International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 3, May-June-2026, ISSN (Online): 2395-566X

Automated Parallel Hybrid Data Extraction and Entity Resolution for Sports Data Aggregation: Architecture, Challenges, and Trade-offs

Olalekan Oluyinka
Ladoke Akintola University of Technology Ogbomoso, Nigeria olalekan.oluyinka@gmail.com

Abstract- This paper presents the design and implementation of an architecture for automated hybrid data extraction, integration, and entity resolution for sports aggregation. The system consolidates inconsistent records from multiple heterogeneous sources into a centralized, deduplicated interface for sports event discovery and streaming access. Data is extracted in real time across seven heterogeneous sources and directly ingested in the automated pipeline. A multi-step entity resolution algorithm, combined with data pre-processing within a Single Source of Truth (SSOT) framework transforms heterogeneous data into a unified, deduplicated index. The architecture employs edge caching and batching to reduce latency and improve operational performance in constrained environments. A prototype further demonstrates the practicality of automated multi-source sports event aggregation through entity resolution.

Keywords – entity resolution, Levenshtein distance, record linkage, sports data aggregation

I. INTRODUCTION

Sports aggregator platforms function as centralized hubs for collecting data on match fixtures, real-time updates, and external stream sources. They collect data from various sources to curate on their platforms for easy access for end users. Automating sports data collection, aggregation and processing is inherently challenging due to the fragmented and heterogeneous nature of data distributed across disparate sources. The primary challenge in aggregating data from multiple sources is inconsistency in data formats, including the absence of a common key (Christen 2012, Shearer 2024), non-standardized names arising from abbreviations and nicknames, and variations in language.

For a single sporting event, different sources may represent the same fixture using diverse naming conventions, including abbreviations, supplementary title descriptors, variations in language formatting, and different orders of the competing teams. Since each event is also associated with streaming URLs, directly merging records based solely on their raw titles without adequate normalization and entity resolution will cause identical events to be treated as distinct records, resulting in severe user interface fragmentation and the presentation of duplicate event listings to end users.

Aggregator platforms that provide access to authorized sports and media content typically integrate streaming availability information by redirecting users to licensed providers through official watch links. JustWatch (www.justwatch.com) is a notable example for movies and TV shows. However, these platforms do encounter challenges related to inconsistencies, and discrepancies across heterogeneous datasets from multiple sources. Sports aggregation systems that integrate information from multiple web-based and third-party sources must address substantial challenges arising from fragmented, unstructured, and inconsistently formatted data. Variations in event naming conventions, fixture formats, and source reliability complicate automated integration and synchronization processes. Consequently, scalable aggregation architectures require robust extraction, normalization, and entity resolution mechanisms to transform heterogeneous records representing the same entity into a unified index suitable for discovery and presentation.

II. LITERATURE REVIEW

2.1 API and Web Scraping

The first stage of data handling is the actual data acquisition. Data can be acquired in different ways depending on the specific environment in which the data is collected and processed (López & Ojanperä 2025). Many platforms provide direct Application Programming Interfaces (APIs), a set of structured HTTP requests yielding clean JSON data. Web scraping is the automated extraction of unstructured data from webpages. Web scraping enables the

extraction of data embedded within HTML and converts it from unstructured content into structured, usable formats (Hassan et al. 2025). Web scraping is necessary when data is not available in machine readable format like JSON or XML, or when an API is out of reach, limited, or not offered (Hassan et al. 2025, Wickham et al. 2023). Traditional scraping relies heavily on Document Object Model (DOM) parsing and Regular Expressions (Regex). Regex is one of the earliest and most fundamental approaches to web scraping. They remain a central and indispensable tool in modern data extraction, valued for their speed, precision, and versatility (Hassan et al. 2025).

Both web scraping and APIs enable fast access to large volumes of data across the internet. However, API when compared with web scraping provides more reliable data quality and should be adopted where possible (Wickham et al. 2023). Also, the ethical and legal landscape surrounding web scraping is nuanced and frequently uncertain. Some web scraping practices raise legitimate concerns related to copyright, privacy, and unauthorized use of data. At the same time, web scraping serves as a catalyst for technological innovation and the generation of critical data-driven insights (Hassan et al. 2025).

