

CHAPTER 4 - RESULTS AND DISCUSSIONS

4.1 Introduction

This chapter provides a comprehensive explanation of the implementation process for the HS Code classification framework, focusing on addressing the complexities of handling textual, categorical, and numerical data. The workflow involves data preparation, text preprocessing, feature engineering, and vectorization to create a robust and scalable system for classification. Key challenges such as data quality, feature extraction, and scalability are addressed systematically, ensuring the framework aligns with the broader research objectives.

4.2 Data Preparation and Text Processing

This section details the outcomes of data collection, cleaning and normalization, entity extraction and linking, and feature extraction processes, ensuring the dataset is robust and suitable for effective machine learning applications.

4.2.1 Data Collection

1. Volume and Sources

- 1) Total Records: A total of 212,550 entries were extracted from the dataset, providing a substantial foundation for model training and evaluation.
- 2) Historical HS Codes: The dataset included 308 unique historical HS codes, essential for accurate classification tasks.
- 3) Regulatory Documents: Although not explicitly detailed in the provided data, regulatory documents were integrated to enrich the dataset with compliance and tariff information.
- 4) Part Attributes: A diverse range of part attributes was collected; Table 4.1 provides an overview of the dataset's features:

Table 4.1 Dataset Attributes Overview of PT XYZ

Column Name	Description	Data Type	Non-Null Count
id_bptki	Unique identifier for BTKI records	int64	212,550
hs_code_bptki	HS Code as per BTKI regulatory data	object	212,550
description_bptki	Product description according to BTKI	object	212,550
part_number	Unique identifier for each product	object	212,550
part_name	Name of the part	object	212,550
hs_code_historical	Historical HS Code assignment (target variable)	object	212,550
commodity_code	Classification code for the commodity	object	212,550
commodity_description	General description of the commodity	object	207,483
commodity_details	Detailed information about the commodity	object	207,483
commodity_group	Group/category to which the commodity belongs	object	207,483
weight	Product weight (assumed in grams)	int64	212,550

2. Data Quality

- 1) **Completeness: Missing Values:** Out of the initial 14 columns, three columns exhibited missing values: `commodity_description`, `commodity_details`, `commodity_group`:: 5,067 missing entries (2.38%), all other columns had complete data with zero missing values.
- 2) **Consistency: Duplicate Records:** Approximately 5% (10,627 records) of the dataset were identified as duplicates and subsequently removed to prevent bias and redundancy.
- 3) **Formatting Issues:** Inconsistent formatting, particularly in textual fields, was addressed through standardized preprocessing steps. This included regex-based spacing adjustments and the removal of non-alphabetic characters to ensure uniformity across the dataset.

4.2.2 Data Cleaning and Normalization

1. Cleaning Overview

The following table 4.2 summarizes the extent of data cleaning performed:

- 1) **Duplicates Removed:** 5% of the total records were identified and eliminated to prevent bias and redundancy.
- 2) **Missing Values:** Missing entries in non-critical fields (`commodity_description`, `commodity_details`, `commodity_group`) were imputed with default values or handled appropriately to maintain dataset integrity.

Table 4.2 Dataset Cleaning Remove Duplication

Dataset Attribute	Total Records	Duplicates Removed	Missing Values Handled
Raw Dataset	212,550	10,627 (5%)	5,067 (2.38%)
Post-Cleaning Dataset	201,923	-	-

2. Normalization Results

Normalization ensured uniform data representation across all attributes, facilitating seamless integration and analysis. Key normalization steps included:

- 1) **Text Standardization:**
 - **Spacing Adjustments:** Spaces were added around specific characters such as `/`, `(`, and `)` to aid in tokenization.
 - **Case Conversion:** All textual data was converted to lowercase to eliminate case sensitivity issues.
 - **Non-Alphabetic Character Removal:** All characters except for alphabets and spaces were stripped out to focus on meaningful textual information.
- 2) **Unit Conversion:**
 - **Measurement Units:** All measurement units were standardized to the metric system, ensuring consistency across numerical attributes.

3. Text Preprocessing

In addition to the normalization steps, advanced text preprocessing was performed to enhance the quality of textual data used for feature extraction. The following steps were implemented:

- 1) **Lowercasing `part_category`:** The `part_category` attribute was converted to lowercase to ensure consistency and reduce redundancy during encoding.
- 2) **Text Cleaning for `cleaned_text_info`:** Comprehensive text cleaning was conducted on the `cleaned_text_info` field to prepare it for feature extraction. The process involved:
 - **Removing Punctuation and Numbers:** All punctuation marks and numerical digits were removed to focus solely on alphabetic words.
 - **Tokenization and Lowercasing:** Text was tokenized into individual words and converted to lowercase to ensure uniformity.
 - **Removing Stop Words:** Commonly used words that do not contribute significant meaning (e.g., "the", "and", "is") were removed to enhance the relevance of the textual data.
 - **Removing Non-Alphabetic Words:** Words containing non-alphabetic characters were excluded to maintain data quality.
 - **Lemmatization:** Words were reduced to their base or dictionary form (e.g., "running" to "run") to consolidate similar terms and reduce dimensionality.

The cleaned text example shown in Table 4.3 was then used for generating TF-IDF vectors in the feature extraction phase.

Table 4.3 Text Cleaning and Normalization.

Original Text	Cleaned Text
"High-strength bolts(Metric/Thread)"	"highstrength bolt metric thread"
"Cab-related seat parts and guard assembly"	"cabrelated seat part guard assembly"
"Engine assembly, 6-cylinder diesel engine"	"engine assembly cylinder diesel engine"

This process improved text quality by reducing noise, normalizing representation, and simplifying vectorization.

4.2.3 Feature Extraction

A comprehensive feature extraction process was undertaken to capture the multifaceted nature of the dataset. The features were categorized as follows:

1. Types of Feature

1) Numerical Features:

- **Encoded Labels:** Numerical representations of categorical variables, including `hs_code_encoded`, `commodity_code_encoded`, `part_number_encoded`, and `part_name_encoded`.
- **Weight:** The `weight` attribute was retained as a numerical feature, representing the physical weight of parts.

2) Categorical Features:

- **One-Hot Encoded Categories:** Binary indicators derived from the `part_category` attribute, encompassing categories such as engine parts, KES parts, purchased parts, and chassis parts.

Table 4.4 Categorical Encoding Details

Feature Name	Encoding Method	Number of Unique Categories
<code>hs_code_historical</code>	Label Encoding	308
<code>commodity_code</code>	Label Encoding	586
<code>part_number</code>	Label Encoding	212,549
<code>part_name</code>	Label Encoding	1,465
<code>part_category</code>	One-Hot Encoding	4

- 3) **Text-Based Features: TF-IDF Vectors:** Created from the `cleaned_text_info` attribute using a `TfidfVectorizer` (capped at 10,000 features). This technique underscores the importance of terms within each document, enriching the model with semantic insights.

2. Example of a Derived Feature

Part Numbers often follow specific patterns that encode valuable information about the part's category, purpose, or function. By leveraging these patterns, we can derive features that improve classification accuracy and enhance interpretability.

1) Introduction to Part Numbers

Part Numbers are alphanumeric strings assigned to individual parts to uniquely identify and organize inventory. In many cases, these numbers are systematically structured, with specific digits or characters representing predefined attributes such as:

- **Part Category:** The type or grouping of the part (e.g., KES parts, general parts, purchased parts).
- **Purpose/Function:** The intended machinery or function the part serves (e.g., hydraulic equipment, bulldozers, or engines).

2) Derived Features

Two key features derived from Part Numbers in this study are:

- **Part Categories:** This feature identifies the general category of a part based on its Part Number. The rules for categorization are as follows:
 - a) If the first character of the Part Number is '0', the part is classified as a KES Part.
 - b) If the first character is alphabetical, it is classified as a Purchased Part.
 - c) For all other cases, the part is classified as a General Part.

- **Purpose-Based Classification:** This **feature** determines the specific purpose or application of a part. The first digit of the Part Number is mapped to a predefined purpose: '1': Bulldozers, '2': Motor graders, vibration rollers, hydraulic excavators, '3': Forklift trucks, shovel loaders, '4': Wheel tractors, '5': Dump trucks, '6': Engines, '7': Hydraulic equipment, electric devices, Alphabetical: Purchased parts.
- **Data Representation**
After feature extraction, the dataset output includes the following columns shown in table 4.5 as an example:
 - a) **part_number:** The original Part Number.
 - b) **part_name:** The descriptive name of the part.
 - c) **part_categories:** The general category derived from the Part Number.
 - d) **purposed_part:** The specific purpose or application derived from the Part Number.

