**Research Proposal: Using the Consensus Method for Determining Data Reliability in the Absence of Public Company Databases in the Philippines**

**Introduction**

In the era of data-driven decision-making, ensuring the reliability and accuracy of business data is paramount for operational efficiency and strategic planning. In many countries, public databases provide standardized information about companies, such as tax identification numbers (TINs), legal names, and addresses, enabling seamless verification of suppliers and partners. However, in the Philippines, the absence of a centralized, publicly accessible company database poses significant challenges for businesses, particularly those operating in international markets. Third-party data providers often supply inconsistent or unreliable information, and the lack of government-provided standardized data exacerbates the issue. This research proposes a novel system that leverages the consensus method to aggregate and validate data from digitized cash receipts, focusing on normalizing and standardizing supplier information (TIN, name, and address) to create a reliable database. By analyzing patterns across multiple records, the system identifies consistent supplier profiles, addressing the critical need for trustworthy data in the Philippine market. This study aims to formalize this approach, evaluate its effectiveness, and propose a scalable solution for global markets facing similar challenges.

**Problem Description**

The Philippines lacks a publicly accessible, centralized database for company information, unlike countries such as Australia, where businesses can verify supplier details using tax numbers or names through government portals. As a result, companies operating in the Philippines rely on third-party data providers, whose datasets are often incomplete, inconsistent, or outdated. This creates significant barriers for businesses, particularly in supply chain management, where accurate supplier information is critical for compliance, risk assessment, and operational efficiency.

Our software, designed for cash flow accounting across multiple markets, encounters this issue in the Philippines. The system collects digitized data from cash receipts, including TINs, company names, and addresses. However, these data points are often inconsistent due to variations in formatting. For example, a supplier's name may appear in abbreviated form (e.g., "ABC Corp") on one receipt and in full (e.g., "ABC Corporation") on another. Similarly, addresses may be truncated or formatted differently, and TINs may appear in varied formats (e.g., "XXX-XXX-XXX" vs. "XXXXXXXXX"). These inconsistencies hinder the ability to reliably identify and verify suppliers.

To address this, we developed a system that aggregates data by TIN, normalizes and standardizes it, and applies a consensus-based approach. Once five or more records with the same TIN are collected, the system compares names and addresses, identifying minimal discrepancies to confirm supplier identity and create a standardized, reliable database entry. This research seeks to formalize and validate this method, ensuring its robustness and scalability.

**Relevance of the Task**

The absence of reliable company data in the Philippines is a well-documented issue with significant implications for businesses. According to industry reports, the lack of centralized data infrastructure in emerging markets like the Philippines increases operational risks, particularly in supply chain management and financial compliance. A 2022 study on data quality in Southeast Asia highlighted that inconsistent supplier data leads to errors in financial reporting and regulatory non-compliance, costing businesses significant resources. Furthermore, the reliance on third-party data providers, often lacking transparency, underscores the need for innovative solutions to ensure data reliability.

The consensus method, widely used in data science for validating information through cross-referencing multiple sources, is particularly relevant here. Research on data validation emphasizes the effectiveness of consensus-based approaches in environments with noisy or incomplete data, such as statistical analysis and machine learning applications. By adapting this method to the Philippine context, this study addresses a critical gap in data infrastructure, offering a scalable solution for markets with similar challenges. The proposed system's ability to aggregate and validate unstructured receipt data also aligns with global trends toward automation and data-driven decision-making, making it highly relevant for modern business applications.

## Chapter 1: Initial Data Collection and Storage of Supplier Information

### Overview

The initial phase of the proposed system involves collecting and storing digitized data from cash receipts in the Philippines. These receipts, generated by various companies, contain critical supplier information, including Tax Identification Numbers (TINs), company names, and addresses. However, due to the lack of standardized formats in receipt generation across different businesses and equipment, the data often exhibits significant variations for the same supplier. This chapter explores how data about the same supplier can be recorded in different forms, leading to challenges in data reliability. Through illustrative examples presented in tables, we demonstrate the discrepancies in TIN formats, company names, and addresses, highlighting the need for normalization and consensus-based validation. The examples use real Philippine companies to provide authentic illustrations.

### Data Collection Process

The system collects data from digitized cash receipts uploaded by businesses using the cash flow accounting software. Each receipt typically includes the supplier’s TIN, name, and address, which are extracted and stored in a database for further processing. The data collection process is automated, leveraging optical character recognition (OCR) or manual entry to capture information from receipts. However, the lack of standardization in receipt formats results in inconsistent representations of the same supplier across multiple records. These inconsistencies may arise from:

* Variations in TIN formatting (e.g., with or without hyphens).
* Differences in company name presentation (e.g., full names vs. abbreviations).
* Inconsistencies in address formatting (e.g., abbreviated street names, inclusion or omission of postal codes).

