**Next up - Lew (where AGG opportunity goes to die)**

**Group by Market & Reason Code for NO-AGG condsitions**

**Have AGGOps - re-review lew submissions outside of Lew**

**Measure conversions of disqualified candidaters**

**Questions (COPS):**

**1) Are all "origins" put into Lew and worked by vendors (e.g. ZipRecruiter, T.Turkey, Seymour)?**

**2) Aggregation opportunities are reviewed once before disqualification, classified as "No\_Agg / Claimed" How does the status of "No\_Agg / No\_Agg", accounting for 74% of the total, get classified?**

**3) TLDS (.com & .biz) are considered dupes**

**4) Query for URL vs Domain Duplication**

* **Use: *domain\_id* as *SELECT distinct(domain\_id)*  at the end of the query. Depending on context of the query *distinct(url\_id)***
* **Used Group by to show Multiple Origins - what happens to multiple origins? Combined? Ignored? Which takes precedence? (e.g. 104, Paid List or Rizzo?)**

**5) Human Review:**

* **Redundancy check for human interaction - good, STATE = NO\_AGG presents all the problematic reason codes.. If URL comes in from multiple origins, and is state = NO\_AGG (with those reason codes) and we see it again in the future, it should be reconsidered?**
  + **Behaviour will vary for websites when address is entered from different global locations**

***6) IMpact metrics - - only* and account for 3.6% of the total? - *can we consider re-exampiningthe “*No\_Agg / Claimed"**

**Solution: Start over? - reconsider all "No\_Agg / Claimed" - We have 1) Humans interacting with the system 2) web-based interactions from all over the world, with all types of DNS, which may or may not have reliable name servers - contributing to issues experienced**

***PREVIOUS\_STATE='NO\_AGG' AND STATE='NO\_AGG'* is a duplicate URL in a domain cluster that inherited a review result upon entry to LEW. These do not represent unique URL results and should not be included when pulling metrics for totals (see closest above bullet). See** [**ex3**](https://docs.google.com/spreadsheets/d/1kuSdcri2Rck91-QZzg1E4cpf80GYYx83e3OnMBy-N9Q/edit#gid=1699512105) **for context of state migration of a reviewed domain.**

**Notes & Discovery:**

* [**Workflow LEW uses**](https://wiki.indeed.com/display/AGGC/Lew+State+Migration) to determine whether or not the submission is AGG-able - the "State" of each submission.
* [**Labelling (Reason)**](https://wiki.indeed.com/display/AGGC/Labeling+guide) the vendor provides when qualifying or disqualifying Agg opportunities
  + **Duplicate-signature** (3 cases)
  + TLD's are stripped **- e.g. Jobsite.com & Jobsite.biz are considered duplicates - lew\_domain table contains the domain signatures**
  + **Subdomains**  *(*[*indeed.com*](http://indeed.com/)*,* [*examples.indeed.com*](http://examples.indeed.com/)*)* **-** whitelisted - considered to have different signatures
  + Paths ([indeed.com/jobs](http://indeed.com/jobs), [indeed.com/getjobs](http://indeed.com/getjobs)) - considered to have different signature
* **"State" & "Previous State"** (**Claimed, New, New\_Feed, No\_AGG, Reopened**)
  + When urls are added they are in the NEW state
    - *When a unique URL (across LEW and AggC) is added to LEW it will receive state NEW*
  + Once a url is picked for processing, it moves to the CLAIMED state
    - *When a user opens the extension or finishes review of previous URL the next URL is processed and receives 'CLAIMED' if it is eligible to be reviewed.*
  + If the processing is inconclusive, the url moves to the REOPENED state, from which it can go back to CLAIMED
    - *If user does not give an end result for the URL within 24 hours state REOPENED will be issued. State CLAIMED will follow if URL continues to be eligible for human review.*
  + **If the processing is conclusive, the url will** move to either NEW\_FEED or NO\_AGG and stay there

**Aino adjusted my** [***FYI table***](https://docs.google.com/spreadsheets/d/1eqgG0LSpQ8U_B7LFGU2umofywB8Jl6tMsEu5L2njF44/edit#gid=1177560675) **from the bottom of the email as per above information. We should note though that the *historical reason code revamps have caused duplication and some ambiguity between codes. Manual adjustments could be warranted to correct.***

* **Expired-Jobs and Old-Jobs refer to the same category**
* **One-Job and Onepageagg---one-job *could* refer to the same category**
* **Empty-Job-List-Page and No-Jobs *could* refer to the same category**
* **Miscellaneous, No-Label and Unsure are all ambiguous and could refer to the same category**

