People Data Labs

OUR DATA BUILD PROCESS

The three-step process of sourcing, cleaning, and maintaining our robust datasets.

Ever wonder what it takes to create a dataset of nearly **3 billion person records** and **18 million companies**?

Customers are often interested in understanding this process, whether it's for assessing data quality, ensuring regulatory compliance, or even just satisfying curiosity.

In this document, we will take a behind-the-scenes look at the way People Data Labs generates and maintains our production datasets, including the key steps and our rationale for some of our engineering decisions. Specifically, we'll answer the following questions:

- Where do we get our data from?
- How and why do we standardize our data?
- How do we ensure our data is correct?

A general overview of our data build process looks like this:

1. We take in raw data from a variety of sources
2. We standardize and deduplicate the raw data to integrate it into our dataset
3. At every step along the way, we perform quality assurance checks to ensure quality and compliance

Each quarter, we release an updated production dataset with improved data and features. We dedicate significant engineering resources towards each release cycle to ensure our dataset contains the most up-to-date and highest quality data, while also ensuring compliance with regulatory policies.

Now, let's dig into each of these steps.

Data Sources

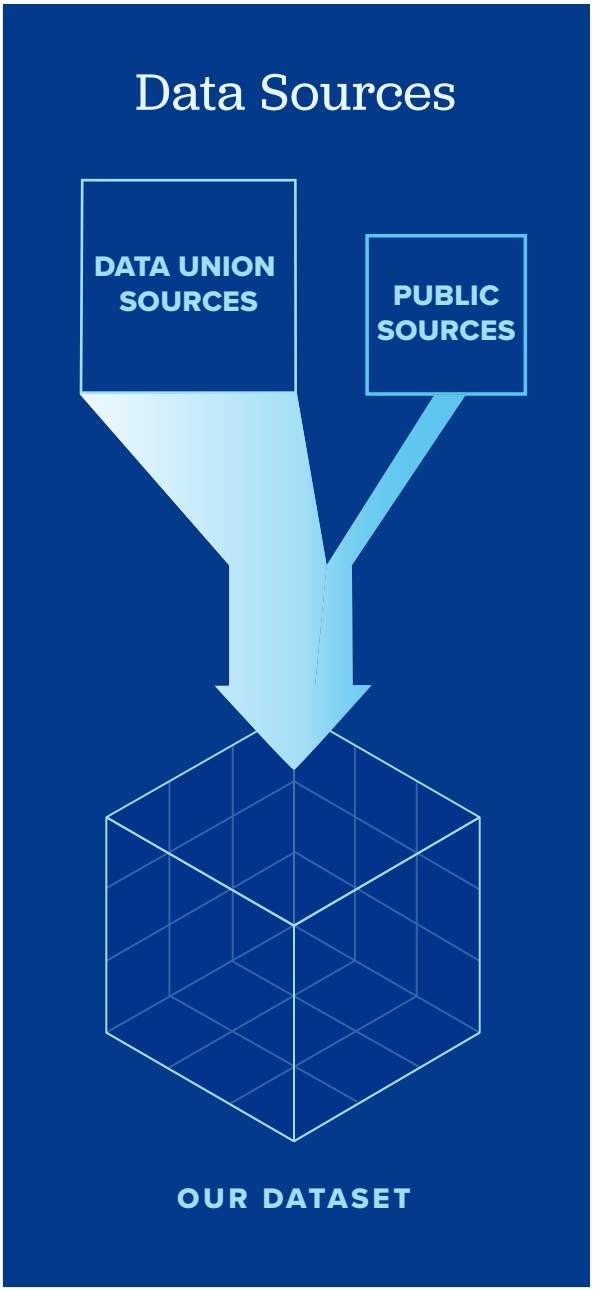
The first step in our data build process is sourcing raw data, which is how we incorporate new records and fields into our dataset. Our entire production dataset is actually composed of thousands of individual data sources, of which there are two categories of data sources in our dataset: Data Union sources and public data sources.