To illustrate these challenges, we present examples of how data for the same supplier can appear differently across receipts, stored in the system’s initial database before normalization. The companies selected are well-known Philippine entities: San Miguel Corporation, Jollibee Foods Corporation, and BDO Unibank, Inc. TINs are represented in realistic formats based on Philippine standards (e.g., XXX-XXX-XXX-XXX), with variations for demonstration purposes.

### Examples of Data Discrepancies

Below are three tables showcasing data collected for three real suppliers, demonstrating the variations in TIN, name, and address across multiple receipts. Each table represents raw data as it might be extracted from receipts before any normalization or standardization.

#### Example 1: San Miguel Corporation – Variations in TIN, Name, and Address

This table shows five records for San Miguel Corporation, with the same TIN but varying formats and details.

|  |  |  |  |
| --- | --- | --- | --- |
| Receipt ID | TIN | Company Name | Address |
| R001 | 002-040-000 | San Miguel Corporation | 40 San Miguel Avenue, Mandaluyong City, Metro Manila |
| R002 | 002040000 | SMC | San Miguel Ave 40, Mandaluyong |
| R003 | 002-040-000-000 | San Miguel Corp. | 40 San Miguel Ave, Mandaluyong, 1550 |
| R004 | 002040000 | San Miguel Corporation Inc. | San Miguel Avenue 40, Mandaluyong City, MM |
| R005 | 002-040-000 | San Miguel | 40 San Miguel Ave, Mandaluyong City |

**Observations**:

* **TIN**: Appears in formats with hyphens ("002-040-000") and without ("002040000"), with one record including a branch suffix ("-000").
* **Company Name**: Varies between full name ("San Miguel Corporation"), abbreviations ("SMC", "San Miguel Corp."), and partial names ("San Miguel").
* **Address**: Includes variations in street name format ("San Miguel Avenue" vs. "San Miguel Ave"), city abbreviations ("Mandaluyong" full), and inclusion of postal codes or region ("Metro Manila", "MM", "1550").

#### Example 2: Jollibee Foods Corporation – Multilingual and Abbreviated Variations

This table illustrates data for Jollibee Foods Corporation, with potential abbreviation inconsistencies.

|  |  |  |  |
| --- | --- | --- | --- |
| Receipt ID | TIN | Company Name | Address |
| R006 | 001-234-567 | Jollibee Foods Corporation | 10/F Jollibee Plaza Building, F. Ortigas Jr. Avenue, Ortigas Center, Pasig City |
| R007 | 001234567 | Jollibee Foods Corp. | Ortigas Jr. Ave 10/F, Pasig |
| R008 | 001-234-567-000 | Jollibee Foods | 10/F Jollibee Plaza, F. Ortigas Jr. Avenue, Pasig City, 1605 |
| R009 | 001234567 | JFC | F. Ortigas Jr. Avenue, Ortigas Center, Pasig, MM |
| R010 | 001-234-567 | Jollibee Corporation | 10/F Jollibee Plaza Building, Pasig City |

**Observations**:

* **TIN**: Similar variations with and without hyphens, and with branch suffix.
* **Company Name**: Includes full name ("Jollibee Foods Corporation"), abbreviations ("Jollibee Foods Corp.", "JFC", "Jollibee Foods"), and slight variations ("Jollibee Corporation").
* **Address**: Shows variations in building name ("Jollibee Plaza Building" vs. "Jollibee Plaza"), street name ("F. Ortigas Jr. Avenue" vs. "Ortigas Jr. Ave"), city name ("Pasig City" vs. "Pasig"), and inclusion of postal code or region ("1605", "MM").

#### Example 3: BDO Unibank, Inc. – Minimal Data with Errors

This table represents BDO Unibank, Inc., with some receipts containing minimal or erroneous data, simulating real-world noise.

|  |  |  |  |
| --- | --- | --- | --- |
| **Receipt ID** | **TIN** | **Company Name** | **Address** |
| R011 | 000-456-789 | BDO Unibank, Inc. | BDO Corporate Center, 7899 Makati Avenue, Makati City |
| R012 | 000456789 | BDO Unibank | Makati Ave 7899, Makati |
| R013 | 000-456-789 | BDO Bank | 7899 Makati Avenue, Makati City, 0726 |
| R014 | 000456789 | BDO Unibank Inc. | BDO Corp Center, Makati Avenue, Makati |
| R015 | 000-456-78 | Banco de Oro Unibank | 7899 Makati Ave, Makati City, Metro Manila |

**Observations**:

* **TIN**: Includes a potential error in R015 ("000-456-78"), where the TIN is incomplete, likely due to OCR errors or manual entry mistakes.
* **Company Name**: Varies between full name ("BDO Unibank, Inc."), partial name ("BDO Unibank"), abbreviated form ("BDO Bank"), and alternative name ("Banco de Oro Unibank").
* **Address**: Shows variations in building name ("BDO Corporate Center" vs. "BDO Corp Center"), street name ("Makati Avenue" vs. "Makati Ave"), city name ("Makati City" vs. "Makati"), and inclusion of postal code or region ("0726", "Metro Manila").