2.2 Entity Resolution

Entity Resolution (ER) is a technique for identifying multiple data records that refer to the same real-world entity (Shearer 2024, Talburt 2011) and to merge the records into a new combined record (Christen 2012). An entity resolution system should link two references if and only if the references are equivalent (Talburt & Zhou 2015). ER helps to match disparate, heterogeneous data together when unique identifiers which allow a common context to be quickly established between two records are not available (Shearer 2024). Two important string matching methods used in determining the similarity between values are exact and fuzzy matching (Christen 2012, Talburt 2011).

Exact matching is the process of comparing record pairs using a set of functions that permit only exact similarities. It compares two attributes and returns a similarity value of 1 if both attributes being compared are identical or return a value of 0 if both attributes are different (Christen 2012, Talburt 2011).

Fuzzy string matching is often employed in order to compare and check that two records describe the same entity. Fuzzy string comparison methods include Levenshtein distance, Jaro similarity, Jaro-Winkler similarity, phonetic matching, Damerau-Levenshtein (a variation of Levenshtein distance), Smith-Waterman, Monge-Elkan, Extended Jaccard, Q-gram based string comparison and various other methods which are specific to certain domains (Christen 2012, Talburt 2011). Fuzzy string comparison methods typically return a normalized numerical similarity score in the range [0, 1], where higher values indicate a greater degree of similarity between the two attribute values. A similarity score of 0 indicates that the attribute values being compared are completely different while a score of 1 corresponds to exact match.

Levenshtein distance is the foundational and most common fuzzy string-matching technique. It assigns every single edit operation (insertion, deletion, substitution) an equal weight of 1. Other variations of the Levenshtein distance allocate different weights for different edit operations, for instance, a weight of 1 for insertions and deletions, and 0.5 for substitutions while more advanced approaches learn the optimal edit weights directly from training data (Christen 2012). While other fuzzy matching methods have their own peculiar strengths, Levenshtein distance is the most versatile (Bilenko & Mooney 2003, Shearer 2024). It calculates the minimum number of single-character edits or operations required to transform one string into another (Levenshtein 1966). The similarity S(a,b) between two strings, a and b (of length |a| and |b| respectively), is mathematical defined as:

$$S(a,b) = 1 - \text{LevenshteinDistance}(a,b) / (\max(|a|, |b|) \quad (1)$$

2.3 Related Work

(Jiang et al. 2014) developed a rule-based module for retrieving deduplicated search results from five leading biomedical databases, where missing data, erroneous entries and inconsistent recording formats are prominent issues of data integration. Their method uses text approximation techniques to sequentially compare articles with the same publication year to find potential matching record pairs.

In the approach of (Tsuruoka et al. 2007), a machine learning-based classification model was used in the learning of the measure of similarity of strings directly

from a gene and protein dictionary containing variants of names as well as canonical names.

(Mamun et al. 2014) reported the development of an efficient sequential and parallel algorithm for record linkage across any number of datasets. The algorithm employs edit distance, utilizes hierarchical clustering as a foundation and radix sorting on specific attributes to eliminate identical records.

(Zehtaban et al. 2016) developed a framework for identifying similarities between computer-aided design (CAD) models to facilitate design reuse and reduce development time. Their approach utilizes a classification coding system combined with distance functions to filter and retrieve existing designs that match new product parameters, effectively transferring manufacturing knowledge to the conceptual design phase.

III. METHODOLOGY

3.1 Data Extraction

This study utilizes dynamic data from real-time APIs and web pages through automated ingestion pipelines. Since these sources are continuously updated, the exact records evolve over time and cannot be represented as a single immutable snapshot.

A hybrid extraction approach is employed for data acquisition from seven heterogeneous sources. Two sources provide structured JSON-based APIs, enabling the system to directly retrieve structured and machine-readable data. The remaining sources do not provide accessible APIs and therefore require automated scraping of the dynamic HTML based content. Both extraction methods are executed in parallel to maximize processing efficiency. The extracted data primarily consist of event titles and their associated Uniform Resource Locators (URLs), which are subsequently ingested into the automated data pipeline.