Data Union Sources

Through transparent agreements customers and other data partners, we created a data-sharing community that we refer to as our Data Union. Sources we ingest via the data union supply the large majority of the person data in our dataset. Currently, we are ingesting ~45 million records every month from thousands of Data Union sources. Data Union sources help us ensure that we stay within global compliance regulations -- each partner attests that they have provided any required notices and obtained any required consents concerning the collection, use, processing, transfer, and disclosure of personal information – something that we confirm and verify.

Public Sources

Our second primary source of data, apart from the Data Union, are public data sources. We define public sources as a source of data that is available to anyone in the world with an internet connection. We crawl the web to extract information in the same way Google, Bing, or any other search engine does. Public sources generally provide us with information on companies, schools, and locations, as well as an individual's work history, education, and more.



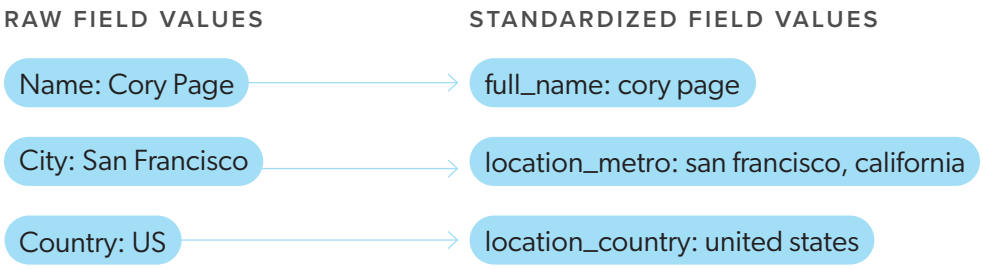
Data Standardization & Deduplication

After sourcing raw data, our next step is to integrate it into our production dataset, which involves standardizing the data formats and merging data for duplicate records together.

Standardization

Illustration of the data standardization process.

Raw field values are processed and standardized with consistent formatting, spelling and for some fields, canonicalized values.



Parsing, or standardizing, or cleaning the data means converting raw data into a standardized format. What this format is and our methods vary by field, but the key is that every field in our data has some baseline cleaning. This cleaning might mean simply lower-casing the data and stripping whitespace (e.g. turning " Cory " into "cory"), or confirming that the data field follows a specific format (like local@domain.com for an email).

There are many reasons why standardization is important. Internally, as a data provider, standardizing our data enables us to seamlessly integrate new sources of data into our full production dataset. In particular, it helps with merging and updating records together as part of our entity resolution/deduplication process, by making our data fields as clean as possible. The benefits of standardization even extend to customers as well, since creating data standards provides an easy and consistent way to understand and consume our data.

In fact, standardization is a key reason that allows our products (such as our Enrichment and Search APIs) to work the way they do. For example, all of the cleaning we do during our standardization process, we also do in the Enrichment API. This means that our Enrichment API logic is functionally taking a customer input and directly tapping into our cleaning and entity resolution expertise to generate a match. This is essential because for us to be able to match an Enrichment or Search request for *Sean Thorne in SF* against our record for *sean thorne in san francisco,* we need to have standardization.

For those who are curious or would like to understand and leverage our data standards, we publicly documented our standardization format for each field in our Person Manual, which summarizes the possible values for each field and our persistence commitments to ensure forward and backward compatibility. Additionally, for several key fields (referred to as canonical fields) we have also provided datasets of the enumerated possible values that we have defined (such as for countries, majors, etc.).

Deduplication (Entity Resolution)

After standardization, another key transformation we perform during our data build process is what we refer to as deduplication or entity resolution.

In simple terms, we want to know whether each record we ingest belongs to an existing profile in our dataset or is a completely new profile that we should create. The challenge, however, is that we have over 2.5 billion records in our database and we add hundreds of millions (if not billions) of records each quarter that we need to merge into our existing database. To directly compare billions of records is an impossible task.

Assuming we have three billion total records we want to compare, we would need to make 9.0e18 comparisons. Even though each comparison might take one millisecond, in total the whole job would take 285,198,882 years!

Instead, we try to isolate target groups of records together by creating "blocks" — small groups of records that share a common key. By picking a key (for instance: full name) and sorting records into each key, we significantly decrease the amount of time we need to make comparisons. Three billion records at 1ms/record will take 50 minutes to group. We end up with small groups (say the max size is 1,000 records), which can be constructed in a matter of minutes, rather than hundreds of millions of years.

