ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/25aug118

Harmonizing Multilingual Product Data Using Machine Learning: A Case Study of the Rwanda Revenue Authority

Raymond Kamana[1]

[1]Adventist University of Central Africa, Kigali, Rwanda

Publication Date: 2025/08/08

Abstract: This study focuses on solving the problem of inconsistent and multilingual product names in the Rwanda Revenue Authority's (RRA) Electronic Billing Machine (EBM) system. Because product names are entered manually, many spelling differences and translations make it hard to track and analyze tax data. To fix this, the study uses Natural Language Processing (NLP) and Machine Learning (ML) to clean and group similar product names. A total of 4.1 million records from 2020 to 2022 were translated into English and processed. Sentence meaning was captured using MiniLM embeddings, then simplified using UMAP, and finally grouped using HDBSCAN. The cleaned and grouped product names make it easier to detect possible fraud, spot underpricing, and improve the accuracy of tax reporting. This method helps RRA improve data quality and tax compliance.

Keywords: *Multilingual Harmonization, Natural Language Processing (NLP), Machine Learning (ML), Product Name Clustering, Sentence Embedding, MiniLM, HDBSCAN, KMeans, UMAP, Language Detection, Tax Data Quality, Rwanda Revenue Authority (RRA), Electronic Billing Machine (EBM), Anomaly Detection, Cross-lingual Processing.*

How to Cite: Raymond Kamana (2025) Harmonizing Multilingual Product Data Using Machine Learning: A Case Study of the Rwanda Revenue Authority. *International Journal of Innovative Science and Research Technology*, 10(8), 79-84. https://doi.org/10.38124/ijisrt/25aug118

I. INTRODUCTION

Since 2013, the Rwanda Revenue Authority (RRA) has progressively shifted from manual invoicing systems to digital tax administration through the introduction of Electronic Billing Machines (EBMs). The first version, EBM v1, served as a transitional tool to move away from paper-based records. It captured invoices as images and extracted limited information using techniques like regular expressions.

The real digital transformation came with EBM v2.0 and v2.1, which introduced structured, item-level data marking a full transition to digital invoicing. These versions aimed to improve accuracy, transparency, and traceability across the taxpayer base. However, one persistent challenge remained: the manual entry of product names in multiple languages, leading to inconsistent and unstandardized item descriptions. For instance, "sugar" could be recorded as *sucre*, *isukari*, or *sukari*, making data analysis and fraud detection more difficult.

➢ *EBM v2.1 was Adapted to Various Taxpayer Needs:*

- **Windows EBM**: For large businesses with high sales volumes
- **EBM Mobile**: For informal or small-scale non-VAT-registered businesses
- **Online EBM**: For low-volume service providers (e.g., legal, rental services)
- **VSDC**: Integrated into companies' internal systems

While these platforms expanded digital tax coverage, they also introduced data quality issues due to language and input variation.

II. LITERATURE REVIEW

Multilingual inconsistencies in tax records arise due to linguistic diversity and manual data entry errors. Resolving these challenges requires sophisticated text processing approaches (Christen, 2012).

Modern NLP methods such as MiniLM and Sentence-BERT can capture cross-language semantic similarities by embedding product descriptions into vector space representations (Reimers & Gurevych, 2019).

Clustering algorithms like K-Means and HDBSCAN allow for grouping similar product names even when they are written in different languages or formats (McInnes et al., 2018).

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/25aug118

Effective tax data analytics depends on key data quality attributes: accuracy, completeness, consistency, timeliness, validity, and uniqueness (Batini et al., 2009).

Cross-lingual learning enables semantic alignment across languages, an essential capability for standardizing multilingual product descriptions in tax compliance systems (OECD, 2017, 2022).

III. METHODOLOGY

- The study adopts a **Design Science Research** approach, focused on building and evaluating an artifact (the harmonization pipeline). The pipeline follows a **modular architecture**, integrating the following stages:

- Text Preprocessing
- Language Detection and Translation
- Semantic Embedding
- Dimensionality Reduction
- Clustering and Interpretation
- Development Stack

- *The System was Implemented using **Python** with the Following Libraries:*

- Lang detects, Google trans (language detection and translation)
- sentence-transformers (embedding generation)
- scikit-learn, UMAP, HDBSCAN (clustering and dimensionality reduction)

- *Dataset Overview*

- **Total Records**: 4,124,005 (FY 2020–2022)
- **Languages**: Kinyarwanda, French, Swahili, English
- **Features**: Item Name, Transaction Type, Unit Price, Quantity, Value, Tax Category
- **Task**: Harmonize product names across datasets (Import, Purchase, Sales)
- **Training/Test Split**: 80/20 on embedded harmonized data

IV. SYSTEM IMPLEMENTATION

- *The Harmonization Pipeline Addresses the Challenge of Inconsistent Product Names in Tax Records from:*