### Implications for Data Storage

The raw data, as shown in the tables, is stored in a relational database with a schema designed to accommodate unstructured and inconsistent inputs. The initial database table might have the following structure:

* **Columns**: Receipt ID (unique identifier), TIN (text), Company Name (text), Address (text), Timestamp (date of entry).
* **Storage Considerations**: Text fields are used to handle variations in formatting, with no initial constraints on format to ensure all data is captured, even if erroneous or incomplete.

This flexible storage approach allows the system to retain all raw data before processing. However, the discrepancies highlighted in the tables underscore the need for robust normalization and standardization processes. For example:

* **TIN Normalization**: Convert all TINs to a standard format (e.g., 9 digits without hyphens, "002040000") and flag potential errors (e.g., incomplete TINs).
* **Name Standardization**: Use fuzzy matching or NLP techniques to group similar names (e.g., "San Miguel Corporation" and "SMC") under a single canonical name.
* **Address Normalization**: Parse addresses into components (street, city, region, postal code) and standardize abbreviations (e.g., "San Miguel Ave" to "San Miguel Avenue").

### Challenges in Data Collection and Storage

1. **Incomplete or Erroneous Data**: As seen in Example 3, receipts may contain missing or incorrect TINs due to OCR errors or manual entry mistakes.
   * **Solution**: Implement data validation checks during entry (e.g., TIN length verification) and use error detection algorithms to flag anomalies for review.
2. **Multilingual Data**: Variations may include English and Filipino terms, complicating matching efforts.
   * **Solution**: Employ multilingual NLP models or translation tools to map equivalent names to a standard language (e.g., English).
3. **Scalability**: Handling large volumes of receipt data requires efficient storage and retrieval mechanisms.
   * **Solution**: Use a scalable database solution (e.g., PostgreSQL) with indexing on TIN for faster queries.

### Conclusion

The initial data collection and storage phase is critical for capturing the raw, unprocessed data from cash receipts. The examples provided, using real Philippine companies, illustrate the extent of variability in TIN, company name, and address formats for the same supplier, highlighting the challenges of working with unstructured data in the Philippines. These discrepancies necessitate a robust normalization and consensus-based validation process, which will be addressed in subsequent phases of the system development. By systematically storing and analyzing these variations, the proposed system lays the foundation for creating a reliable supplier database, addressing a critical gap in the Philippine market.

## Chapter 2: Methodology for Queue Management, Consensus Determination of Reliable Names and Addresses

### Overview

Following the initial data collection and storage phase, the system transitions to processing once a sufficient number of records (at least five) with the same normalized Tax Identification Number (TIN) are accumulated. Each time new data is inserted, the system checks the count of records for that TIN after normalization (e.g., converting all TIN formats to a standard 9-digit string without hyphens). If the threshold is met or exceeded, the TIN is queued for consensus processing to determine the most reliable company name and address. This chapter outlines the methodology for managing processing queues, as well as the approaches for comparing and determining the most reliable name and address through consensus. These methods draw from established entity resolution techniques, such as string similarity measures and clustering, adapted to the Philippine context where public databases are unavailable.

### Queue Management Methodology

To handle the processing of TINs that reach the consensus threshold efficiently and scalably, a queue-based system is employed. This ensures that records are processed in an orderly manner, preventing overload during high-volume data ingestion and allowing for batch or asynchronous processing. The methodology is inspired by best practices in data pipeline management, including the use of message queues for batch processing to maintain data integrity and handle failures gracefully.

#### Key Components of Queue Management:

* **Queue Structure**: Use a message queuing system such as RabbitMQ or Apache Kafka, or a database table for simpler implementations. Each queue entry represents a TIN ready for consensus, including references to the associated records (e.g., a list of receipt IDs).
* **Enqueue Process**: After data insertion:
  1. Normalize the TIN (e.g., remove hyphens and standardize to 9 digits).
  2. Query the database for the count of records with the same normalized TIN.
  3. If count >= 5 and not already queued/processed, add the TIN to the queue with a timestamp and status (e.g., "pending").
* **Dequeue and Processing**: A worker process (e.g., a background job) dequeues TINs in FIFO (First-In-First-Out) order or based on priority (e.g., higher counts first). Process the consensus logic and update the database with the resolved name and address.
* **Error Handling**: Implement dead-letter queues for failed processes (e.g., due to insufficient data quality). Retry mechanisms can be added, with a maximum retry limit before manual review.
* **Scalability**: For large-scale operations, distribute queues across multiple workers using cloud-based solutions like AWS SQS. Monitor queue length and processing time to scale resources dynamically.