We block on a variety of keys to maximize our number of merges, focusing on fields that are the most likely to be unique and generate matches. Finally, we have two types of techniques that we use to determine whether or not to merge records: deterministic and probabilistic methods.

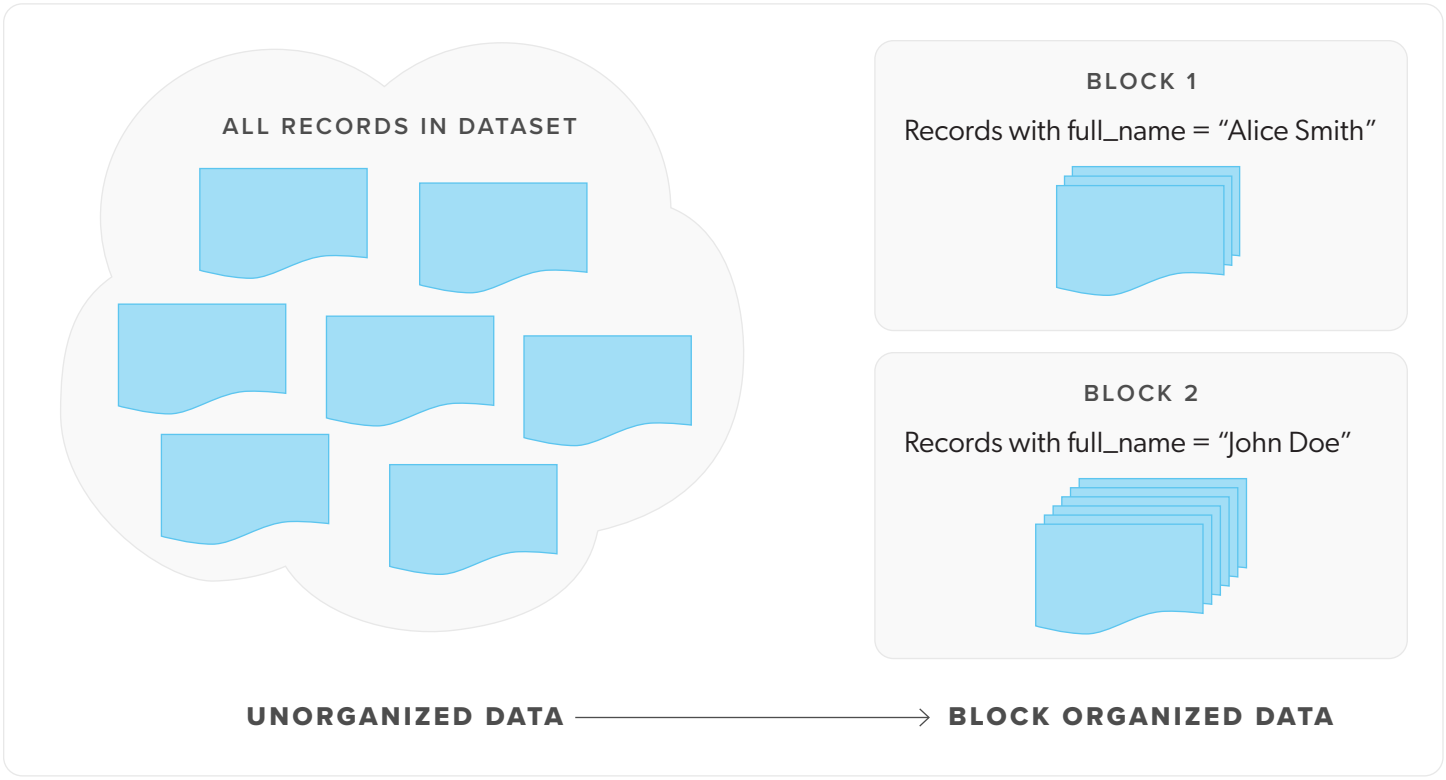


Figure illustrating blocking process which reduces the time needed to compare records during the entity merging process. The blocking keys in this toy example are just the full_name field.