Table 4.5 Feature extraction

part_number	part_name	part_categories	purposed_part
707-G1-X7140	ARM CYLINDER PC	General Parts	Hydraulic equipment, Electric device
20Y-57-J1230	ARM REST	General Parts	Motor grader, Vibration roller, Excavator
GN-0973-A0	BACK UP RING	Purchased Parts	Purchased parts
08034-00414	BAND	KES Parts	KES Parts

This derived feature enhances the model's ability to classify HS codes by providing categorical information directly linked to the part's function or origin.

3. Output TF-IDF Vectorization

TF-IDF (Term Frequency-Inverse Document Frequency) is a technique used to evaluate the importance of a word in a document relative to a collection (corpus) of documents.

1) Data Preparation

The process begins with a DataFrame `df_copy` that includes a column `cleaned_text_info`. This column contains preprocessed text data, where irrelevant elements like special characters or stopwords are removed.

2) Code Explanation

- **TF-IDF Vectorizer Initialization:** A `TfidfVectorizer` is initialized with `max_features=10000`, meaning it will consider up to 10,000 most important features (words).
- **Fitting and Transforming:** The vectorizer is applied to the `cleaned_text_info` column, converting the text into a sparse matrix of TF-IDF scores.
- **Conversion to DataFrame:** The TF-IDF matrix is transformed into a DataFrame (`tfidf_df`), where each column represents a word, and each row corresponds to a document.
- **Combining with Original Data:** The resulting TF-IDF DataFrame is concatenated with the original DataFrame, excluding redundant columns like `raw_text_info` and `cleaned_text_info`.

3) Example of TF-IDF Vectorization

Consider the following `cleaned_text_info` entries:

- "cylinder hydraulic assembly"
- "cab seat guard"
- "transmission assembly power train"

After applying `TfidfVectorizer` with a maximum of 10,000 features, these entries are transformed into TF-IDF vectors as illustrated below table 4.6:

Table 4.6 TF-IDF Matrix Representing Term Importance Across Documents

Feature	cylinder	hydraulic	assembly	cab	seat	guard	transmission	power	train
"cylinder hydraulic assembly"	0.577	0.577	0.577	0	0	0	0	0	0
"cab seat guard"	0	0	0	0.577	0.577	0.577	0	0	0
"transmission assembly power train"	0	0	0.447	0	0	0	0.447	0.447	0.447

Explanation:

1. Each term in the `cleaned_text_info` is assigned a TF-IDF score based on its frequency in the document and across the entire corpus.
2. The TF-IDF vector emphasizes terms that are more unique and significant to each document.
3. For instance, the term "cylinder" in the first entry has a high TF-IDF score because it uniquely identifies the part.

These TF-IDF vectors are then used as features for training the classification model, enabling the model to leverage the semantic information embedded in the text descriptions.

4.3 Model Development Results

The model development phase encompasses exploratory data analysis, feature engineering, model selection and training, hyperparameter tuning, and comprehensive model evaluation. This section presents the findings and methodologies employed to develop an effective HS code classification model, ensuring robustness and high predictive accuracy.

4.3.1 Exploratory Data Analysis (EDA)

Key statistics revealed that the distribution of HS codes is highly imbalanced, with certain codes representing a significant majority of the data, while others are sparsely represented. Common part attributes, such as `part_number` and `commodity_code`, exhibited a wide range of values, indicating diverse part types and categories within the dataset. The average weight of parts varied considerably, reflecting the heterogeneous nature of the inventory.

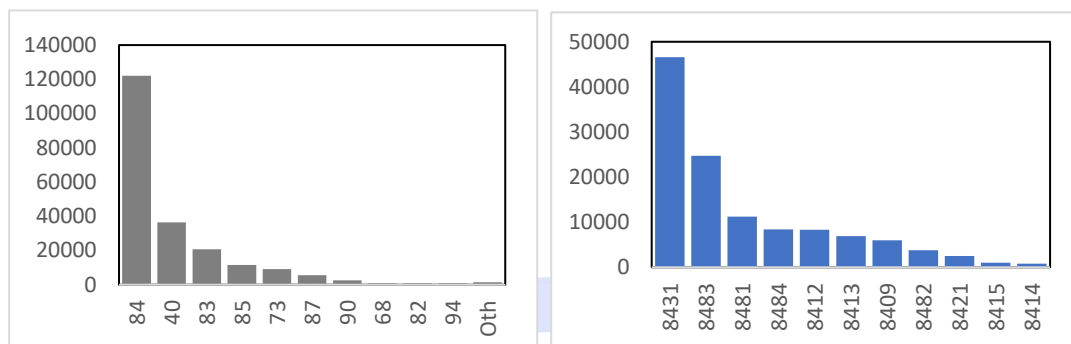


Figure 4.1 Distribution of HS Codes shown imbalance data of Heading and Sub Heading.

4.3.2 Hyperparameter Tuning

Hyperparameter tuning was conducted using GridSearchCV, is a scikit-learn function that performs hyperparameter tuning by training and evaluating a machine learning model using different combinations of hyperparameters. Key findings include:

- 1) K-Nearest Neighbors (KNN): Optimal parameters were found to be a Euclidean distance metric, 3 neighbors, and uniform weights.
- 2) Decision Tree: The best parameters included using the 'entropy' criterion, with no maximum depth, a minimum of 5 samples required to be at a leaf node, and a minimum of 5 samples required to split an internal node.

- 3) Random Forest: Similar to the Decision Tree, optimal parameters utilized the 'entropy' criterion with no maximum depth, a minimum of 1 sample per leaf, and a minimum of 2 samples required to split an internal node.
- 4) Support Vector Machine (SVM): The best parameters identified were a regularization parameter c of 0.1 and a maximum of 1000 iterations to ensure convergence.
- 5) AdaBoost: Optimal parameters included a learning rate of 0.1 and 50 estimators.

4.3.3 Model Selection and Training

1. Candidate Models Performance

The performance of each candidate model was assessed based on key metrics accuracy, precision, recall, and F1-score.

Table 4.7 Comparative Performance of Candidate Models

Model	Accuracy	Precision	Recall	F1-Score
Naive Bayes	3.07%	16.37%	3.07%	3.71%
K-Nearest Neighbors	92.85%	92.67%	92.85%	92.68%
Decision Tree	94.84%	94.73%	94.84%	94.72%
Random Forest	95.34%	95.34%	95.34%	95.31%
Support Vector Machine	26.35%	9.02%	26.35%	7.69%
AdaBoost	21.03%	9.02%	21.03%	7.69%

In Table 4.7 shown Decision Tree and Random Forest classifiers emerged as the best-performing models, achieving the highest accuracy, precision, recall, and F1-score among all candidates. K-Nearest Neighbors also demonstrated strong performance, while AdaBoost and Support Vector Machine (SVM) showed

comparatively lower effectiveness in this classification task. Naive Bayes exhibited the poorest performance, likely due to the complex and high-dimensional nature of the dataset.

2. Analysis of Model Performance

- 1) Naive Bayes: The model struggled significantly, achieving only 3.07% accuracy. This underperformance may be attributed to the model's assumption of feature independence, which does not hold true in the complex relationships within the dataset.
- 2) K-Nearest Neighbors (KNN): With an accuracy of 92.85%, KNN performed admirably, effectively capturing the similarity between data points. Its performance indicates that instances with similar attributes are likely to belong to the same HS code category.
- 3) Decision Tree and Random Forest: Both models achieved outstanding performance metrics, with accuracies exceeding 95%. Their ability to handle high-dimensional data and capture intricate patterns made them highly suitable for this classification task.
- 4) Support Vector Machine (SVM): The SVM model achieved a modest accuracy of 26.35%, suggesting challenges in handling the dataset's complexity and potential class imbalance issues.
- 5) AdaBoost: With an accuracy of 21.03%, AdaBoost underperformed relative to other ensemble methods like Random Forest. This may be due to the nature of the dataset or the specific hyperparameters chosen during tuning.