3.2 System Architecture

3.2.1 The Single Source of Truth (SSOT) Model

The system architecture designates an external, open, crowd-sourced sports database (TheSportsDB, www.thesportsdb.com) as the strict Single Source of Truth (SSOT) for canonical metadata related to sports events, including kick-off time, title, team names and logos, competition etc.

All raw data ingested from API and third-party websites (discussed in 3.1) is treated as a pool of available “candidates”, which are only permitted to render in the user interface (UI) if they successfully match with and are merged to an existing SSOT event. These “candidates” are systematically evaluated against events across multiple partitions of the SSOT (i.e., leagues or competitions). Expanding the number of target database partitions broadens the search scope, thereby increasing the likelihood of identifying a true match for a given source record.

However, this comes at the cost of greater computational overhead and increased processing time.

When a match is identified, the extracted record is merged with a corresponding SSOT event. This consolidation ensures that the end user is presented with a clean, deduplicated interface, in which a single event representation aggregates multiple viewing options. For example, if five different sources provide records for the same event, the user is presented with a single match card that embeds and displays the available streaming options.

```
const LEAGUES = [

  "4328-English-Premier-League",
  "4482-FA-Cup",
  "4571-fa-community-shield",
  "4335-spanish-la-liga",
  "4332-italian-serie-a",
  "4331-german-bundesliga",

  "4334-french-ligue-1",
  "4337-dutch-eredivisie",
  "4570-efl-cup",
  "4483-copa-del-rey",
  "4511-supercopa-de-espana",
  "4507-supercopa-de-espana",
  "4506-coppa-italia",
  "4485-dfb-pokal",
  "4484-coupe-de-france",
  "4901-french-trophée-des-champions",
  "4902-dutch-knvb-cup",

  "4329-english-league-championship",
  "4399-german-2.-bundesliga",
  "4400-spanish-la-liga-2",
  "4394-italian-serie-b",
  "4401-french-ligue-2",

  "4480-uefa-champions-league",
  "4481-uefa-europa-league",
  "4490-uefa-nations-league",
  "4502-uefa-european-championships",
  "4512-uefa-super-cup",
  "4524-uefa-cup",
  "5519-uefa-european-championships-qualifying",
  "5071-uefa-conference-league",
  "4889-uefa-womens-champions-league"

];
```

Figure 1. Different partitions of the SSOT database. The data contained within the database partitions are retrieved through the TheSportsDB API.

3.2.2 Data Pre-processing

Data clean-up is the first stage of data pre-processing. The source records are pre-processed before attempting to match the extracted records to an SSOT event.

(i) String Normalization: Strings are converted to lowercase. Also, the titles of the extracted records which typically includes the names of the competing teams are separated into two components by splitting the title at common delimiters (vs, vs., x, v, @, -) using a regex delimiter.

(ii) Tokenization: This is the process of splitting a piece of text into smaller units known as tokens. A token is a specific instance of a character sequence treated as a meaningful semantic unit for processing (Manning et al. 2008). In this work, tokens are generated after applying various normalization techniques.

(iii) Noise Removal and Token Filtering: Generic descriptors (e.g., “fc”, “the”, “united”, “club”, “city”, “real”, “athletic”, “town”, “olympique”, “west”, “north”, “u17” etc) are stripped from the strings to isolate the core team identifiers. These are extremely common texts which

would be of little value in helping in the record matching process (Manning et al. 2008). This approach reduces the number of tokens needed to be iterated over and evaluated during entity resolution. Non-alphanumeric characters are also removed, and words shorter than three characters are discarded.

(iv) Canonical Mapping (Standardization): A lookup table is setup and used to standardize character sequences so that tokens can match despite their superficial differences (Manning et al. 2008). Canonical mapping can be updated and it provides the necessary flexibility to changing data needs. Although it might be time-consuming depending on the data requirements, the effort is justified by the gains that can be obtained in the matching quality (Christen 2012).