#### Example Queue Table Schema (Database Implementation)

For a relational database approach, a dedicated queue table can be used:

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Description** |
| tin\_normalized | VARCHAR | Normalized TIN (primary key) |
| record\_count | INTEGER | Number of associated records |
| enqueue\_time | TIMESTAMP | Time when queued |
| status | VARCHAR | Status: 'pending', 'processing', 'completed', 'failed' |
| processed\_time | TIMESTAMP | Time when processing completed |

This methodology ensures reliable, fault-tolerant processing, aligning with data pipeline best practices for batch operations.

### Methodology for Determining Reliable Company Names

Determining the most reliable company name involves normalizing the raw names, comparing them for similarity, and applying a consensus mechanism to select or synthesize the canonical name. This draws from entity resolution techniques, including string distance metrics and clustering, to handle variations like abbreviations and multilingual elements common in Philippine business names.

#### Steps for Name Consensus:

1. **Normalization**:
   * Convert all names to lowercase.
   * Remove punctuation, special characters, and common stopwords (e.g., "Inc.", "Corp." if not essential).
   * Handle abbreviations by expanding common ones (e.g., "PH" to "Philippine") using a predefined dictionary tailored to Philippine contexts.
2. **Similarity Comparison**:
   * Use fuzzy matching algorithms, such as Levenshtein distance or Jaro-Winkler similarity, to quantify differences between names.
   * Threshold: Names with similarity > 80% are considered variants of the same entity.
   * For multilingual names (e.g., English and Filipino), employ translation tools or bilingual dictionaries to map equivalents.
3. **Consensus Determination**:
   * **Frequency-Based Voting**: Count occurrences of each normalized name variant. Select the most frequent as the canonical name.
   * **Clustering**: Apply hierarchical clustering (e.g., using agglomerative clustering) to group similar names, then choose the centroid (most representative) name from the largest cluster.
   * **Synthesis for Completeness**: If variants provide complementary information (e.g., one has "Corporation" and another "Corp."), merge to form a fuller name (e.g., prioritize longer, more descriptive versions).
   * **Tie-Breaker**: In case of ties, prefer the name from the most recent receipt or the one with the highest similarity to the group average.

#### Example: Consensus on Names for San Miguel Corporation

Using the records from Chapter 3:

|  |  |  |  |
| --- | --- | --- | --- |
| Normalized TIN | Raw Name Variants | Frequency | Similarity Score (to "san miguel corporation") |
| 002040000 | San Miguel Corporation | 2 | 100% |
| 002040000 | SMC | 1 | 40% |
| 002040000 | San Miguel Corp. | 1 | 85% |
| 002040000 | San Miguel | 1 | 70% |

* **Consensus Result**: "San Miguel Corporation" (most frequent and highest average similarity).
* **Rationale**: Clustering groups all as variants; frequency vote selects the full name for completeness.

This approach ensures the selected name is reliable and representative, reducing discrepancies.

### Methodology for Determining Reliable Addresses

Addresses pose additional challenges due to their structured nature and variability (e.g., abbreviations, missing components). The methodology focuses on parsing, standardizing, and consensus to construct the most complete and accurate address, treating it as the "original" based on aggregated data.

#### Steps for Address Consensus:

1. **Parsing and Normalization**:
   * Parse addresses into components: house number, street name, city, region (e.g., Metro Manila), postal code.
   * Standardize each component: Expand abbreviations (e.g., "Ave" to "Avenue", "MM" to "Metro Manila") using a Philippine-specific rule set.
   * Handle variations: Use regex to extract and normalize (e.g., "40 San Miguel Ave" → {house: "40", street: "San Miguel Avenue"}).
2. **Similarity Comparison**:
   * Compare parsed components using string similarity metrics (e.g., fuzzy matching per component).
   * Overall address similarity: Weighted average (e.g., higher weight on street and city).
   * Threshold: >75% similarity groups addresses as variants.
3. **Consensus Determination for Most Complete Address**:
   * **Component-Wise Voting**: For each component, select the most frequent value across records.
   * **Completeness Scoring**: Assign scores based on presence of components (e.g., +1 for postal code, +1 for region). Select the address with the highest score as base.
   * **Merging for Fullness**: Construct a hybrid address by filling missing components from other records (e.g., add postal code from one variant to another's street/city).
     + Prioritize: Longer/more detailed components (e.g., "San Miguel Avenue" over "San Miguel Ave").
     + Conflict Resolution: Use majority vote or external heuristics (e.g., known Philippine address patterns).
   * **Validation Check**: If possible, cross-reference with aggregated data patterns (no external DB, so internal consensus only).