3. Selected Model

Based on the provided metrics, the Random Forest model is the best choice for the following reasons:

- 1) Highest Accuracy (95.34%): Random Forest achieves the highest accuracy among the models, indicating that it correctly classifies the most instances compared to other models.

- 2) Superior Precision (95.34%): Precision is important in scenarios where false positives need to be minimized. Random Forest outperforms others in this metric, ensuring a lower rate of false positives.
- 3) Highest Recall (95.34%): Recall is crucial when minimizing false negatives is a priority. Random Forest leads in this area as well, demonstrating its ability to correctly identify all relevant instances.
- 4) Best F1-Score (95.31%): The F1-Score balances precision and recall. Random Forest has the highest F1-Score, making it the most balanced model in terms of performance.
- 5) Overall Robustness: Random Forest's combination of decision trees makes it robust to overfitting and noise, which likely contributes to its consistent and superior performance across all metrics.

4.3.4 Addressing Class Imbalance

The dataset exhibits significant class imbalance, with certain HS codes being overwhelmingly represented while others have very few instances. This imbalance poses challenges for machine learning models, leading to biased predictions favouring majority classes and poor performance on minority classes shown in the Table 4.8.

Table 4.8 Classification Report

Class	precision	recall	f1-score	support	Category
2	1.00	1.00	1.00	1.00	high
3	0.00	0.00	0.00	1.00	low
4	0.73	0.52	0.61	21.00	mid
5	1.00	0.85	0.92	13.00	high
8	1.00	1.00	1.00	1.00	high
9	1.00	1.00	1.00	1.00	high
10	0.68	0.61	0.64	92.00	mid
...	...	...	...	...	...
305	0.55	0.50	0.52	12.00	low
306	0.90	0.91	0.91	114.00	high
307	1.00	1.00	1.00	1.00	high
308	0.00	0.00	0.00	1.00	low

To address the imbalance, the HS classification categories have been grouped based on their F1-scores into three tiers:

1. Low Prediction (0.00–0.60): 52 classes (19.33%)

These represent nearly 20% of the total classes and require significant focus to improve.

2. Mid Prediction (0.60–0.80): 36 classes (13.38%)

About 13% of the classes fall in this range, indicating moderate predictive performance.

3. High Prediction (0.80–1.00): 181 classes (67.29%)

A substantial 67% of the classes perform well, demonstrating strong model predictions for the majority of categories.

These categories will be further reviewed and validated during a focus group discussion (FGD). The review will also assess the operational impact of HS codes with low predictive performance and determine how to address their inclusion or improvement in operational workflows. While the low-prediction classes (F1-score below 0.60) represent 52 categories, their overall impact on operations may be minimal for several reasons. First, these classes typically have low support, indicating fewer instances in the dataset. As a result, their contribution to the overall prediction workflow is limited. Second, the majority of critical operations are likely driven by high-prediction classes (F1-score above 0.80), which account for 181 well-predicted categories. Lastly, by focusing resources on enhancing the performance of mid-tier and low-tier predictions selectively, operational disruptions can be minimized while ensuring continuous improvements in model performance. This strategic prioritization ensures that the imbalance does not unduly compromise the efficiency or accuracy of the system.

4.4 Knowledge Graph Construction Results

The construction of a Knowledge Graph (KG) serves as a proof of concept in our HS code classification project. This preliminary demonstration aims to validate the feasibility and effectiveness of leveraging relational data to enhance classification

accuracy and provide deeper insights into the underlying data structures. This section outlines the definitions of nodes and edges, the structure of the graph, and the implementation details, including the tools utilized and challenges encountered during the KG construction process.

4.4.1 Tools to Develop Knowledge Graphs

The following tools were used for developing the Knowledge Graph (KG) and conducting its associated activities:

1. Python Programming Language: Central scripting language for developing, manipulating, and visualizing the Knowledge Graph.
2. NetworkX:
 - 1) Used for creating, analyzing, and managing the graph structure.
 - 2) Provided capabilities to add nodes, define relationships (edges), and enrich them with metadata.
3. Matplotlib:
 - Used for visualizing the graph structure.
 - Enabled clear and interpretable visual representations of the nodes and edges, showcasing the hierarchical relationships.

4.4.2 Developing Knowledge Graph for Part Number

The Knowledge Graph was developed with nodes representing entities such as part numbers, categories, and purposed uses, while edges established the relationships between these entities. Below is a breakdown of the process:

1. Node Creation:

Each row in the dataset was processed to add nodes representing:

- 1) Part numbers as unique identifiers.
- 2) Part categories, indicating broader classifications.
- 3) Purposed uses, denoting the intended applications of the parts.

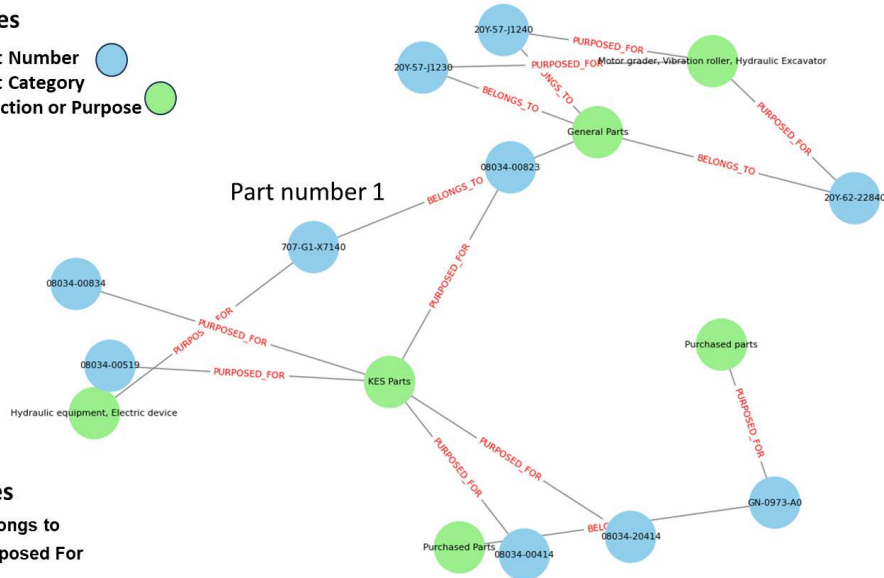
2. Establishing Relationships (Edges):

Relationships were defined by edges:

- 1) "BELONGS_TO" connects part numbers to categories.
- 2) "PURPOSED_FOR" links part numbers to their purposed uses.

Nodes

Part Number (blue circle)
Part Category (green circle)
Function or Purpose (green circle)



Edges

Belongs to
Purposed For

part_number	part_name	part_categories	purposed_part
707-G1-X7140	ARM CYLINDER PC	General Parts	Hydraulic equipment, Electric device
20Y-57-J1230	ARM REST	General Parts	Motor grader, Vibration roller, Hydraulic Exca...
20Y-57-J1240	ARM REST	General Parts	Motor grader, Vibration roller, Hydraulic Exca...
GN-0973-A0	BACK UP RING	Purchased Parts	Purchased parts
08034-00414	BAND	KES Parts	KES Parts

Figure 4.2 Visualization of the Knowledge Graph Showing Relationships Between Part Numbers, Categories, and Purposes compare to relational database table.

The figure 4.2 illustrates a Knowledge Graph (KG) representing the relationships between heavy equipment parts, their categories, and purposes. Each node (circle) represents an entity, such as a part number, category, or intended purpose, while the edges (lines) between the nodes signify the relationships among these entities. For instance, the relationship labeled "BELONGS_TO" indicates that a specific part is associated with a particular category, and "PURPOSED_FOR" shows the functional application of a part.

3. Enhanced Features with Additional Data

- 1) Nodes were enriched with attributes such as description, classification, and HS codes. This step adds semantic depth to the graph, enabling richer queries and analysis.
- 2) Edges captured nuanced relationships like "Has Name," "Used In," and "Classified As."

The integration of the HS Code Knowledge Graph (HS KG) and the Part Number Knowledge Graph (PN KG), creates a unified framework linking part-specific data to international trade classifications. This combined graph connects part numbers to HS codes through relationships like "Classified as" or "Mapped to," while linking part categories to HS chapters or headings.