Merging by Deterministic Methods

The deterministic method is very easy to understand. We have created a defined set of rules that, when true, indicate to us that we can 100% confirm two records are the same. An example of a rule might be: **We know that two records that share a name and phone number are the same, regardless of other information.**

In order to determine these rules, we've done massive amounts of data build QA. For the most part, we don't make changes to our deterministic matching logic. There's not a lot of lift to be gained by changing our merging rules without significantly decreasing the quality of our data. This means, to link more records together deterministically, we need more data that gives us more blocking keys and matchable data. If two records don't share enough common data, we can't deterministically merge them.

Merging by Probabilistic Methods

Whereas our deterministic methods are relatively fixed, our probabilistic methods are always evolving. In fact, when you hear us say we are *adding more linkages,* this generally means we are typically improving our probabilistic methods.

Our probabilistic methods are comprised of two phases: linkage creation and linkage thresholding.

Linkage Creation

In this stage, we have a set of records we want to evaluate whether or not to merge. Linkage creation means finding the most likely matching record in our data (i.e. within the same block) for each of these records, and computing a likelihood score to rate how good of a match these two records are. To find the most likely record, we perform a search nearly identical to what is done in our Person Enrichment API, and generate the likelihood score using a variant of TF/IDF, which uses all the fields in our data to create this likelihood.

An example to summarize what this process is doing is as follows: *how likely is it that there are two people named Hayden Conrad, living in San Francisco, with the title Assistant to the Regional Manager?*

Linkage Thresholding

After creating and scoring the linkages in the previous step, we create a "score threshold" we are comfortable with calling a 100% linkage. We typically check thousands of potential linkages by hand whenever we make changes to the linkage creation logic. This gives us a mapping between ranges of scores and the probability that those records are the same person. We then pick the range we want and merge those records.

Unlike deterministic linkage methods, we don't necessarily need more data to create more probabilistic linkages. We can add linkages by manipulating either of the two steps: we could improve our linkage creation logic or decrease our minimum score and allow more of our created linkages to actually merge. We tend to not decrease our minimum score unless we're highly confident in that score decrease.

Quality Assurance

Finally, let's discuss how Quality Assurance is handled throughout our data build process.

Data Ingestion QA

Having good data starts with the data we take in. Before we ingest a source, we test the quality of the data using both automated processes and hand checking.

Automation

Automation helps us identify glaring issues like false positive linkages in the data source. We view our existing dataset as a source of truth. This means that if there's a contradiction between our data and the new data source, we've either identified a quality issue in our data or in the source.

We tag sources as either trusted or untrusted based on the quality of their linkages. If a source is trusted, we use it as part of our entity resolution (merging) process. If a source is untrusted, we append its data when there is a possible linkage, but we don't expose that linkage to customers unless multiple untrusted sources agree. Sources should be very close to 100% accurate to be marked as trusted.

Hand Checking

We employ hand checking to review data that's harder to evaluate via automation (i.e. job titles). If a job title doesn't exactly match our existing data, that doesn't necessarily mean it's wrong, so we use hand checking to see if anything odd about the source stands out. We'll check full records from the source, looking up people's social profiles, addresses, etc. and comparing data.

Why QA?

First of all, quality assurance is really important. The consequences of poor QA can be quite negative to both us and our customers.

For example, if there is a data quality issue, we may have to rebuild the data which can take up to a week. This costs money, computational resources and delays releases that many customers wait and rely on. Additionally, these issues have the potential to impact a massive amount of customers. When we do a data release, we are releasing to both the API and to our license customers and each has their own nuances that make QA'ing our data before we release crucial.

As an additional check, we also aggregate the common values for every data field in the source. For example, if we see that the name *tim* appears an uncharacteristically high number of times, that might be a cause for suspicion. Some other relevant examples include:

- Data sources inferring emails (we see the same email appearing many times across records)
- A "default value" for a field. For example, often when a birth day // month is unknown, a data source will fill it in with 01 or some other default value.
- Bias in the data that makes it more // less valuable (all records in this source are in Vietnam)

Standardization QA

Before we push the data into the build, we also run it through our data standardization process, so we can check how fields parse. Some sources have an uncharacteristically low rate of standardization for some fields, which is an indicator of low-quality data. For example, if many people in the data source don't have a complete last name (just an initial) or fill rates for a field we deemed valuable turn out to be very low, we may decide to drop the source. This happens more frequently than you might think, particularly if you don't have a good random sample of data for your initial QA. Some sources tend to be amalgamations of multiple data files, where for example, the top 1mm people might be great, but then the next 99mm are not so great. To mitigate this, we split our data sources into part files with a maximum file size and then randomly sample across all the files from the start.