In this work, a predefined map expands common team name abbreviations and nicknames to the target forms of the SSOT (e.g., “bvb” to "borussia dortmund", “psg” to “paris sg”, internazionale to inter milan). Arabic team names are also mapped to the expected canonical forms defined by the SSOT e.g., "أرسنال" to "arsenal". This is subsequently used for tokenization and Levenshtein distance algorithm.

```
const englishMappings = {
  "olympiquelyonnais": "lyon",
  "internazionale": "inter milan",
  "psg": "paris sg",
  "spurs": "tottenham Hotspur",
  "wolves": "wolverhampton wanderers",
  "atleti": "atlético madrid",
  "larissa": "ael",
  "bvb": "borussia dortmund",
  "az": "az alkmaar",
  "cahn": "Công An Hà Nội",
  "man utd": "manchester united",
  "man city": "manchester city"
};
```

Figure 2. Mapping (Look-Up Table)

3.2.3 Event-Driven Data Pipeline and Edge Caching

The architecture leverages event triggers to automate data extraction and entity resolution, ensuring efficient sports data aggregation.

(i) Data Persistence Layer: The system utilizes Cloudflare Workers KV (free plan) as a high-performance, low latency data store.

(ii) Trigger Mechanism and Initialization: The set-up is activated by the first HTTP request. If the cache is empty or expired upon the first request of the day, this triggers the automated extraction sequence across all defined sources. Once the data is fetched, normalized, processed in the entity resolution engine and merged to the SSOT event, the resulting JSON payload is persisted to the Cloudflare KV. The first-user trigger can be initiated by a human user opening the application, a headless browser executing a scheduled cron job on a remote server, or automated shortcuts on mobile/desktop devices at a predefined time (e.g., 00:00 UTC). This ensures the edge cache is perpetually "warm", entirely eliminating the cold start penalty for subsequent users. Herein, a cron job was set up on GitHub Actions to trigger the system at 00:01 UTC automatically, ensuring the KV cache is "warm" for all subsequent users.

(iii) Edge Caching: For all subsequent users, the system bypasses the fetching or scraping logic and serves data directly from the cache, significantly

reducing server load and providing near-instantaneous load times.

3.2.4 Entity Resolution

The system executes a multi-step matching process to map tokens to the SSOT and determine whether records match.

1. Direct and Swapped Checks

To account for inconsistencies in how sports streaming sources order home and away teams (e.g., listing the away team first instead of the home team), the method evaluates both direct and swapped arrangements. Let M(a,b) be a matching function that returns "True" if entity a and entity b are resolved as the same, direct (D) and swapped (S) checks are represented mathematically as:

$$D = M(LS0URCE, HSS0T) \wedge M(RS0URCE, ASS0T) \quad (2)$$

$$S = M(LS0URCE, ASS0T) \wedge M(RS0URCE, HSS0T) \quad (3)$$

where

$LS0URCE$: Left component (source record)

$HSS0T$: Home team or left component (database)

$RS0URCE$: Right component (source record)

$ASS0T$: Away team or right component (database)

2. Two-Way Token Contribution Requirement for Validation

A fundamental rule of the record matching process requires that a source record is validated and merged into the SSOT event only if it shares at least 2 matching tokens with the SSOT, such that each competing team contributes at least one token to the matched set. A single team cannot satisfy the match condition alone. If this condition is met (i.e., both teams contribute a matching token), the match is validated as a "High Confidence Match".

3. Exact matching and Levenshtein distance similarity

Token pairs from extracted records are evaluated against records across multiple SSOT partitions in the search for a true match for the same event. If there is no exact match between the tokens of the source record and SSOT, then Levenshtein distance similarity is evaluated.

Levenshtein distance exhibits high sensitivity to variance in short strings because every single edit operation (insertion, deletion, substitution) has the same weight associated with it (Christen 2012). For short strings, even a minor difference (like a typo, diacritics on character etc) has a high penalty per error, thereby disproportionately reducing similarity scores. In many languages, diacritics are essential components of the writing system. In light of these considerations, a similarity threshold of 0.75 (75%) was selected for Levenshtein distance for validating a match between two strings.