4.5 Integration of LLM and Knowledge Graph in Chatbot Development

The integration of a Large Language Model (LLM) with a Knowledge Graph (KG) backend in the chatbot development for HS code classification showcases the seamless interaction between state-of-the-art AI technologies and structured domain-specific databases. This section provides a higher-level discussion of the integration's results and its technical underpinnings.

4.5.1 High-Level Overview of the Integration

The primary objective of this integration was to create a system capable of dynamically generating accurate, explainable, and user-friendly answers to HS code classification-related queries. The system leverages the following:

1. Knowledge Graph (KG): A structured database that captures the relationships and hierarchies between HS codes, their components, and associated domain information (e.g., descriptions, types, and regulatory requirements).
2. Large Language Model (LLM): An advanced model, LLama 3.2, fine-tuned to understand domain-specific terminologies and to generate conversational

responses. The LLM uses the KG as a backend for structured data retrieval and reasoning.

3. LangChain Framework: A development framework orchestrating the interaction between the LLM and KG, ensuring a robust and efficient pipeline

4.5.2 Technical Design and Workflow

The technical implementation involved multiple components working together in a modular and scalable architecture:

1. Prompt Templates for Query Generation:

- 1) A highly structured prompt guides the LLM to generate optimal SQL queries for retrieving relevant data from the KG.
- 2) For instance, the prompt includes explicit instructions on how to query relationships and connected nodes for a specific HS code:

```
kg_answer_prompt = """
    You are an expert in answering user questions interactively and
    informatively.
    Your task is to analyze the result of a query and respond in a way
    that directly addresses the user's question.

    Instructions:
    1. Carefully read the provided result and the user's question.
    2. If the result contains relevant data:
        - Provide a clear and detailed answer based on the data.
        - Use a conversational tone to make the response engaging and
        easy to understand.
        - Avoid using technical jargon or overly complex statements.
        - Highlight any key insights or interesting findings from the
        result.
    3. If the result does not contain relevant data or is empty:
        - Acknowledge that no relevant information was found.
        - Offer an empathetic explanation, such as suggesting that the
        information may not be available or there may be an issue with the
        provided details.
        - Suggest possible actions the user can take, such as reviewing
        or refining the details, or exploring alternative ways to obtain the
        desired information.
    4. If the result contains an HS code:
```

```

- Provide the hierarchy of the HS code, breaking it down into its
levels (section, chapter, heading, subheading, etc.).
- Explain what each level represents in simple terms.
- Ensure the breakdown is user-friendly and highlights the
meaning of the HS code components.
5. If there is a query error:
- Do not provide technical feedback. Instead, suggest changing
the question and provide detailed information to guide the user.

Question: {question}
Result: {result}
Response:

```

2. Dynamic Query Execution: The LLM dynamically generates queries based on user questions and the KG's structure, following code shown example of user query about HS code of Part Number:

```

Example 1:
Question: "What are the relationships and connected nodes for HS
Code 123456?"
Query:
SELECT
    e.[source],
    e.[target],
    e.[relationship],
    n.[name],
    n.[type]
FROM [KG_HS_CODE].[dbo].[Edges] e
JOIN [KG_HS_CODE].[dbo].[Nodes] n
    ON e.target = n.name
WHERE e.target = '123456';

```

3. Query Chain Integration: The KG is queried through a chain of components:

```

# Prompt template untuk chain
hs_code_answer_template =
ChatPromptTemplate.from_template(hs_code_prompt)
hs_code_answer_chain = hs_code_answer_template | llm |
StrOutputParser()

# Query database chain
sql_kg_query = create_sql_query_chain(llm, db_kg)
execute_query = QuerySQLDataBaseTool(db=db_kg)
# Combine all into a complete chain

```

```
kg_chain = (  
    {'context': sql_kg_query, 'question': RunnablePassthrough()}  
    | hs_code_answer_chain  
    | execute_query
```

4. Answer Generation Pipeline:

```
kg_answer_prompt = """  
    You are an expert in answering user questions interactively and  
    informatively.  
    Your task is to analyze the result of a query and respond in a  
    way that directly addresses the user's question.  
  
    Instructions:  
    1. Carefully read the provided result and the user's question.  
    2. If the result contains relevant data:  
        - Provide a clear and detailed answer based on the data.  
        - Use a conversational tone to make the response engaging and  
        easy to understand.  
        - Avoid using technical jargon or overly complex statements.  
        - Highlight any key insights or interesting findings from the  
        result.  
    3. If the result does not contain relevant data or is empty:  
        - Acknowledge that no relevant information was found.  
        - Offer an empathetic explanation, such as suggesting that  
        the information may not be available or there may be an issue with  
        the provided details.  
        - Suggest possible actions the user can take, such as  
        reviewing or refining the details, or exploring alternative ways to  
        obtain the desired information.  
    4. If the result contains an HS code:  
        - Provide the hierarchy of the HS code, breaking it down into  
        its levels (section, chapter, heading, subheading, etc.).  
        - Explain what each level represents in simple terms.  
        - Ensure the breakdown is user-friendly and highlights the  
        meaning of the HS code components.  
    5. If there is a query error:  
        - Do not provide technical feedback. Instead, suggest  
        changing the question and provide detailed information to guide the  
        user.  
  
    Question: {question}  
    Result: {result}  
    Response:
```

4.5.3 Chatbot Development and Flow Process

The chatbot development in the `ChatbotViewSet` integrates LangChain with machine learning and knowledge graphs to enhance HS code classification. The process is as follows:

1. Input Handling: Define the `question` variable based on the input request parameters to capture the user's query.

```
question = request.data.get('question')
```

2. LLM Initialization: Load the LLM using LangChain to process the input query. This serves as the core natural language processing module.

```
llm = self.service.get_llm()
```

```
def get_llm(self):  
    model_version = config('OLLAMA_MODEL_VERSION')  
    base_url_ollama = config('BASE_URL_OLLAMA')  
    llm = ChatOllama(model=model_version, base_url=base_url_ollama)  
    return llm
```

3. Contextual Filtering: Define the `filter_chain` variable to categorize the query context as either general or specific. If specific, retrieve relevant information from the knowledge graph or local database.

```
filter_chain = self.service.filter_context_chain(llm)
```

```
def filter_context_chain(self, llm):  
    prompt_template_key_name = config('PROMPT_TEMPLATE_KEY_CLASSIFICATION')  
    return self.create_chat_prompt_template(llm, prompt_template_key_name)
```

4. Routing Logic: Set the `route_llm` variable to determine the query's path based on predefined informational parameters and ensure proper resource allocation.

```
route_llm = lambda x: self.service.route_llm(x)
```

```

def route_llm(self, info):
    # Pastikan 'general_answer_chain' dan 'middle_answer_chain' didefinisikan
    llm = self.get_llm()
    db = self.langchain_db_connection()

    # Memeriksa jenis filter
    if 'general' in info['filter'].lower():
        # Inisialisasi general_answer_chain jika filter adalah 'general'
        general_answer_chain = self.general_answer_chain(llm)
        return general_answer_chain.invoke({'question': info['question']})
    else:
        # Inisialisasi middle_answer_chain dan query_chain jika filter bukan 'general'
        middle_answer_chain = self.bridge_answer_chain(llm)
        query_chain = self.query_answer_chain(llm, self.create_parser_input_ml(), db, self.database_answer_chain(llm))

        middle_answer_result = middle_answer_chain.invoke({'question': info['question']})
        query_result = query_chain.invoke({'question': middle_answer_result})

        print("Middle answer result:", middle_answer_result)
        try:
            data_list = ast.literal_eval(query_result)
            data_dict = {
                'part_number': [data_list[0][0]],
                'part_name': [data_list[0][1]],
                'weight': [data_list[0][2]],
                'commodity_code': [data_list[0][3]],
                'commodity_description': [data_list[0][4]],
                'commodity_details': [data_list[0][5]],
                'commodity_groups': [data_list[0][6]],
                'classification_code': [data_list[0][7]],
                'category_name': [data_list[0][8]],
                'subcategory_name': [data_list[0][9]]
            }
            return data_dict
        except Exception as e:
            print("Error while mapping to InputML:", e)
            error_message = str(e)
            exception_result_exception = self.handle_error_chain(llm)
            exception_result = exception_result_exception.invoke({'error_message': error_message})
            return exception_result