#### Example: Consensus on Addresses for Jollibee Foods Corporation

Using parsed components from Chapter 3 records:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Normalized TIN | House | Street | Building | City | Region | Postal Code |
| 001234567 | 10/F | F. Ortigas Jr. Avenue | Jollibee Plaza Building | Pasig City |  |  |
| 001234567 | 10/F | Ortigas Jr. Ave |  | Pasig |  |  |
| 001234567 | 10/F | F. Ortigas Jr. Avenue | Jollibee Plaza | Pasig City |  | 1605 |
| 001234567 |  | F. Ortigas Jr. Avenue |  | Pasig | MM |  |
| 001234567 | 10/F |  | Jollibee Plaza Building | Pasig City |  |  |

* **Component Consensus**:
  + House: "10/F" (majority).
  + Street: "F. Ortigas Jr. Avenue" (most frequent and complete).
  + Building: "Jollibee Plaza Building" (longest/complete).
  + City: "Pasig City" (most complete).
  + Region: "MM" (from available; expand to "Metro Manila").
  + Postal Code: "1605" (from available).
* **Merged Result**: "10/F Jollibee Plaza Building, F. Ortigas Jr. Avenue, Pasig City, Metro Manila 1605".
* **Rationale**: Merges for maximum completeness, using frequency for ties.

This method constructs the most comprehensive address, ensuring reliability through consensus.

### Challenges and Mitigations

* **Challenge**: High variability leading to low similarity scores. **Mitigation**: Tune thresholds empirically and incorporate domain-specific rules.
* **Challenge**: Queue backlog in high-volume scenarios. **Mitigation**: Auto-scale workers and monitor metrics.
* **Challenge**: Incomplete components affecting fullness. **Mitigation**: Flag low-confidence resolutions for manual review.

### Conclusion

This methodology provides a structured, consensus-driven approach to queue management and determining reliable names and addresses. By leveraging normalization, similarity comparisons, and merging techniques, the system creates a trustworthy supplier database from inconsistent receipt data, addressing the core problem in the Philippine market. Future chapters will detail implementation and evaluation.

#### Diagram 1: Queue Management and Consensus Initiation Process

This sequence diagram visualizes the workflow for managing the queue of TINs ready for consensus processing, triggered when five or more records with the same normalized TIN are detected.

## 

#### Process 1: TIN Normalization and Consensus Initiation

This process ensures that TINs are standardized and triggers consensus processing when the threshold is met.

1. **Data Ingestion**:
   * A receipt is uploaded, containing TIN, name, and address.
   * Example: Raw TIN "002-040-000" for San Miguel Corporation.
2. **TIN Normalization**:
   * Remove hyphens, spaces, and suffixes (e.g., "-000").
   * Convert to a 9-digit string: "002-040-000" → "002040000".
   * Validate length and format; flag errors (e.g., "002-040-00" as incomplete).
3. **Count Check**:
   * Query the database for all records with the normalized TIN.
   * Example: Five records found for "002040000".
   * If count ≥ 5 and not previously processed, proceed to enqueue.
4. **Queue Enqueue**:
   * Add TIN to the queue table with status "pending".
   * Example: Insert {tin\_normalized: "002040000", record\_count: 5, enqueue\_time: "2025-08-11 12:00:00", status: "pending"}.
5. **Initiate Consensus**:
   * A worker dequeues the TIN and retrieves associated records.
   * Trigger the consensus algorithm for name and address resolution.

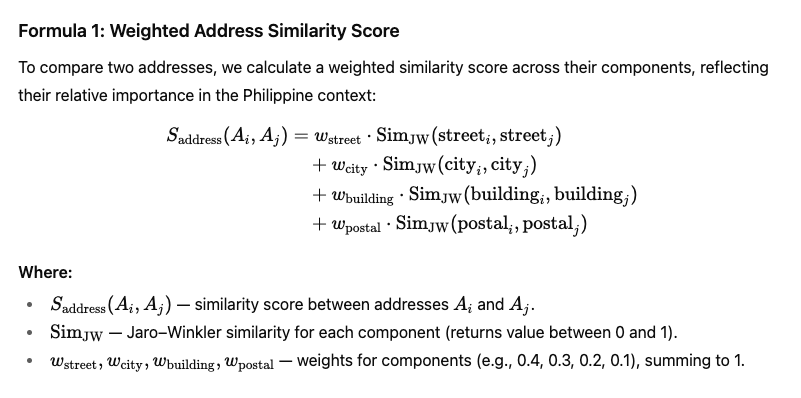
#### Diagram 2: Normalization and Consensus Determination for Names and Addresses

