The first part is taking a code and codeSystem and mapping it to some HCCs that include the HCC number and the label set version (model version). This code and code system can be V12_18, APXCAT for APXCAT codes or 25050, 2.16.840.1.113883.6.103 for ICD9 codes.

Part 2:
Once the HCC number and label sets are determined, then a date of service is used to figure out which sweeps the hcc and label set version are valid. This filtering is algorithmic and able to be described in the following way:

If the month is in the later half of the year, starting after June, the following are valid sweeps:

(midYear,YEAR+1) and (finalReconciliation,YEAR+1) and (initial,YEAR+2)

If the month is in the first half of the year, starting before July, the following are valid sweeps:

(initial,YEAR+1) and (midYear,YEAR+1) and (finalReconciliation,YEAR+1)

An example of this is the date of service 07/03/2014, this is valid in the (midYear,2015), (finalReconciliation,2015), and (initial,2016) sweeps.

Note: customer ops doesn't use terminology of initial, midYear, and finalReconciliation. These correspond to September, March, and January sweeps respectively.

Part 3 (Optional):
This has been called "bifurcation," but I don't really understand why. It is purely the act of attaching to the set of HCCs all of their children HCCs in the hierarchical model. This is very useful in the case where an ICD9 of 25050 (corresponds to HCC 18,V12) and someone attempts to also accept an ICD9 of 25000 (corresponds to HCC 19,V12). This is an instance of the first code being for diabetes with complications, and the second is for diabetes without complications. The higher HCC takes precedence (and is compensated more), and the lower HCC is consider a "subtracted" or invalid code.

Example Hierarchy:
Neoplasm 1
8: Metastatic Cancer and Acute Leukemia
9: Lung and Other Severe Cancers
10: Lymphoma and Other Cancers
11: Colorectal, Bladder, and Other Cancers
12: Breast, Prostate, and Other Cancers and Tumors

Finally:

Parts 1 and 2 are combined to provide HCC Descriptors, and there is some inherently known interactions about valid label sets in different payment years that serves as a final filtering of the sets of HCCs. This same operation happens with part 3 codes. The end result is an HCC code, a label set version (model version), sweep, and payment year.

Hoisting APXCAT

See Hoisting (above) for general hoisting.

APXCAT codes are the codes that are determined in our pipeline by event extraction. They are a broad label that map to multiple HCCs. This is often the case for blended models, but it also applies to HCCs that are somewhat related. Part 3 (bifurcation) has never been applied to codes of this type. This is in code here and here.

Example 1.
V12_18 is an APXCAT code that maps to 18,V12 and 18,V22

If we applied the Date 03/21/2014, then the following are valid hoisted HCCs:
(18,V12,initial,2015)
(18,V12,midYear,2015)
(18,V12,finalReconciliation,2015)
(18,V22,initial,2015)
(18,V22,midYear,2015)
(18,V22,finalReconciliation,2015)

If we applied the Date 07/03/2015, then the following are valid hoisted HCCs:
(18,V22,midYear,2016)
(18,V22,finalReconciliation,2016)
(18,V22,initial,2017)

Example 2:
V12_112 is an APXCAT code that maps to 6,V22 and 112,V12 and 115,V22, and the exercise is left to the reader to attach the date of service logic.

Hoisting ICD codes

ICD codes (versions 9 and eventually 10) come from “subtraction” data and user “accept” annotations. Subtraction data comes in the form of CMS_KNOWN events that come from RAPS returns files (and other sources). The important part of ICD codes is that they should always have part 3 of hoisting (expanding to all children HCCs) included because having a code precludes other codes. This is done in code here and here.

Example:
25050 is an ICD9 that maps to 18,V12 and 18,V22

If we applied the Date 07/03/2015, then the following are valid hoisted HCCs (with Part 3):
(18,V22,midYear,2016)
(18,V22,finalReconciliation,2016)
(18,V22,initial,2017)
(19,V22,midYear,2016)
(19,V22,finalReconciliation,2016)
(19,V22,initial,2017)

The existence of code 25050 as evidence (either known or found) precludes the following other ICD9s: 24900, 24901, 24940, 24941, 24950, 24951, 24960, 24961, 24970, 24971, 24980, 24981, 24990, 24991, 25000, 25001, 25002, 25003, 25040, 25041, 25042, 25043, 25050, 25051, 25052, 25053, 25060, 25061, 25062, 25063, 25070, 25071, 25072, 25073, 25080, 25081, 25082, 25083, 25090, 25091, 25092, 25093, 3572, 36201, 36203, 36204, 36205, 36206, 36207, 36641, V5867

Outbound Rules

tbd

Output Report

RAF is the Risk Adjustment Factor for a specific HCC. Since HCCs are “hierarchical” in nature, it r