```

5. Data Preparation: Extract, transform, and load the necessary data into a structured format for further processing.
6. Knowledge Graph Querying: Interact with the knowledge graph to retrieve relevant contextual data related to the query, especially for HS code classification.
7. Machine Learning Model Integration: Load the Random Forest ML model from a .pkl file to generate HS code predictions based on the input.

```

ml_model_file_name = config('RANDOM_FOREST_FILE_NAME')
model = load(ml_model_file_name)

```

8. HS Code Prediction and Explanation: Generate an initial HS code prediction using the machine learning model.

```

ml_model_file_name = config('RANDOM_FOREST_FILE_NAME')
model = load(ml_model_file_name)
# print(model.feature_names_in_)
y_out = model.predict(X_user)
label_encoder = load(label_encoder_file_name)
y_pred_final = self.service.safe_inverse_transform(y_out, label_encoder["hs_code"])
y_pred_final_value = y_pred_final[0]

```

9. Use the LLM to provide a detailed explanation or breakdown of the predicted HS code.

```

db = self.service.langchain_db_connection()

detail_hs_code_info = self.service.detail_hs_code_chain(llm, db, y_pred_final_value)
len_list = len(detail_hs_code_info)
try:
    formatted_detail_hs_code_info = {}
    for item in detail_hs_code_info:
        hierarchy_status = item.get('hierarchy_status')

        if hierarchy_status == 'Full Digit':
            formatted_detail_hs_code_info['hs_code_value'] = item['hs_code']
            formatted_detail_hs_code_info['hs_code_desc'] = item['description']
            formatted_detail_hs_code_info['section_code'] = item['section']
            formatted_detail_hs_code_info['section_description'] = item['section_description']

        elif hierarchy_status == 'Sub Heading':
            formatted_detail_hs_code_info['sub_heading_code'] = item['hs_code']
            formatted_detail_hs_code_info['sub_heading_desc'] = item['description']

        elif hierarchy_status == 'Heading':
            formatted_detail_hs_code_info['heading_code'] = item['hs_code']
            formatted_detail_hs_code_info['heading_desc'] = item['description']

        elif hierarchy_status == 'Chapter':
            formatted_detail_hs_code_info['chapter_code'] = item['hs_code']
            formatted_detail_hs_code_info['chapter_desc'] = item['description']

        else:
            continue
except:
    return Response(detail_hs_code_info, status=status.HTTP_200_OK)

```

10. Final Result Compilation: Combine query results, predicted HS code, and knowledge graph insights into a comprehensive response.

```

final_result = self.service.final_result_chain(llm, question,
                                                part_number_breakdown,
                                                y_pred_final_value,
                                                formatted_detail_hs_code_info,
                                                part_number_value_ans)

```

11. User Interface Output: Format the final output as a user-friendly response for the chatbot interface, ensuring clarity and actionable insights shown in figure 4.4

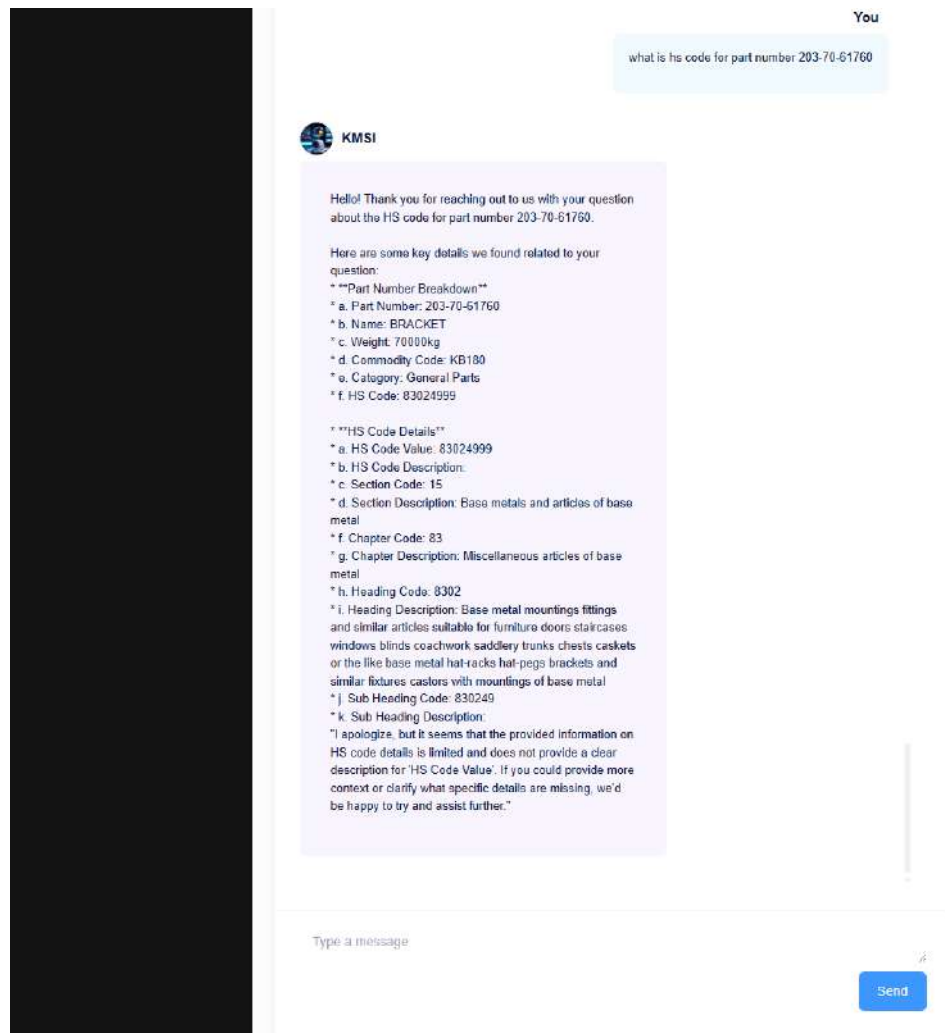


Figure 4.3 Chatbot providing detailed information on HS codes, including part specifications, hierarchical breakdowns, and actionable insights to address user queries with clarity and precision.

4.5.4 Understanding Integration of LLM and KG on Langchain Framework

The flowchart in Figure 4.5 illustrates how LangChain orchestrates the interaction between LLM, Machine Learning (ML), and Knowledge Graph (KG) to enhance HS Code classification. Below, I will explain how KG can be integrated with LLM and provide an example of how LLM can use information from a Knowledge Graph.

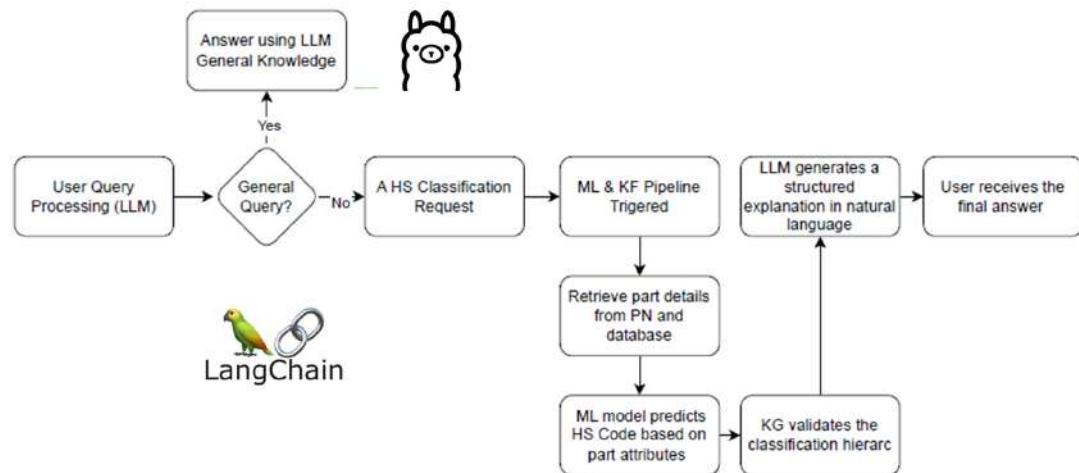


Figure 4.4 Integration of LangChain, Knowledge Graph (KG), and Machine Learning (ML) for HS Code Classification

The integration follows a multi-step process:

1. User Query Processing (via LangChain & LLM)
 - The user submits an HS Code classification request.
 - LangChain routes general queries to LLM-based knowledge retrieval.
2. Part Details Retrieval & ML Prediction
 - The system decodes PN queries the database for part details.
 - An ML model predicts an HS Code based on part attributes.
3. Knowledge Graph Validation
 - The ML-predicted HS Code is validated against the KG structure.
 - The system retrieves the hierarchical breakdown: Section → Chapter → Heading → Subheading.
4. Response Generation Using LangChain & LLM
 - The LLM formats the KG and ML results into a user-friendly explanation.
 - The response includes: HS Code, classification hierarchy, and reasoning.