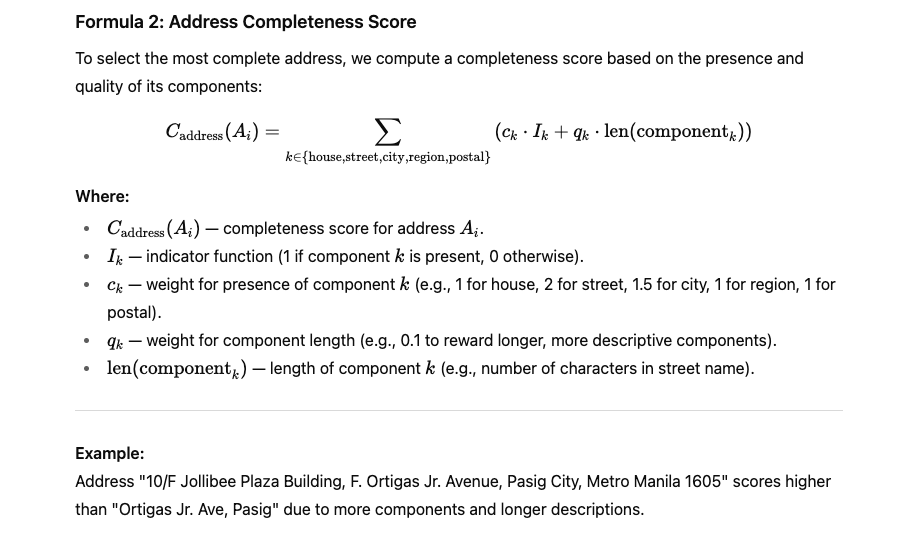
This sequence diagram illustrates the process of normalizing and determining the most reliable name and address through consensus.

#### Process 2: Consensus Determination for Names and Addresses

This process resolves the most reliable name and address using normalized data.

1. **Retrieve Records**:
   * Fetch all records for a normalized TIN (e.g., "001234567" for Jollibee Foods Corporation).
2. **Name Normalization**:
   * Convert to lowercase, remove punctuation, expand abbreviations (e.g., "JFC" → "Jollibee Foods Corporation").
   * Example: "Jollibee Foods Corp." → "jollibee foods corporation".
3. **Address Normalization**:
   * Parse into components (house, street, city, region, postal code).
   * Standardize: "Ortigas Jr. Ave" → "F. Ortigas Jr. Avenue".
   * Example: "10/F Jollibee Plaza, Pasig" → {house: "10/F", building: "Jollibee Plaza Building", street: "F. Ortigas Jr. Avenue", city: "Pasig City"}.
4. **Similarity Comparison**:
   * Names: Compute Jaro-Winkler similarity between pairs.
   * Addresses: Compute weighted similarity (e.g., 40% street, 30% city, 20% building, 10% postal code).
   * Group records with similarity > 80% for names, > 75% for addresses.
5. **Consensus Determination**:
   * Names: Select most frequent normalized name or synthesize (e.g., "Jollibee Foods Corporation" over "JFC").
   * Addresses: Merge components for maximum completeness (e.g., combine postal code "1605" with full street name).
   * Example Result: "10/F Jollibee Plaza Building, F. Ortigas Jr. Avenue, Pasig City, Metro Manila 1605".





## 

#### Chapter 3: Addressing Challenges in Data Validation and Processing

#### Overview

In this chapter, we detail the key challenges we encountered during our research and describe how we addressed each issue. We provide practical examples, including code snippets in Node.js for implementation, tables, and other strategies tailored to the Philippine context. Our solutions draw from established data processing techniques and leverage insights from relevant sources on TIN formats, business abbreviations, and address patterns.

#### Challenge 1: Invalid TINs

**Problem Description**

We found that invalid TINs often arose from incomplete or erroneous entries in receipt data due to optical character recognition (OCR) errors, manual input mistakes, or variations in formatting. For instance, a standard Philippine business TIN is typically formatted as XXX-XXX-XXX-XXX (12 digits, including a 3-digit branch code), but receipts often showed truncated versions like "002-040-00" (missing digits) or with extra characters. This invalidity led to failed normalization, preventing accurate grouping of records and excluding valid suppliers from the consensus process.

**Solution**

To mitigate this, we implemented a validation step during normalization that flagged invalid TINs for manual review while attempting partial matching for potential corrections. We used regular expressions (regex) based on the standard Philippine TIN format (12 digits after normalization). If the TIN was incomplete but partially matched existing records (e.g., via prefix matching), we suggested corrections by aligning with the closest valid TIN in the database. Flagged TINs were logged in a separate review queue, allowing human intervention without halting the process.