4. Null-Token Levenshtein Fallback

In cases where either or both competing teams have an empty token, the algorithm bypasses token comparison and falls back to calculating the Levenshtein distance similarity on the normalized raw strings. This case may arise when a team's entire name is stripped during normalization where all the generated tokens are discarded by the tokenization regex, leaving an empty token array for the team.

5. Validation

This final step in the entity resolution process is to determine if we have an overall match for the two records being compared (Shearer 2024). A source record and database event are validated as exactly the same if they satisfy the two-way token contribution requirement and return a logical truth for the disjunction:

$$V = D \vee S \quad (4)$$

V is the validation state of an overall match between a source record and a database event. D and S defined in equation (2) and (3) respectively.

Every extracted record is compared against records across different partitions of the SSOT to identify a true match corresponding to the same event. For each source record, the matching procedure (steps 1-5) is executed until a corresponding match is found or all the targeted database partitions have been evaluated. The process flow is illustrated in Figure 3.

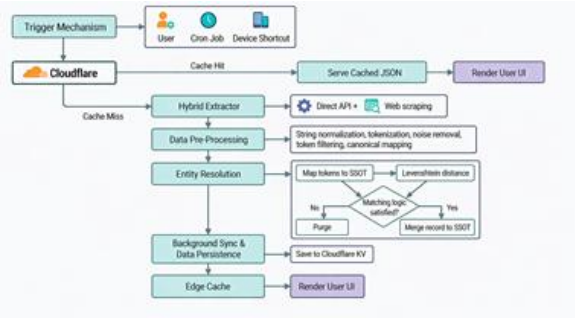

Figure 3. Architecture Process Flow

3.2.5 Operational Constraints

TheSportsDB enforces strict rate limits on their free tier APIs (30 requests per minute) and if the limit is breached, then users will need to wait for another minute for requests to start working again. In order to respect the API's restrictions without triggering "429 Too Many Requests" errors or IP blocks, a batch-processing queue is used to fetch match metadata.

3.3 Technologies Used

The system was implemented using HTML, CSS, and JavaScript to develop the user interface and core application

functionalities. Cloudflare Workers were employed to support serverless backend operations.

IV. RESULTS

The architectural design successfully ingested varying data formats (both structured API payloads and unstructured HTML formats) into a unified, normalized JSON structure. The predefined mapping ensured extreme variations in team names can be captured and matched with SSOT strings. By stripping generic identifiers (FC, Athletic, United, Club, Town, Olympique etc.) before entity resolution with multi-step matching logic, the risk of false negatives resulting from overlapping similar generic identifiers in competing team names are eliminated. The strict two-team token contribution rule ensures the possibility of false positives is eliminated where one team's name heavily overlapped with another fixture. Heterogeneous records from disparate multiple sources are successfully merged with the corresponding SSOT schedule representing the same event, and feeds directly into a single client-side user interface card. Additionally, implementing Cloudflare KV storage enabled highly efficient data delivery.

Figure 4. Browser developer console displaying the records extracted from different sources. Each event has attributes such as streaming URLs.

Figure 5. A user interface featuring event cards that present data unified from multiple disparate sources.

V. DISCUSSION

5.1 Significance of the Work

The digital sports ecosystem is highly fragmented, with users frequently forced to navigate multiple platforms to locate specific events. This work demonstrates an automated architecture for resolving sports data fragmentation and inconsistency, ensuring the end-user interacts with an organized, deduplicated interface with streaming links. Combining automated hybrid data extraction with entity resolution, data pre-

processing and forcing data into SSOT framework seamlessly translates chaotic, unstructured web data into a centralized, deduplicated index, ensuring the end-user interacts with an organized interface.

5.2 Challenges

Despite its effectiveness, the methodology presented here features the following challenges:

(i) Extraction Brittleness: While the direct API integrations are stable, scraping relies entirely on static HTML structures and specific Regex patterns of target websites. If a target website or broadcast provider updates its HTML structure, the corresponding scraper will fail silently until corrected.