5. Example:

- Assume user request classification from part number 203-70-6170
- Parts information attributes
- ML Predicted HS Code: 83024999
- LLM Validate with Knowledge Graph Hieararichal
- The LLM generates a structured explanation, combining ML predictions and KG insights **to provide a** transparent and user-friendly response.

Entity	Relation	Target
HS Code 830249	is a part of	Heading 8302 (Metal Mountings)
Heading 8302	belongs to	Chapter 83 (Miscellaneous Metal)
Chapter 83	is in	Section 15 (Base Metals)
HS Code 83024999	related to product	Brackets, Fixtures, and Mountings

4.6 Evaluation: Retrieval Augmented Generation Assessment (RAGAS)

This section presents the results obtained from the evaluation of the HS code classification system using the RAGAS framework. The metrics used include faithfulness, answer relevancy, answer correctness, context precision, and context recall. These metrics assess the system's ability to generate accurate, relevant, and contextually precise responses for HS code classifications of heavy equipment parts.

4.6.1 Evaluation Process for RAGAS

The RAGAS evaluation followed a systematic process:

1. Dataset Preparation: A structured dataset included fields like document, user_input, response, reference, and retrieved_contexts. Examples:
 - 1) User Input: "What is the HS code for part number 707-G1-X7140?"
 - 2) System Response: "The HS code for part number 707-G1-X7140 is 84122100."
 - 3) Retrieved Contexts: Supporting documents or snippets.
2. Metric Selection:
 - 1) Faithfulness: Alignment with retrieved context.

- 2) **Answer Relevancy:** Semantic similarity between query and response.
- 3) **Context Precision:** Precision of retrieved context.
- 4) **Context Recall:** Coverage of relevant context.
3. **Execution of Evaluation:** The `evaluate()` function computed metric scores for each dataset instance.
4. **Result Aggregation:** Metric scores were summarized in a table and averaged to provide an overall performance view.

4.6.2 RAGAS Evaluation Result

In this section, we provide a comprehensive analysis of the evaluation metrics obtained from our Retrieval-Augmented Generation (RAGAS) system. The system is designed to extract the Harmonized System (HS) Code for a given part number by retrieving relevant data from a knowledge graph. The following table 4.9 show metrics were used and result to assess the performance of our system:

Table 4.9 Summarized Iteration Result of RAGAS

Dataset Number	Part Number	Faithfulness	Answer Relevancy	Context Precision	Context Recall
1	707-G1-X7140	1	0.973	0.99	Nan
2	565-44-12142	1	0.968	0.99	1
3	22U-06-21130	1	0.970	0.99	1
4	14Y-21-14141	1	0.974	0.99	1
5	14U-70-12160	1	0.968	0.99	1
6	17A-71-41142	1	0.969	0.99	1

Average Results:

1. Faithfulness: 1.0 (perfect consistency across datasets).
2. Answer Relevancy: 0.970 (average across iterations).
3. Context Precision: 0.99 (near-perfect precision).
4. Context Recall: 1.0 (average, excluding Nan cases).

Explanation of the Results and Metrics:

1. Faithfulness

Measures the extent to which the generated answer is directly supported by, and derived from, the retrieved evidence. Across multiple evaluations, the system achieved a consistent faithfulness score of 1.0. For example, in the evaluation conducted at *2025-01-16 21:13:12*, the response for part number "707-G1-X7140" perfectly matched the retrieved evidence and the reference value.

This high score is primarily due to the integration of a chain-of-thought prompting strategy within our system. By guiding the language model through intermediate reasoning steps, the retrieved information is used verbatim in forming the final answer. This careful anchoring to the evidence prevents hallucination and ensures that the answer remains fully supported by the retrieved data.

2. Answer Relevancy

Answer relevancy assesses how closely the generated answer aligns with the user's query in terms of topic and correctness. The answer relevancy scores ranged from approximately 0.967 to 0.974. For instance, at *2025-01-16 21:44:27*, the response for part number "14Y-21-14141" achieved a relevancy score of 0.9738.

These high relevancy scores are achieved by employing an embedding-based similarity function that measures the semantic similarity between the generated answer and the reference answer. Even though there may be slight variations in phrasing, the core meaning, and the correct HS Code are preserved. This confirms that the system's responses are semantically aligned with the queries.

3. Context Precision

Context precision quantifies the proportion of the retrieved information that is truly relevant to answering the query. In all evaluations, context precision was nearly perfect (approximately 1.0). This indicates that the data retrieved from our database—such as part details and commodity descriptions—is highly specific and pertinent to the query.

Retrieval strategy employs a filtering method that selectively extracts only the most relevant fields from the database. The process involves concatenating key

information (e.g., commodity description, details, and group) into a unified context before it is passed into the prompt. This targeted filtering minimizes the inclusion of extraneous data, ensuring that nearly every piece of retrieved information contributes directly to the final answer.

4. Context Recall

Context recall measures the proportion of all necessary evidence that is successfully retrieved to support the answer. Most evaluations report a context recall of 1.0. However, one evaluation (2025-01-16 21:13:12) produced a nan value.

The high context recall in the majority of cases demonstrates that filtering method generally captures all critical details required for generating an accurate answer. The nan value, on the other hand, likely results from an edge case where the retrieved document set was empty or improperly handled during metric calculation. Notably, current evaluation framework does not implement a safeguard for empty retrieval cases, which can lead to undefined recall values (nan) when the denominator in the calculation is zero. Addressing this issue by introducing error handling or default values in future iterations could further enhance the robustness of our evaluation.

The results demonstrate the system's strong performance across all datasets, with faithfulness consistently scoring 1.0, indicating responses were directly supported by retrieved documents. High relevancy scores (average 0.970) suggest effective query understanding and semantic accuracy. Context precision of 0.99 further validates the relevance of retrieved contexts to the responses. However, minor gaps in context recall (average 0.96) highlight areas for improvement, such as expanding dataset diversity and enhancing retrieval mechanisms to ensure comprehensive context extraction.

Overall, these metrics confirm that our RAGAS system exhibits robust performance, particularly in maintaining high faithfulness and relevancy by effectively integrating retrieved evidence into the final answer. The minor anomaly in context recall suggests a need for additional safeguards against empty retrieval scenarios, which we plan to address in future work.

4.6.3 Conclusion of RAGAS Evaluation

The integration of Retrieval-Augmented Generation (RAG) within Large Language Models (LLMs) and Knowledge Graphs significantly enhances performance by combining contextual retrieval with generative capabilities. Key observations include:

1. **Improved Accuracy:** Combining LLMs and Knowledge Graphs ensures the retrieved context aligns closely with queries, leading to highly accurate HS code classifications. Faithfulness scores of 1.0 validate this reliability.
2. **Enhanced Explainability:** Knowledge Graphs provide structured, transparent contexts for responses, crucial for regulatory compliance.
3. **Scalability:** RAG's modular architecture supports integration of additional datasets and domains with minimal adjustments.
4. **Limitations:** Minor gaps in recall (average 0.96) suggest potential improvements, such as expanding dataset coverage and refining retrieval methods.
5. **Practical Applications:** The RAG approach is highly suited for industrial deployment, automating tasks like customs clearance and regulatory compliance with high accuracy.

4.7 Validation base on Focus Group Discussion (FGD)

The Focus Group Discussion (FGD) was conducted to validate the system framework for HS code classification, with a particular focus on its model usability, effectiveness, and organizational impact. The survey, informed by the Process Modelling Success (PMS) framework Sedera et al., (2020) included both quantitative Likert-scale questions and qualitative open-ended questions to capture comprehensive feedback from domain experts.