If we make any logical changes to how we standardize a data field (i.e. change our logic for matching locations, fix a bug in the name parser, etc.), we test it by running the cleaner code on ~50k unique possible inputs. We store the legacy output for that test and run a comparison. We then hand check to ensure any changes between the old and the new output are intended, or that a change is a net positive. As an example, often a change to our cleaning logic might lead to 8% of records improving, but 1% losing data. We generally see multiplicative improvements (i.e. more than 2-3x) in the parsing logic as a good thing, even if it causes a few records to be adversely impacted.

Deduplication QA

Entity resolution (merging) is a particularly risky process, and it is quite easy to do real damage if our process is over-aggressive. Just like we check our data sources for false-positive merges and discard them if they have too many errors, we also hand check our own data. Any time we make logical changes to our record linkage we risk creating *Frankenstein records*.

PDL definition: *Frankenstein record* is a single data record that represents multiple people. Like Frankenstein's monster, it is composed of parts of multiple people (person A's email, person B's Linkedin, etc.). These records negatively impact customers across all use cases.

We are extremely sensitive to this and whether this means performing hand checks on the source of truth records, or pulling stats, we check everything we can to ensure we don't run into issues here. We also tag Frankenstein records in our data. Once a record has a certain amount of PII, we conservatively determine it is a Frankenstein record. We delete these records from the data licenses and only surface these records as an API match if there is no other possible record we could surface.

Final Build QA

Once a data build is complete, there are multiple ways in which we try to ensure the quality of our final dataset. We use a variety of techniques ranging from spot-checking segments of our data (both randomly and ones known or predicted to be problematic) as well as high-level aggregate checks. One simple example of an aggregate check performed by our Data Pipeline team is described below.

Example Data Quality Check: Aggregate Statistics

We post a sample of the data and some high level stats so that our data pipeline team can take a closer look. Stats are similar to what we expose to customers in our dataset stats as well as in our recent quarterly release notes. We want to ensure there's no unexpected increases or decreases in linkages or record counts. For example, if we had a bug in our major parser, it might dramatically decrease the number of majors in the data, in which case we would flag this and begin an investigation.

Release QA

There are many more tests that our data pipeline team might also perform, but assuming they have given the OK on their quality assurance evaluation, they pass the data to the applications team, who will begin testing the data for license delivery and API release. This involves indexing the data to the staging API, running some test license deliveries and assessing the output and fill rates to provide a final internal evaluation of the build stats before releasing.

Customer QA

In general, despite our best attempts at internal QA, there will always be bugs that leak out simply due to the size of our dataset. This means our customers also play an important role in ensuring that our data quality meets their expectations and standards. For the most part, customers report bugs that are minor but often these bug reports are actually quite valuable as potential symptoms of wider problems. Therefore, we take customer feedback seriously and it is to our benefit to do so.

We have thousands of customers with a minimum of one engineer looking at our data. That is 10x the size of our current engineering team and 70x the size of our data pipeline or applications team. This is a key value proposition for our data as well, since fixes brought up by any individual customer raise the data quality across the dataset for all customers.

Future of QA

QA is an area where startups are generally weak, but because of how tightly our product correlates to our customer's success and how integral our product is to customers, this is something we aim to be exceptionally good at. On the data side, we face a unique problem that most other companies don't face. As we grow out and mature our QA, we are confident that we will be building out data pipeline features that rival or surpass the best in the industry.

We hope this detailed description of our build process illustrates both the complexity and care required to productize a substantial dataset.

We believe that the result of our data build process provides our customers with a unique value proposition for data that is accurate, up-to-date, compliant, and continuously improving.

Interested in learning more? Get in touch with our team of data experts to see how our data can power your business.