Say No to Fussy Match for Multiple Customer Dataset Matching

A Scalable and Cost-Efficient Cloud Based Data Science Solution for Multiple Datasets Matching

Bobo Hu†
Business Technology
Sysco
Houston, TX, US
hu.bobo@corp.sysco.com

Varun Kumar
Business Technology
Sysco
Houston, TX, US
Kumar.Varun@corp.sysco.com

ABSTRACT

With the rapid development of social networking platform, maturing Points-of-Interest (POI) database, and crowd-sourced review systems, there is a growing demand to utilize third-party data for data enrichment. Such data enrichment will provide a more comprehensive understanding to their existing or potential customers for many different industries. From the enriched dataset, business is able to derive more informed decision for their strategic planning, obtain valuable business insight without significant financial and time costs. After third party datasets being dumped into a datalake or data warehouse, they are often ingested as discrete datasets with individual primary keys. However, such raw information does not provide significant useful information for data analytics, and record matching between different datasets with internal dataset is the vital first step to unlock the full potential of data enrichment.

In the past, Fuzzy Match method based on measuring customer name and address similarities is most widely used for matching internal and external datasets. However there are several drawbacks associated with this method: (1) Lack of consistency and accuracy: It is a co-occurrence statistical approach by using a hard threshold to determine record pair matching likelihood. It is not a rule-based supervised ML algorithm that has clear accuracy measurement for matching and consistent matching criteria based on derived data features. (2) High memory consumption: A query based Fuzzy Match searches through all available records for matching candidates, and this will require a large amount of memory and heap space for large datasets.

Such drawbacks are proved as detrimental for further data science analysis that using information derived from the ingested dataset. False positive matches will negatively impact downstream analytic software credibility, while high percentage of false negative will miss valuable market insights and cannot achieve the full potential of data enrichment.

In this paper, we proposed an applied data science framework of multiple datasets matching, that has been extensively deployed, and approved to be a scalable, cost-effective and ML algorithm-based cloud solution. The architecture separates the heavy computational task of record matching from the data ETL and catalogue pipeline, and used most suitable and cost effective cloud services for these two tasks.

For the record matching task, we deployed an in-house two-stage, supervised ML model. The model ROC-AUC accuracy varies between 99.58% to 99.64% depends on the amount of information provided by input data. The target used to train the ML algorithm is generated from a semi-supervised learning approach to reduce the turn-around time for data labeling, while improved the label accuracy by avoiding a manual label process. The two-stage ML model is used to reduce the memory consumption of record comparison process by gradually increase number of records for comparison with different constraint scopes. Because there are several different databases need to be matched parallelly, so input records are partitioned by different databases, and a fleet of M4 large EC2 instances are adopted. To launch and continuously monitor the EC2 instance health, AWS EC2 Fleet service are used. For data engineering after ML model computation, schema interference and table hosting, AWS's glue workflow is used for its versatile data pipeline and fast data catalog and scheme refinement.

KEYWORDS

Data Enrichment, ML model for matching multiple databases, Cost-efficient Scalable Cloud Architecture

1 Current Challenges for Data Enrichment

There are two challenges for current Data Enrichment workflow: (1). How to develop a supervised ML algorithm that is computationally efficient and highly accurate, so third-party data records can be accurately linked with internal data. Without an accurate algorithm, the business value and enterprise analytics built on top of the enriched data is questionable, and therefore cannot provide useful guidance for decision makers. (2). How to design a robust and cost-effective architecture to keep the process running on a fixed schedule, so the business information provided by various data sources are up-to-date and valuable.

In terms of challenge 1, customer data, no matter how detailed, is fundamentally a snapshot in time. Business reputation, service quality and ownership can alter over time. Accurate detection of such information will provide valuable insights for supply chain management and customer services. Given the possibility of all these changes, data enrichment processes need to run on a continuous basis with a quantitative supervised ML algorithm. There are several cloud database services already providing some