130M job updates/day

Job sites have a structure to them that’s critical to find jobs

Start → Navigation → Job List → Job

\*\*Sometimes

a form must be submitted (e.g. drop downs for job type selection)

Job must be pieced together from multipl pages - (e.g. location on previous page)

Components: URL, Title, Company, Application Source Type, Job Type

* Find jobs, that are 1) still available 2) are really jobs 3) extract semantic data
* Internet is mostly not jobs,
* Hard to use machine learning and AI to scrape jobs, easier to create tools and give to people to find the jobs
* Feed: XML or HTML (OR PDF) or Start URL, Navigation, job list, job details o - & rules for how to paginate and metadata targets
* CSS & RegEx selectors to change the data as selected

**Engine running feed definitions:**

* (originally) Strawman architecture: handwritten PHP scripts in data centers
* (durable and able to handle load) RabbitMQ (message queue) - increased network resiliency - messages can queue into mysql if load issues happen
  + Multiple writers - in case one writer goes down can keep writing (paralisation helps with QA for throughput)
  + Across multiple data centers, scalability
* Current architecture is ≈1000 jobs/second
  + Currently only 1 Data Center for writing (is tricky with MySQL)
  + Multiple DC’s for multipel fetch engines
  + Redistribute fetches across job fetch engines
  + Scaler helps proxy pool and domain throttling
* Problems:
  + Flaky sites get redesigned
  + ≈150k sites, 2k feeds/week need fixing
  + Require JS to render jobs - jobs have bad information and we want to look good on indeed so we get formatting information
  + AGGMonitor - runs heuristics and interprets signals for broken feeds - autocreates tickets.
    - “Watson” auto-fixes a feed
* Code:
  + Javascript library progression: was HTMLUnit → Phantom JS → Selenium
  + 467 Selenium notes running chrome or Firefox to
  + 25k feeds - only 125k to go
* Expiration
  + Feeds run every 6 hours - Verify or Remove check
    - Look for expiration text (e.g. “position filled”, “no longer available”
    - No job page anymore?
    - Description update on page?
  + Verification
    - Every time a job is clicked on, on indeed, we check to see if job still exists
    - Randomly check all searchable jobs
    - After jobs is expired, it’s rechecked
    - FACTS:
      * 1% of Agg3 jobs expire on click
      * 3% of XML jobs are expired on click
  + AggDiscovery - out of Tokyo - responsible for monitoring and discovery
    - Rizzo - looks twitter for new jobs
    - Pepe -
    - Syemour - .com crawler
      * Bunsen - categorizes pages
    - Lew - sent to people to look at

Conntying to Agg Systems

Waldorf - Job Write

Statler - Read Jobs from DB

Jobsearch backend

Agg Core:

90% of breaks - pareto curve - different degrees of breaking

Maybe page breaks, but most of the time, it fails one of the rules

E.g. missing non-critical data target

Priority - sponsored feeds get fixed first ($$$)

How much of the internet are we Agging? How many .com domains from lists we’ve purchased

**One-off jobs -”rather not feeds**” - chrome extension, highlight elements on page and submits to Waldorf

E.g. Craigslist - sent us cease and desist

Handling sites that block IP, recaptcha, changing HTML elements

Blocked? -

* we reach out to them to see if we can work soemthing out. Soemtimes we use predictable IP address, we scrape before asking (try not to be annoying), we use proxy’s for others, so we don’t always hit from same IP
* Expiration checks - we rate limit (smooth out requests) so we don’t hammer smaller sites
* We run JS (helps)
* Captcha - not so much

Look for JSON or XML to steal, but mostly we disable those feeds and try agian later - lots of tricks learned over the years

DUPLICATES - AGG - not their problem (e.g. employer + job board) - that’s a search quality problem - agg mirrors everything on internet, de-duped downstream

DRADIS - duplicate of scraped jobs - dradis takes priority

**Using headless chrome with puppeteer** - we’re trying it - headed mode takes a lot of cpu

Selenium requires is and also required for web extension

Google is moving to job search space - asking sites to add meta content to their stes for job crawl - good for us, because we can parse that as well

Account-required sites - like linkedin - create accounts or have automation - we do not agg site, we would not want a seeker to be redirected to a site like that after clicking on job within indeed

AGG - how long between discovery & post - adding loggin now - couple weeks to create feed, takes 1-6 hours for crawler to run, takes 1-6 hours to post to site

We make it difficult for crawlers to pull jobs from indeed

Challenges when you scrape from harder security & blockers -

AGG goes after sources from most countries, with most # jobs first