

CHAPTER 3 – RESEARCH METHODS

3.1 Research Framework

This chapter outlines the methodologies developed to address the challenges and gaps identified in Chapter 2 regarding HS code classification. The proposed framework integrates Machine Learning (ML), a Knowledge Graph (KG), and Large Language Models (LLMs), each contributing unique capabilities to enhance the classification process. This methodology is designed to ensure accurate, explainable, and scalable HS code predictions while accommodating evolving regulatory requirements and user feedback.

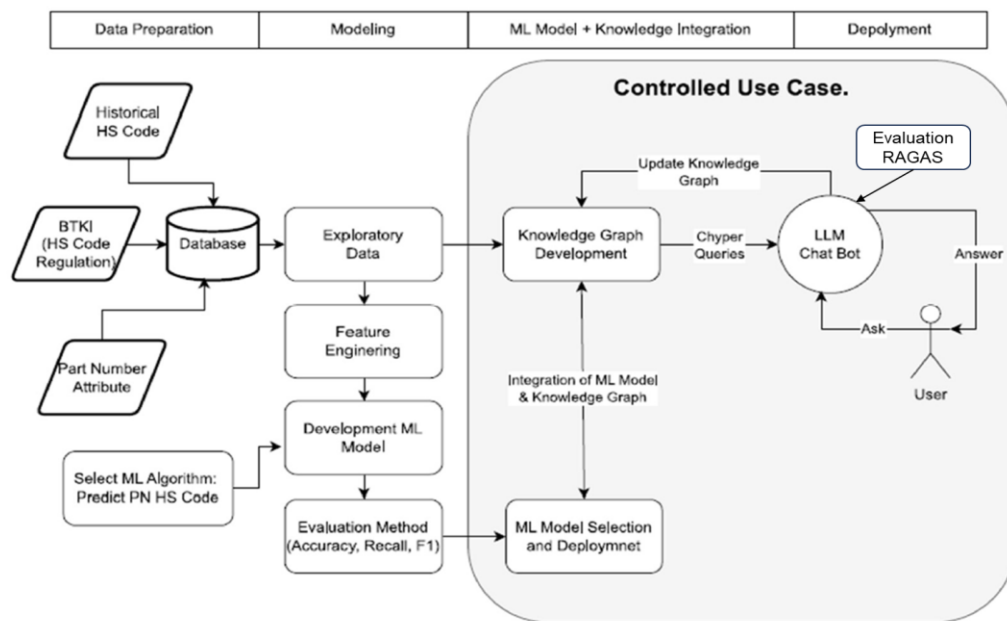


Figure 3.1 Depicts the overall structure of the research framework, illustrating the seamless integration of ML, KG, and LLMs across different phases of the HS code classification system.

The research framework shown in Figure 3.1 is organized into four key phases, as outlined below:

1. Data Preparation

This phase involves collecting and organizing data necessary for the classification process. The primary tasks include:

- 1) **Data Collection:** Gathering historical HS code data, regulatory guidelines, and detailed part attributes from various reliable sources.
- 2) **Data Storage:** Centralizing the collected data into a unified, scalable database to ensure accessibility and consistency across subsequent phases.

This phase ensures the availability of high-quality, structured data, forming the foundation for the classification framework. For a proof-of-concept, the scope of data preparation may be limited to demonstrate feasibility rather than exhaustive coverage.

2. Modelling Phase

The second phase focuses on building and refining predictive models for HS code classification. The steps involved are as follows:

- 1) **Exploratory Data Analysis (EDA):** Conducting a thorough analysis of the collected data to identify trends, patterns, and correlations relevant to HS code classification.
- 2) **Feature Engineering:** Extracting and developing relevant features from the dataset to improve model performance.
- 3) **Model Development:** Training multiple machine learning models using the processed dataset to predict HS codes effectively.
- 4) **Model Selection and Optimization:** Evaluating the models based on key metrics such as accuracy, precision, and recall. Hyperparameters are fine-tuned to identify the best-performing model for integration with subsequent components, with the tuning process limited to demonstrating feasibility as part of the proof-of-concept.

This phase establishes the core predictive capability of the framework.

3. Knowledge Graph Integration

To enhance the predictive model's accuracy and interpretability, this phase integrates the ML model with a Knowledge Graph:

- 1) **Integration with ML Models:** The KG is used to capture and represent relationships between part attributes, historical data, and HS codes. This integration enhances model predictions by incorporating domain knowledge and ensuring traceability of classifications.
- 2) **Enhancement of Accuracy and Explainability:** The combined ML-KG system improves the transparency of the classification process, providing users with an explanation of the relationships and reasoning behind the predictions.