- Electronic Billing Machines (EBMs)
- Electronic Single Window (eSW) for customs/importation

- *Due to Manual, Multilingual Entries, Effective Data Integration Becomes Difficult. The Solution was Built with Five Core Modules:*

- Text Cleaning Module
- Language Detection and Translation
- Sentence Embedding Generation (MiniLM)
- Dimensionality Reduction (PCA + UMAP)
- Clustering Engine (KMeans and HDBSCAN)

V. RESULTS AND INSIGHTS

- *Dataset Summary*
- **Total Records**: 4,124,005
- **Unique Harmonized Product Names**: 425,103
- **Language Breakdown**:
✓ English: 1,080,651 (26.2%)
✓ Translated (non-English): 3,043,354 (73.8%)

- *Clustering Performance*

- **KMeans Clusters**: 20 (Broad categories)
- **HDBSCAN Clusters**: 6,295 (Fine-grained clusters)
- **Noise Points**: 167,085 (Typos, rare labels)

- *Interpretation:*

- KMeans provided high-level groupings.
- HDBSCAN exposed subtle naming inconsistencies.
- Noise points often indicated potential errors or outliers.

- *Example Harmonized Items*

Table 1 Example Harmonized Items

Original Name	Price (RWF)	Harmonized Name
a3 paper	15,000	a paper
impapuro a4 4	12,000	a paper
papier brustol a3	1,000	a paper
baguette	2,000	baguette
baguette	5,000	baguette
baguette	8,000	baguette
envelopes	1,500	envelopes
enveloppes	2,000	envelopes
enveloppes	4,000	envelopes

- *Underpricing and Anomaly Detection*

The harmonized clusters enabled comparison of import/purchase values against sales prices for the same items. This revealed cases of underpricing and potential misreporting:

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/25aug118

- Key Insight: Several harmonized product clusters showed significant sales price variation for the same product.
- Example Analysis:
- Cluster “a paper” showed prices ranging from 1,000 RWF to 15,000 RWF.
- Similar discrepancies were noted in “baguette” and “envelopes” clusters.

VI. DISCUSSION AND IMPACT

The harmonization system effectively unified over 4,124,005 million multilingual product records into 425,103 standardized names, significantly improving data consistency. The large proportion of non-English entries highlighted the critical need for multilingual processing.

Clustering results showed broad categorization via K-Means and detailed groupings with HDBSCAN, uncovering subtle variations and data quality issues like typos and rare labels. Importantly, the system identified significant price discrepancies within harmonized product clusters (e.g., “a paper”), revealing potential underpricing and misreporting. This validates the system’s ability to enhance tax anomaly detection.

Impact-wise, the solution improves tax compliance monitoring, reduces manual data cleaning efforts, and provides actionable insights for policy and audit functions. Its modular design allows scalability and adaptation to other sectors and languages, supporting Rwanda Revenue Authority’s goal of efficient, digital tax administration.

VII. CONCLUSION

This study demonstrated the effectiveness of NLP and clustering techniques in harmonizing multilingual product names within Rwanda’s tax datasets. The system improved data consistency, revealed pricing anomalies, and supported better tax compliance monitoring. Its scalable design offers strong potential for broader application in government data systems.

ACKNOWLEDGMENT

The author expresses sincere appreciation to **Dr. Pacifique NIZEYIMANA**, academic supervisor, for continuous guidance and encouragement. Special thanks to the **Rwanda Revenue Authority**, especially the **Strategy and Risk Analysis Department**, for providing datasets and technical insights. Gratitude is also extended to the **Adventist University of Central Africa** and fellow MSc Big Data Analytics classmates for their collaboration and valuable feedback.

REFERENCES

[1]. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of NAACL-HLT 2019, 4171–4186. https://doi.org/10.48550/arXiv.1810.04805

[2]. Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP). https://doi.org/10.48550/arXiv.1908.10084

[3]. McInnes, L., Healy, J., & Melville, J. (2018). UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. arXiv preprint arXiv:1802.03426. https://doi.org/10.48550/arXiv.1802.03426

[4]. Campello, R. J. G. B., Moulavi, D., & Sander, J. (2013). Density-Based Clustering Based on Hierarchical Density Estimates. In Advances in Knowledge Discovery and Data Mining (pp. 160–172). Springer. https://doi.org/10.1007/978-3-642-37456-2_14

[5]. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

[6]. Turney, P. D., & Pantel, P. (2010). From Frequency to Meaning: Vector Space Models of Semantics. Journal of Artificial Intelligence Research, 37, 141–188. https://doi.org/10.1613/jair.2934

[7]. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. arXiv preprint arXiv:1301.3781. https://doi.org/10.48550/arXiv.1301.3781