(ii) The "Cold Start" Penalty: Scheduled cron jobs are not guaranteed to execute at the specified time and there can be delays. If the scheduled trigger (cron job or device shortcut) is delayed or fails to run, the primary impact falls on the first user of the day, who may encounter delayed initialization. This user bears the full latency of multiple processing cycles including data extraction, entity resolution, and KV write operations. As a result, the UI remains unpopulated until all batch cycles have finished executing.

(iii) Concurrent Related Fixtures: If different divisions of the same competing teams are scheduled to play on the same day, while the correct streaming links from the sources will be merged to the respective SSOT event, the current setup may also assign additional streaming links from the related concurrent fixtures to each other. For example, if (1) Manchester United vs Arsenal and (2) Manchester United Women vs Arsenal Women are scheduled to be played simultaneously, event (1) may receive streaming links intended for both event (1) and event (2), and vice versa. This scenario is a potential edge case and expected to be rare.

5.3 Directions for Future Research

The current framework provides a foundation for future enhancements, particularly through the integration of machine learning. Incorporating machine learning-driven Natural Language Processing (NLP) models for entity resolution would enable the system to intelligently identify semantically related entities, even in the absence of direct lexical similarity. Such an advancement would transform the system from a rule-based approach into a more adaptive engine.

VI. CONCLUSION

This work demonstrates that the synthesis of data pre-processing, and entity resolution, together with the Single Source of Truth (SSOT) model creates a robust framework for integrating heterogeneous sports data from multiple sources. By implementing a multi-step matching process and verification rule, the system effectively mitigates merging errors and record duplication. Furthermore, by leveraging edge caching and batching, this architecture offers a cost-effective method for transforming fragmented web data into cohesive, low-latency user experiences.

REFERENCES

1. Bilenko, M. & Mooney, R. J. (2003). Adaptive duplicate detection using learnable string similarity measures. Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. Washington, D.C., Association for Computing Machinery: 39–48.
2. Christen, P. (2012). The Data Matching Process. Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection. Berlin, Heidelberg, Springer Berlin Heidelberg: 23-35.
3. Hassan, H., Manan, M., Sehar, S., Shahzadi, A. & Fatima, M. (2025). "Automation in Web Data Extraction: Opportunities, Challenges, and Ethical Considerations." International Journal of Computing and Data Science 1(2): 37-46.
4. Jiang, Y., Lin, C., Meng, W., Yu, C., Cohen, A. M. & Smalheiser, N. R. (2014). "Rule-based deduplication of article records from bibliographic databases." Database 2014.
5. Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. Soviet physics doklady, Soviet Union.
6. López, E. & Ojanperä, S. (2025). Digital Data Handling: A Tutorial Approach. Human Development and the Data Revolution. Ojanperä, S., López, E. & Graham, M., Oxford University Press: 0.
7. Mamun, A.-A., Mi, T., Aseltine, R. & Rajasekaran, S. (2014). "Efficient sequential and parallel algorithms for record linkage." Journal of the American Medical Informatics Association 21(2): 252-262.

8. Manning, C. D., Raghavan, P. & Schütze, H. (2008). Introduction to Information Retrieval. Cambridge, Cambridge University Press.
9. Shearer, M. (2024). Hands-On Entity Resolution, " O'Reilly Media, Inc.".
10. Talburt, J. R. (2011). Chapter 1 - Principles of Entity Resolution. Entity Resolution and Information Quality. Boston, Morgan Kaufmann: 1-37.
11. Talburt, J. R. & Zhou, Y. (2015). Chapter 2 - Entity Identity Information and the CSRUD Life Cycle Model. Entity

12. Information Life Cycle for Big Data. Boston, Morgan Kaufmann: 17-29.
13. Tsuruoka, Y., McNaught, J., Tsujii, J. i., chi & Ananiadou, S. (2007). "Learning string similarity measures for gene/protein name dictionary look-up using logistic regression." Bioinformatics 23(20): 2768-2774.
14. Wickham, H., Çetinkaya-Rundel, M. & Grolemund, G. (2023). R for Data Science: Import, Tidy, Transform, Visualize, and Model Data, O'Reilly.
15. Zehtaban, L., Elazhary, O. & Roller, D. (2016). "A framework for similarity recognition of CAD models." Journal of Computational Design and Engineering 3(3): 274-285.