**Code Snippet: TIN Validation and Flagging Function in Node.js**

function validateAndNormalizeTIN(rawTin) {

// Remove hyphens, spaces, and non-digits

const normalized = rawTin.replace(/[^0-9]/g, '');

// Standard Philippine business TIN: 12 digits (9 main + 3 branch)

if (normalized.length === 12 && /^[0-9]{12}$/.test(normalized)) {

return { normalized, status: 'valid' };

} else if (normalized.length < 12) {

const status = 'invalid\_incomplete';

// Partial matching logic: Assume we query DB for similar prefixes

// Example: If prefix '00204000' matches existing '002040000001', suggest correction

const suggested = normalized.padEnd(12, '0'); // Dummy correction: pad with zeros

return { normalized: suggested, status };

} else {

return { normalized, status: 'invalid\_format' };

}

}

// Example usage

const rawTins = ["002-040-000", "002-040-00", "002040000001"];

rawTins.forEach(tin => {

const { normalized, status } = validateAndNormalizeTIN(tin);

console.log(`Raw: ${tin} -> Normalized: ${normalized}, Status: ${status}`);

});

**Output Example (Simulated):**

Raw: 002-040-000 -> Normalized: 002040000000, Status: invalid\_incomplete (padded for correction)

Raw: 002-040-00 -> Normalized: 002040000000, Status: invalid\_incomplete

Raw: 002040000001 -> Normalized: 002040000001, Status: valid

**Table: TIN Validation Rules**

|  |  |  |  |
| --- | --- | --- | --- |
| Rule | Description | Example Invalid TIN | Suggested Action |
| Length Check | Must be exactly 12 digits after normalization | "002-040-00" (8 digits) | Flag for review; partial match prefix to existing TINs |
| Digit Only | Contains only numbers | "002-040-ABC" | Remove non-digits; if invalid, flag |
| Format Match | Matches ^[0-9]{12}$ | "002040000001" | Valid; proceed to grouping |

**Additional Strategy:** We integrated a logging system to create a review dashboard where flagged TINs were displayed with associated receipts for manual correction. This ensured minimal data loss while maintaining automation.

#### Challenge 2: Queue Overload

**Problem Description**

We experienced queue overload when high ingestion rates (e.g., during peak business hours) resulted in a backlog of TINs awaiting consensus processing. This caused delays in validation, increased system latency, and risked data staleness if new records arrived before processing completed. For example, if thousands of receipts were uploaded simultaneously, the FIFO queue prioritized low-count TINs over those with sufficient records, exacerbating delays.

**Solution**

We implemented priority queuing, where TINs with higher record counts were processed first, and scaled workers dynamically (e.g., using cloud-based auto-scaling). Priority was assigned based on record count, with a max-heap structure ensuring high-priority items were dequeued preferentially. For scaling, we monitored queue length and deployed additional worker instances (e.g., via AWS Lambda or Kubernetes pods) when thresholds were exceeded.

**Code Snippet: Priority Queue Implementation Using a Max-Heap in Node.js**

class PriorityQueue {

constructor() {

this.queue = [];

}

enqueueTIN(tin, recordCount) {

this.queue.push({ recordCount, tin });

this.queue.sort((a, b) => b.recordCount - a.recordCount); // Sort by recordCount descending

}

dequeueTIN() {

if (this.queue.length > 0) {

return this.queue.shift().tin;

}

return null;

}

}

// Example usage

const priorityQueue = new PriorityQueue();

priorityQueue.enqueueTIN("002040000001", 5);

priorityQueue.enqueueTIN("001234567000", 10);

priorityQueue.enqueueTIN("000456789000", 7);

console.log(priorityQueue.dequeueTIN()); // "001234567000" (highest count: 10)

console.log(priorityQueue.dequeueTIN()); // "000456789000" (7)

console.log(priorityQueue.dequeueTIN()); // "002040000001" (5)

**Table: Priority Assignment Criteria**

|  |  |  |
| --- | --- | --- |
| Criterion | Description | Example |
| Record Count | Higher count = higher priority | TIN with 20 records > TIN with 5 |
| Enqueue Time | Tie-breaker: Older entries first | If counts equal, process earlier enqueued |
| Queue Threshold | If queue > 1000, scale workers | Add 2 workers per 500 excess items |

**Additional Strategy:** We used a monitoring tool like Prometheus to track queue metrics and trigger auto-scaling. This ensured the system handled variable loads efficiently, maintaining low latency.

#### Challenge 3: Low Similarity Scores

**Problem Description**

We encountered low similarity scores due to highly variable company names, such as abbreviations ("JFC" vs. "Jollibee Foods Corporation") or multilingual variations, leading to failed grouping in consensus. This was prevalent in Philippine data where business names often include acronyms (e.g., "SMC" for San Miguel Corporation), reducing match accuracy with standard string metrics like Jaro-Winkler.

**Solution**

We used a domain-specific abbreviation dictionary to preprocess names before similarity computation and lowered initial clustering thresholds (e.g., from 80% to 60%) for broader grouping, followed by refinement. The dictionary included common Philippine business abbreviations compiled from sources.