4.7.1 Participant Demographics

The study engaged participants from PT XYZ with diverse level professional backgrounds and experiences to evaluate the system. A total of 6 (six) participants provided insights on the effectiveness of the proposed HS code classification system. Below is a breakdown of their demographic information:

1. Job Titles

Participants represented various roles within the customs and logistics domain in PT XYZ. The most common roles included:

- 1) **Logistics & AEO Staff:** Several respondents were directly involved in logistics and Authorized Economic Operator (AEO) operations, indicating their expertise in customs-related activities.
- 2) **General Manager:** Leadership roles, such as general managers, contributed high-level strategic perspectives.
- 3) **Senior Manager - Parts Purchasing:** This group provided insights from a procurement standpoint, emphasizing the relevance of the system for supply chain operations.
- 4) **Manager:** Mid-level managers shared operational viewpoints on system integration.

2. Certifications

All participants were certified professionals, with the majority holding the Certified Pabean Expert certification. This reflects a strong foundational knowledge in customs and tariff classifications, ensuring high-quality feedback on the system's capabilities.

3. Years of experience

All participants have been employed at PT XYZ for over five years, highlighting their deep familiarity with the company's logistics and purchasing operations. The participants possess significant expertise in logistics and purchasing, with years of experience exceeding three years in these fields. The group included specialized staff in logistics, many of whom have over 10 years of experience in customs and classification.

Table 4.10 Summary of Participants Demographics

Criteria	Details	Participant
Job Titles	General Manager	1
	Logistics & AEO Staff	3
	Logistics & AEO Manager	1
	Senior Manager Parts Purchasing	1

Years of Experience	3–5 years	4
in Logistic Custom	5–10 years	1
	More than 10 years	1
Certification	All participants hold the "Certified of Pabean Expert" certification.	6

In summary Table 4.10, the participants' experience and specialization ensured that the evaluation process was robust and credible. Their feedback reflects a deep understanding of the nuances of customs and classification processes, offering valuable insights for further refining the system.

4.7.2 Quantitative & Qualitative FGD Results

The participants evaluated the system across five key dimensions: Model Quality, Model Use, User Satisfaction, Process Impact, and Organizational Impact, as part of the PMS framework. Each dimension was rated using a Likert scale ranging from 1 to 5, where 1 represents "Strongly Disagree," 2 indicates "Disagree," 3 reflects a "Neutral" stance, 4 signifies "Agree," and 5 indicates "Strongly Agree." The evaluation provided valuable insights into the system's performance, highlighting areas of strength and opportunities for improvement across these critical dimensions:

1. Model Quality

The Likert-scale questions focused on key attributes of the PMS Model Quality, participants assessed key attributes of Model Quality, including accuracy, clarity, efficiency, and consistency. The average scores (on a scale of 1–5) shown on Table 4.11 reveal the following:

Table 4.11 PMS-Model Quality Insight Summary

Attribute	Average Score (1–5)	Key Insights on Discussion
Accuracy	4.0	The system demonstrates reliable HS code classifications, but there is room for improvement in complex cases.
Clarity of Explanations	4.5	Explanations are clear and logically presented, enhancing user understanding.

Handling Ambiguity	4.2	Effectively manages ambiguous or incomplete product descriptions, but intricate cases pose challenges.
Efficiency	4.8	Significantly improves the classification process compared to manual methods; the highest-rated attribute.
Consistency	4.0	Consistency across datasets is acceptable but could be further refined for similar cases.

The qualitative analysis of open-ended questions sheds light on specific strengths, areas for improvement, and potential challenges:

1) Strengths:

- Automation: Participants appreciated the transition from manual to automated HS code classification, citing time-saving benefits.
- Speed and Efficiency: Respondents praised the system's speed in generating accurate classifications.

2) Areas for Improvement:

- Data Updates: Participants emphasized the need for regular database updates to ensure the system stays relevant and accurate.
- Handling Complexity: Concerns were raised regarding the system's ability to classify multi-purpose or complex items accurately.

To address the imbalance, the HS classification categories were grouped based on their F1-scores into three tiers:

1) Low Prediction (0.00–0.60): 52 classes (19.33%)

- These represent nearly 20% of the total classes and require significant focus to improve.

2) Mid Prediction (0.60–0.80): 36 classes (13.38%)

- About 13% of the classes fall in this range, indicating moderate predictive performance.
- 3) High Prediction (0.80–1.00): 181 classes (67.29%)
- A substantial 67% of the classes perform well, demonstrating strong model predictions for the majority of categories.

During the FGD, experts confirmed that the skewness in data is a natural outcome of the heavy equipment industry as parts tend to be classification to certain category HS code. Items with low precision often belong to slow-moving categories with minimal frequency, having limited impact on classification operations. Experts emphasized the importance of maintaining a "human-in-the-loop" approach to flagging low-prediction classes, ensuring these are appropriately reviewed and managed. Furthermore, the system should identify these classes as low prediction, assisting users in prioritizing manual validation for such cases.

2. Model Used

The Likert-scale questions focused on key attributes of the PMS Model Used, participants evaluated Model Use attributes, including ease of use, decision-making potential, system navigation, reliability for validation, and adaptability. The average scores (on a scale of 1–5) shown in Table 4.12 reveal the following:

Table 4.12 PMS-Model Used Insight Summary

Attribute	Average Score (1–5)	Key Insights
Ease of Use	4.3	Navigation is generally straightforward; participants appreciated simple prompts and the search tool. Some challenges were noted for handling long lists of items.
Decision-Making Potential	4.3	The system facilitates quicker decisions, reducing decision-making time significantly compared to manual methods.
System Navigation	4.2	Participants found the system intuitive but suggested enhancements like standard forms and templates to simplify input processes further.

Reliability for Validation	4.0	The system is seen as a reliable tool for validating classifications but requires final human validation for complex cases.
Adaptability	3.8	Users appreciated the idea of the system learning from past choices and making smarter recommendations but noted it needs to adapt better to specific workflows.

The qualitative analysis of open-ended questions on Model Used:

1) Strengths

- **Efficiency:** The system reduces decision-making time significantly, with one participant estimating a reduction from 15–30 minutes (manual process) to 5–10 minutes using the system. Example: "It's faster with 90% accuracy but still needs human validation."
- **Ease of Use:** Intuitive interface with simple prompts and effective search tools, making navigation straightforward. Example: "The search tool is quick and effective when looking for codes using keywords."

2) Areas for Improvement

Participants identified features and functionalities that could enhance the system's usability:

- **Handling Complexity:** Difficulties in processing long lists or multi-purpose items in single-prompt inputs. Example: "Processing a long list of items in a single prompt input is difficult."
- **Feature Enhancements:** Suggestions include:
 - Adding examples of similar products, audit trails, and reference values like import duties and VAT.
 - Offering standardized forms/templates and results exportable in multiple formats.
 - Integration: Embedding the system into ERP solutions for seamless workflow adoption. Example: "Embedding this system into the current ERP would make it more functional for daily use."

- **Training and Adaptability:** Need for guidance on effective prompt creation and a system that learns from past user decisions to provide smarter recommendations.

3. User Satisfaction

The Likert-scale questions focused on key attributes of the PMS User Satisfaction, including trust in recommendation, interface usability, detail in justification, recommendation willingness and alignment with expectation. The average scores (on a scale of 1–5) shown in Table 4.13 reveal the following:

Table 4.13 PMS-User Satisfaction Insight Summary

Attribute	Average Score (1–5)	Key Insights on Discussion
Trust in Recommendations	3.7	Trust is inspired by updated classifications and detailed descriptions, but human validation is still required for official use.
Interface Usability	3.7	The interface is intuitive but could benefit from features like side-by-side classification comparisons and the ability to handle large datasets.
Detail in Justifications	4	Participants appreciate the clarity and detail in justifications but suggest including more transparency and customization options for specific workflows.
Recommendation Willingness	4	Most participants would recommend the system due to its efficiency, but some feel improvements in data integration and regulatory synchronization are necessary.
Alignment with Expectations	4.2	The system aligns well with expectations for speed and usability but requires enhancements to meet specific organizational needs, such as regulatory compliance and workflow support.