[8]. Camacho-Collados, J., & Pilehvar, M. T. (2018). From Word to Sense Embeddings: A Survey on Vector Representations of Meaning. Journal of Artificial Intelligence Research, 63, 743–788. https://doi.org/10.1613/jair.1.11259

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/25aug118

FIGURES

```
tqdm.pandas()
##Text Preprocessing
# Function to clean item names
def clean_item_name(text):
    if isinstance(text, str):
        text = text.strip()                          # Remove leading/trailing spaces
        text = re.sub(r'\s+', ' ', text)             # Replace multiple spaces with one
        text = re.sub(r'[^\w\s]', '', text)          # Remove special characters and punctuation
        return text.lower()                          # Convert to lowercase
    return pd.NA

# Apply cleaning with progress bar
df_all_Import_purch_sales['item_name'] = df_all_Import_purch_sales['item_name'].progress_apply(clean_item_name)

# Drop rows with empty item names
df_all_Import_purch_sales.dropna(subset=['item_name'], inplace=True)

# Reset index
df_all_Import_purch_sales.reset_index(drop=True, inplace=True)

# View first few cleaned names
df_all_Import_purch_sales[['item_name']].head()
```

Fig 1 Text Preprocessing

```
# Load or process dataset
if os.path.exists(CHECKPOINT_PATH):
    print("✅ Loading checkpoint file with previous results...")
    df = pd.read_parquet(CHECKPOINT_PATH)
else:
    print("📥 Loading original dataset and cleaning item names...")
    df = pd.read_parquet(INPUT_PATH)

    df['item_name'] = df['item_name'].progress_apply(clean_item_name)
    df.dropna(subset=['item_name'], inplace=True)
    df['is_english'] = None
    df.reset_index(drop=True, inplace=True)

    # Batch detection
    BATCH_SIZE = 50000
    to_process_idx = df[df['is_english'].isnull()].index
    total_to_process = len(to_process_idx)
    batches = math.ceil(total_to_process / BATCH_SIZE)

    print(f"\n🚀 Starting language detection for {total_to_process:,} rows in {batches} batches...")

    for batch_num in range(batches):
        start_idx = batch_num * BATCH_SIZE
        end_idx = min(start_idx + BATCH_SIZE, total_to_process)
        batch_indices = to_process_idx[start_idx:end_idx]

        print(f"\n🔄 Processing batch {batch_num + 1} / {batches} (rows {start_idx + 1} to {end_idx})...")
        df.loc[batch_indices, 'is_english'] = df.loc[batch_indices, 'item_name'].progress_apply(is_english_lang)

        df.to_parquet(CHECKPOINT_PATH, index=False)
        print(f"💾 Checkpoint saved after batch {batch_num + 1}")

# Save English/Non-English subsets
df_english = df[df['is_english'] == True]
df_non_english = df[df['is_english'] == False]

df_english.to_parquet(ENGLISH_PATH, index=False)
df_non_english.to_parquet(NON_ENGLISH_PATH, index=False)
```