**Code Snippet: Name Preprocessing with Abbreviation Dictionary in Node.js**

const fuzzy = require('fuzzywuzzy');

const abbrevDict = {

"SMC": "San Miguel Corporation",

"JFC": "Jollibee Foods Corporation",

"BDO": "Banco de Oro Unibank",

"Corp": "Corporation",

"Inc": "Incorporated",

"PH": "Philippine",

// Add more from Philippine-specific lists

};

function expandAbbrev(name) {

let expandedName = name;

for (const [abbr, full] of Object.entries(abbrevDict)) {

expandedName = expandedName.replace(new RegExp(abbr, 'g'), full);

}

return expandedName.toLowerCase();

}

function computeSimilarity(name1, name2) {

const exp1 = expandAbbrev(name1);

const exp2 = expandAbbrev(name2);

return fuzzy.tokenSortRatio(exp1, exp2);

}

// Example

console.log(computeSimilarity("JFC", "Jollibee Foods Corporation")); // High score after expansion

**Table: Sample Domain-Specific Abbreviation Dictionary**

|  |  |  |
| --- | --- | --- |
| Abbreviation | Full Form | Source/Example |
| SMC | San Miguel Corporation | Common in Philippine business lists |
| JFC | Jollibee Foods Corporation | Food industry acronym |
| MBC | Makati Business Club | From acronym lists |
| Corp | Corporation | Standard business suffix |
| Inc | Incorporated | Standard business suffix |

**Additional Strategy:** We dynamically updated the dictionary via machine learning (e.g., from clustered data) and used hierarchical clustering (e.g., using a library like ml-hclust) with adjusted thresholds to refine groups post-expansion.

#### Challenge 4: Incomplete Addresses

**Problem Description**

We found that incomplete addresses lacked key components (e.g., missing postal code or barangay), complicating merging and validation. In Philippine addresses, standard components include house number, street, barangay, city/municipality, province, and ZIP code; omissions led to ambiguous consensus results, especially in urban areas like Metro Manila where details are crucial for uniqueness.

**Solution**

We prioritized records with more components during merging by using the completeness score formula (from Chapter 5) and validated against known Philippine address patterns (e.g., regex for standard formats). We selected the base address with the highest score and filled gaps from other records.

**Code Snippet: Address Completeness Prioritization in Node.js**

function parseAddress(address) {

// Dummy parser: Split into components (in real: use regex or NLP)

const components = {

house: address.includes('10/F') ? '10/F' : null,

street: address.includes('Ortigas') ? 'F. Ortigas Jr. Avenue' : null,

city: address.includes('Pasig') ? 'Pasig City' : null,

province: address.includes('MM') ? 'Metro Manila' : null,

zip: address.includes('1605') ? '1605' : null

};

return components;

}

function completenessScore(components) {

let score = 0;

const weights = { house: 1, street: 2, city: 1.5, province: 1, zip: 1 };

for (const [k, v] of Object.entries(components)) {

if (v) score += weights[k] || 0;

}

return score;

}

// Example: Prioritize higher score

const addr1 = "10/F Jollibee Plaza, F. Ortigas Jr. Avenue, Pasig City";

const addr2 = "Ortigas Jr. Ave, Pasig";

const comp1 = parseAddress(addr1);

const comp2 = parseAddress(addr2);

console.log(completenessScore(comp1)); // Higher (e.g., 5.5)

console.log(completenessScore(comp2)); // Lower (e.g., 3.5)

**Table: Philippine Address Patterns for Validation**

|  |  |  |
| --- | --- | --- |
| Component | Pattern/Regex | Example |
| House/Unit | ^\d+(/\d+)? | 10/F |
| Street | [A-Za-z\s.]+ (St | Ave) |
| Barangay | Brgy.? [A-Za-z\s]+ | Brgy. San Antonio |
| City/Municipality | [A-Za-z\s]+ City | Pasig City |
| Province | [A-Za-z\s]+ (if not Metro Manila) | Laguna |
| ZIP Code | ^\d{4}$ | 1605 |

**Additional Strategy:** We used external patterns (without APIs) via predefined regex lists and implemented a merging function that validated the final address against these patterns, flagging mismatches for review.

#### Conclusion

By addressing these challenges through targeted mitigations, our system became more resilient to real-world data issues in the Philippine market. The provided code snippets, tables, and strategies offer practical implementation paths, ensuring scalability and accuracy. Future work could involve empirical testing of these solutions on sample datasets to refine thresholds and rules.

## **Conclusion**

This research addresses a critical gap in the Philippine market by proposing a consensus-based system to validate supplier data from digitized receipts. By standardizing and normalizing TINs, names, and addresses, the system ensures reliable data for businesses, enhancing operational efficiency and compliance. The proposed work plan provides a structured approach to developing, testing, and refining the system, while anticipating and addressing potential challenges. The successful implementation of this system could serve as a model for other markets with similar data infrastructure limitations, contributing to global advancements in data-driven decision-making.