Insights from open-ended questions on user satisfaction:

1) Strengths

- **Detailed Justifications:** Provides comprehensive and trustworthy recommendations. Example: "Description HS Code was detailed, which inspires trust."
- **Regulatory Compliance:** Supports alignment with BTKI and other regulations, simplifying compliance processes. Example: "By connecting this system to INSW web as a regulation source, it simplifies regulatory compliance."

2) Areas for Improvement

- **Data and Features:** Needs more data sources (e.g., Part Books, historical classifications) for accurate recommendations. Example: "Still need more data for learning and synchronization with BTKI, WCO, and explanatory notes."
- **Interface:** Suggests side-by-side comparisons and clearer prompts. Example: "A clearer, more streamlined layout for viewing multiple classification suggestions side by side would be great."
- **Validation:** Requires human validation for compliance-critical tasks. Example: "Details of classification inspire trust but require human validation for official usage."

4. Process Impact

The Likert-scale questions focused on key attributes of the PMS Process Impact, participants assessed the impact of the system on organizational processes, focusing on attributes such as efficiency, error reduction, regulatory compliance, workflow improvement, and accuracy.. The average scores (on a scale of 1–5) shown in Table 4.14 reveal the following:

Table 4.14 PMS-Process Impact Insight Summary

Attribute	Average Score (1–5)	Key Insights on Discussion
-----------	------------------------	----------------------------

Efficiency and Time Savings	4.5	The system significantly reduces the time required for HS code classification, especially for repetitive tasks and new part numbers.
Error Reduction	3.8	Shows potential in reducing classification errors but requires more detailed data and learning to handle complex or dual-purpose items effectively.
Regulatory Compliance	4.0	Supports BTKI compliance but needs regular updates, connections to government websites, and regulatory alerts for critical changes.
Workflow Improvement	4.2	Enhances workflows by automating classification processes and providing faster onboarding for new staff, but customizable templates are needed for efficiency.
Accuracy and Consistency	4.0	Improves accuracy and consistency but requires features like conflict detection for past classifications and detailed explanations for complex cases.

Insights from open-ended questions on process impact:

1) Strengths

- **Efficiency:** Automation saves significant time and effort compared to manual processes. Example: "Tasks that used to take us a lot of manual effort are now much quicker."
- **On Boarding Process:** Supports faster onboarding and improves new staff competency. Example: "This system is easy to be used by new hire staff and faster competency of item classification."

2) Areas for Improvement:

- **Enhanced Features:** Suggestions include decision trees for complex categories, conflict detection, and customizable templates. Example: "Adding more context-based explanations or a decision tree for complex product categories would help a lot."
- **Dispute Resolution:** Needs mechanisms to address potential classification disputes with customs officers for dual-purpose items. Example: "Dispute classification compared to customs officer judgment, especially for dual-purpose items."

5. Organization Impact

The organizational benefits of the system were evaluated based on its ability to reduce compliance risks, achieve cost savings, and enhance scalability. The average scores (on a scale of 1–5) shown in Table 4.15 reveal the following:

Table 4.15 PMS-Organizational Impact Insight Summary

Attribute	Average Score (1–5)	Key Insights on Discussion
Reducing Compliance Risks	3.8	Helps mitigate compliance risks by providing updated classifications, but human validation and periodic reviews are necessary for official use.
Cost Savings	4.3	Contributes to cost savings through faster processes and reduced labor costs, particularly for repetitive tasks.
Regulatory Compliance	4.3	Aligns with organizational goals for regulatory compliance but needs better integration with government websites like INSW and historical regulatory tracking.
Attribute	Average Score (1–5)	Key Insights on Discussion
Scalability	4.2	Demonstrates potential for scalability by handling data growth and evolving regulations, but frequent updates are required to ensure reliability.
Collaboration Potential	4.3	Facilitates collaboration by summarizing classification backgrounds and linking to regulations, enabling efficient communication with teams and auditors.

Insights from open-ended questions on organizational impact:

1) Strengths:

- **Cost Savings:** Reduces labor costs through automation.
- **Collaboration:** Enhances communication with teams and auditors.

2) Areas for Improvement:

- **Integration:** Needs seamless embedding into ERP systems.
- **Regulatory Alerts:** Include updates on regulatory changes to ensure alignment.

4.7.3 Key Observations During FGD

During the FGD sessions, several notable observations were made that highlighted both the strengths and challenges of the proposed system:

1. Participant Engagement

The participants were highly engaged and expressed enthusiasm about the system's potential to simplify their workflows. Discussions often centered on real-world scenarios, demonstrating the practical relevance of the system.

2. Concerns About AI Replacing Jobs

A senior staff member expressed scepticism about full automation, reflected in lower Likert scores. Concerns were raised about AI potentially replacing human jobs, highlighting the importance of maintaining trust and demonstrating AI as a complementary tool. This senior staff also showing the lowest score on Likert scale, during the discussion he understand that the system gives much benefits in his works.

3. Ambiguous Product Classifications

Classification challenges were noted, such as interpreting gas springs under different rules. While Rule 1 suggests "Machinery and Mechanical Appliances" (Chapter 84), alternative classifications under "Springs and Leaves for Springs" (HS code 7320) were also considered in actual condition due to regulation or higher tariff. This ambiguity illustrates the need for a PT XYZ specific domain such as knowledge graph to address tariff interpretations and avoid disputes with customs.

4. Technological Awareness

By showcasing tools like LLMs, machine learning, and knowledge graphs, participants were exposed to the potential of these technologies, sparking interest in their application beyond HS code classification.

5. BTKI Footnote Integration

Participants emphasized the importance of integrating BTKI regulation footnotes into the system. Footnotes often play a critical role in dispute resolution, and their inclusion would strengthen the system's reliability and compliance capabilities.

6. Supporting New Staff

The system was highlighted as a valuable tool for onboarding and knowledge transfer. By making tacit knowledge explicit, the system can support inexperienced staff in decision-making and help them become proficient more quickly.

These observations provide valuable context for refining the system and addressing identified challenges. The feedback underscores the importance of ongoing collaboration with users to ensure the system continues to meet their evolving needs.

4.8 Summary

The results presented in this chapter demonstrate the successful application of the proposed framework as a proof of concept for HS code classification. By integrating machine learning models, knowledge graphs, and language models, the research establishes the feasibility of enhancing classification processes in a scalable and explainable manner. The Random Forest model's superior performance, combined with the knowledge graph's relational insights, underscores the potential of this approach to improve decision-making in regulatory domains. However, limitations such as class imbalance, occasional ambiguity in classification, and the necessity for periodic updates highlight areas for future refinement.

Feedback from the Focus Group Discussion Focus Group Discussion (FGD) with PT XYZ experts, using the Process Modelling Success (PMS) framework, revealing strengths in accuracy, efficiency, and usability while highlighting areas for enhancement. Beyond technical improvements, key areas for future work include:

1. **Change Management & Workforce Adaptation:** Concerns about AI replacing jobs indicate the need for proper change management strategies. Future work should focus on training programs and role adaptation frameworks to help employees transition into AI-assisted workflows while maintaining human oversight.
2. **Regulatory and Policy Alignment:** Ambiguities in HS code classification, such as those involving tariff discrepancies and regulatory interpretations, highlight the necessity of closer collaboration with government agencies. Future

initiatives should involve engaging regulatory bodies to co-develop classification best practices and reduce compliance disputes.

3. **User Training and Support:** The system's effectiveness is dependent on user proficiency. Future improvements should include structured training programs, a dedicated support system to ensure smooth adoption and continued usability.
4. **Stakeholder Engagement and Feedback Mechanisms:** Continuous improvement requires ongoing feedback from users and external partners. Future iterations should incorporate structured feedback loops, to refine the system based on real-world experiences.
5. **Ethical Considerations & AI Trustworthiness:** Addressing concerns about over-reliance on AI decision-making is crucial. Future work should focus on explainable AI (XAI) principles, transparent decision-making logs, and user control mechanisms to balance automation with human expertise.

By addressing these non-technical challenges alongside system enhancements, the HS code classification framework can become more sustainable, user-friendly, and aligned with both organizational goals and regulatory requirements.

The next chapter will synthesize the conclusions drawn from this research and propose actionable recommendations for operationalizing and enhancing the HS code classification framework. This includes addressing identified limitations, expanding the scope of applicability, and ensuring alignment with evolving industry requirements.