primitive data enrichment services, e.g. AWS FindMatches, MongoDB Single View. MongoDB Single View is based on query language to filter and match records. While AWS FindMatches has the ML capability to develop a supervised ML model after rounds of human data labeling. However due to the size of the different databases need to be matched for some large business entities (where a single database may contain records up to 10 million), the computational time for such algorithm can be lengthy, and the relatively more expensive AWS Findmatches service is not the most cost-effective solution for such computational intensive task. This paper describes an in-house highly accurate supervised ML model that can be effectively deployed on the more cost-effective virtual machine (AWS EC2 or docker). More specifically, a two stage ML model is used for the first time for third-party data enrichment. After software prototyping and ML algorithm accuracy benchmark, the software application is saved as artifact files in AWS S3 bucket. AWS Code Pipeline is initiated to trigger the ML algorithm with a fixed schedule to generate up-to-date business information at a minimal cost. The final output is QC tested and delivered with AWS glue crawler. The final output of the application is directly served with Redshift Spectrum for further downstream consumption.

For ML algorithm, a two-stage cascade model is adopted to improve the match percentage rate between datasets, while saving unnecessary computational efforts by narrowing down the potential match pairs. The first stage model targets on records with same Zipcode and address number (not the entirely address including both address street and number). This assumption greatly reduced the number of potential match pairs being compared, and almost 85% of match records were generated from this stage. This approach significantly reduces the memory and computation costs associated with other matching algorithm, e.g. fuzzy matching with O(n) running cost. The second stage process including all unmatched records left from the first stage. Comparisons for similar records are performed for these only with same Zipcode(but different address number) at this stage, since the assumption is that records may contain unexpected human errors on address or business might recently move to a new physical location. The number of pairs for comparison significantly increased at this stage. However, this stage is necessary for matching any records that may contain human errors, since it improved the match percentage by another 15%. The XGBoost model is used for both stages of pairing processes. The model parameters and weights are adjusted for the best Roc-Auc accuracy at each stage. Key features extracted from data for XGBoost classifier model are: similarities between Name, address, address number, telephone number and customer type classification. For string type features, Jarowrinkler distance is measured to assess similarities between these short phrases. For numerical features, gaussian distribution based algorithm is used to determine the similarities between these numbers.

The second challenge of capable of serving data in fast turnaround is addressed by fully utilizing the AWS Glue service to directly ingest the JSON documents produced from the ML algorithm as Parquet format, and then using dynamic schema interference of this service to iterate on the self-defined schema without the rigid definition process. This fast turnaround capability is vital for such data enrichment application. Because the downstream data analytics may iterate on the first schema frequently, depends on the analytic topic. Various third-party databases with different focus can be introduced for matching, and each comes with different key and data types. How to adapt the output schema in quick turnaround so that it can satisfy needs of different user case is always challenging. The data post processing and pipeline solution proposed in this can evolve the document-based output according to different business needs, and therefore significantly accelerate the data delivery speed for further analysis. Since this task is relatively less computationally intensive, more mature cloud service can be utilized to accelerate the data sharing process without incurring large cost. When different third-party data are used for data augmentation, the matching algorithm is paralleled to run on individual EC2 machine to accelerate computation. The in-house built algorithm goes through minimal variation for data ingestion from various data sources and can be easily scaled up for parallelization.

2 Methods and Key Results

Some basic data cleansing is performed on both internal data and external data. First, lowercasing all text data, remove special characters from strings to reduce sparsity of embedded words, and strip empty spaces before and after strings. Extensive noise removal is also applied, including punctuation removal, domain specific keyword removal. Then NLP standardization methods are applied next, including stemming and Lemmatization. For this record comparison task, text information comes as short phrase, often in abbreviation format, so there is no need to remove stopping words. Text normalization is applied to address features, since many address data are noisy, and vary a lot based on peoples' spelling habits. A dictionary type mapping is used for cleansing non-standard spelling.

The unique difference between this task and other NLP taxonomy task is that the similarity scores will be calculated for both text information, and numerical features, such as zip-code, phone, address number. Without a comprehensive assessment with both text and numerical features, the accuracy of record matching will be sacrificed, due to limited information. The cleansing methods for numerical features are: special character removal, extract the first five digit of zip code for standardization, remove country code from phone number and reformat the phone number to normalize data. Remove domain specific information in front of address number to improve consistency.

Another practical observation is that there is very limited benefit to adopt text augmentation and embedding that are prevalent for deep learning NLP approach in this record matching task. This is because text information for records are noun types, and the string length associated with records is short.

After hand-crafted these features for each record, the similarities of different records can be calculated for numerical and string features respectively.