Fig 2 Language Detection

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/25aug118

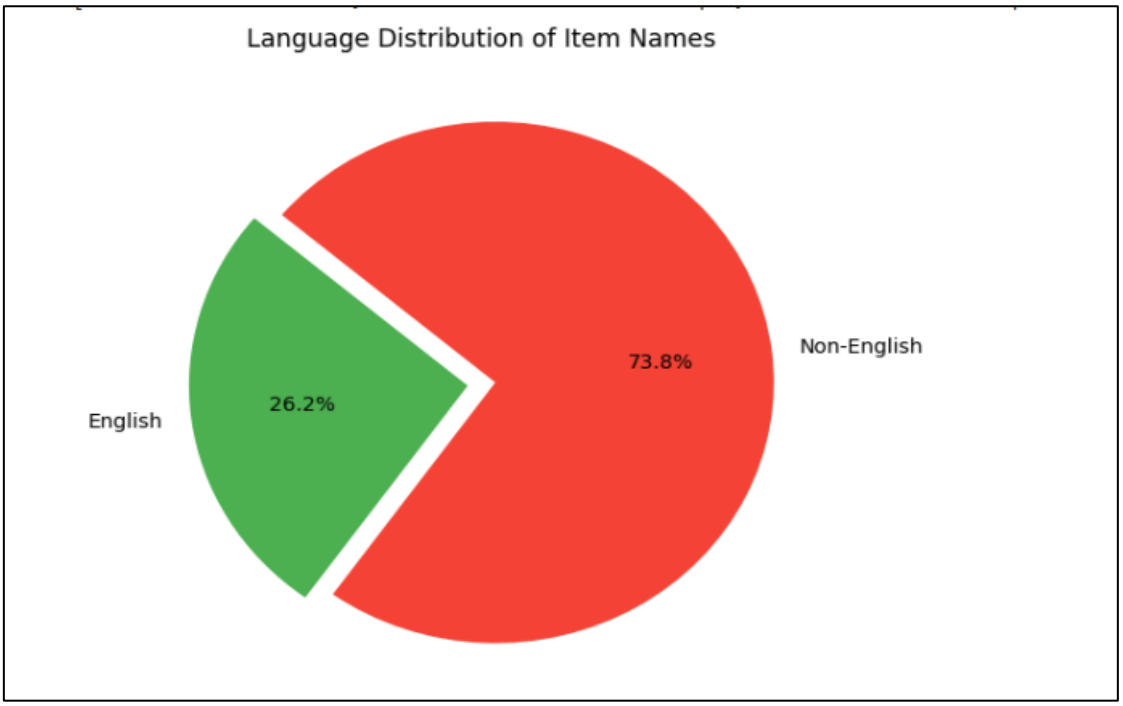

Fig 3 Language Distribution for English and Non-English

```python
# 2 Configurations
BASE_PATH = "/content/drive/MyDrive/colab_language_detection"
SOURCE_PATH = f"{BASE_PATH}/unique_non_english_items.parquet"
TRANSLATED_PATH = f"{BASE_PATH}/unique_non_english_translated.parquet"
API_KEY = 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'  # Replace with your paid API key
BATCH_SIZE = 100
MAX_RETRIES = 5
RETRY_BACKOFF = 2  # exponential: 2^retries
SAVE_INTERVAL = 10  # save every 10 batches

# 3 Load Dataset (resumable logic)
if os.path.exists(TRANSLATED_PATH):
    print("Resuming from saved translations...")
    df_translate = pd.read_parquet(TRANSLATED_PATH)
else:
    df_translate = pd.read_parquet(SOURCE_PATH)
    df_translate['translated_name'] = None  # Add column if not exists

# 4 Filter untranslated rows
not_translated = df_translate['translated_name'].isna()
indices = df_translate[not_translated].index.tolist()
total = len(indices)
print(f"Items left to translate: {total:,}")

# 5 Batch Translation Function (Google Translate API)
def translate_batch(texts, target_lang='en'):
    url = "https://translation.googleapis.com/language/translate/v2"
    params = {
        'q': texts,
        'target': target_lang,
        'format': 'text',
        'key': API_KEY
    }
    try:
        response = requests.post(url, data=params)
        if response.status_code != 200:
            print(f"HTTP Error {response.status_code}: {response.text}")
            return None
        data = response.json()
        return [t['translatedText'] for t in data['data']['translations']]
    except Exception as e:
        print(f"Exception during API call: {e}")
        return None
```

Fig 4 Non-English Translation

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/25aug118

Text	Language	Confidence	Translation
sukali	ln	0.8	sugar
Sugar	en	1	Sugar
Sucre	fr	0.55	Sugar
Sukari	ha	0.64	Sugar

Fig 5 Language Detection Results and Statistics

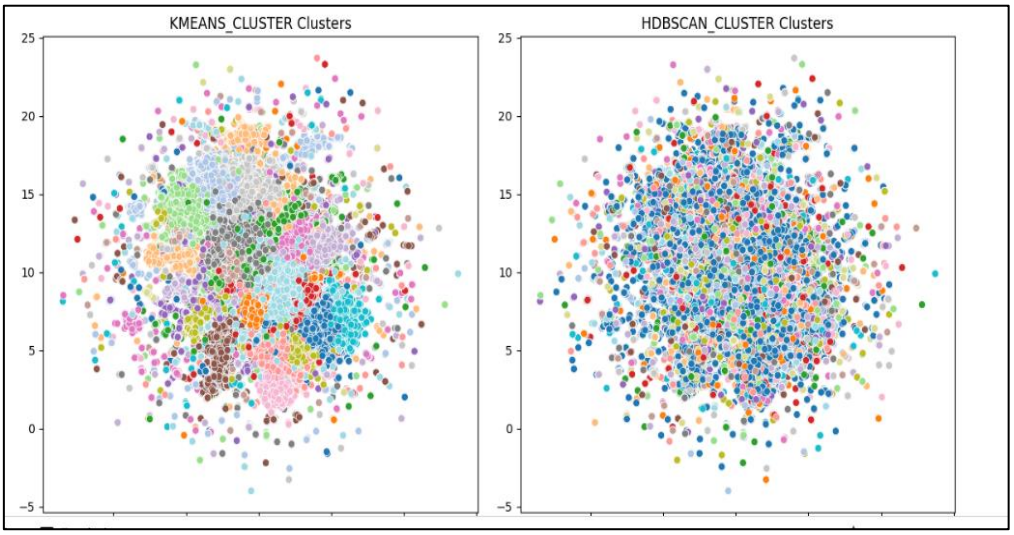

Fig 6 HDBSCAN and K-Means Clustering Results