4. System Deployment

The final phase involves implementing a simplified or prototype version of the fully integrated ML-KG system, focusing on demonstrating feasibility and user interaction as part of the proof-of-concept. This phase includes:

- 1) **Deployment:** Deploying the system in a production environment, ensuring scalability and reliability.
- 2) **User Interface:** Developing a user-friendly chatbot interface to enable seamless interaction with the system. Users can utilize the interface to classify new part numbers or validate existing HS code recommendations. The chatbot interacts directly with the KG to deliver precise and context-aware predictions.

This structured and multi-phase research framework ensures the development of a robust HS code classification system. By integrating ML, KG, and LLMs, the framework achieves a balance between predictive accuracy, scalability, and explainability while primarily aiming to validate the feasibility and potential of this integration as a proof-of-concept.

3.2 Data Collection and Preprocessing

This phase involves gathering and organizing data to build both the ML model and the KG. The sources of data include:

Collect Data Source:

1. **Historical HS Codes:** Data on previously classified heavy equipment parts, including part numbers and assigned HS codes from PT XYZ.

2. BTKI (Indonesian HS Code Regulations): Official regulatory guidelines providing the rules for assigning HS codes, forming the basis for the KG structure.
3. Parts Attributes: Technical specifications and product descriptions from internal databases and product catalogues, defining the characteristics of each part.

Preprocessing Steps:

1. Data Cleaning: Removing inconsistencies, errors, and handling missing data to ensure quality.
2. Normalization: Standardizing formats (e.g., text, units) for consistency across the dataset.
3. Entity Extraction and Linking: Identifying key entities such as part numbers, HS codes, and their relationships, which will later be stored in the KG.
4. Feature Extraction: Extracting relevant features from the data, such as part attributes and graph-based features for the ML model.

3.3 Model Development

This phase focuses on developing and training a Machine Learning (ML) model to predict HS codes using part attributes and historical data. The process is divided into several sub-phases, including exploratory data analysis, feature engineering, model training, and evaluation.

1. Exploratory Data Analysis (EDA)

Objective: Understand the dataset and uncover trends or features relevant to HS code classification.

Techniques:

- 1) Descriptive Statistics: Analyze dataset distributions, including 212,550 entries with attributes such as "part_number," "hs_code_historical," and "commodity_code."
- 2) Missing Data Handling: Address missing values in key columns like "commodity_description" and "commodity_details."

2. Feature Engineering

Types of Features:

- 1) Textual Features: Extracted from "raw_text_info," including tokenized and normalized text descriptions.
- 2) Categorical Features: Encoded features such as "part_category" and "commodity_code."

Techniques:

- 1) Text Processing: Apply TF-IDF vectorization to transform text into numerical features.
- 2) Categorization: Derive "part_category" from "part_number" using predefined rules.

3. Model Selection and Training

Candidate Models: These candidate models will be tested on imbalanced data to evaluate their robustness and ability to handle class imbalance effectively.

- 1) Naive Bayes: A probabilistic classifier based on Bayes' theorem, suitable for text-based classification tasks.
- 2) K-Nearest Neighbors (KNN): A non-parametric algorithm that predicts class labels based on the majority vote of the nearest neighbors.
- 3) Decision Tree: A tree-structured model that splits the dataset into subsets based on feature values, offering interpretability.
- 4) Random Forest: An ensemble method that builds multiple decision trees and combines their outputs for robust and accurate predictions.
- 5) Linear Support Vector Classifier (LinearSVC): A linear model for binary and multi-class classification that aims to maximize the margin between data points and the decision boundary.
- 6) AdaBoost with Decision Tree Base Estimator: An adaptive boosting method that combines multiple weak learners to create a strong classifier, reducing bias and variance.

4. Hyperparameter Tuning

Objective: Hyperparameter tuning is critical for optimizing model performance by identifying the best parameter combinations for each algorithm.

Process Using GridSearchCV:

- 1) Define a range of hyperparameters for each model (e.g., "alpha" for Naive Bayes, "n_neighbors" for KNN, etc.).
- 2) GridSearchCV, a method provided by Scikit-learn, systematically evaluates all possible combinations of specified hyperparameters across a parameter grid. It performs an exhaustive search to identify the best set of hyperparameters by applying cross-validation to each combination and selecting the one with the highest performance score (e.g., accuracy) on the validation set.

Parameters Tuned:

- 1) Naive Bayes: Adjusted the "alpha" parameter to handle smoothing for text classification.
 - 2) K-Nearest Neighbors (KNN): Explored parameters such as "n_neighbors," "weights," and "metric."
 - 3) Decision Tree: Tuned "criterion," "max_depth," "min_samples_split," and "min_samples_leaf."
 - 4) Random Forest: Optimized "n_estimators," "max_depth," and "min_samples_split" for robust predictions.
 - 5) LinearSVC: Tested regularization strength "C" and "max_iter" to ensure convergence.
 - 6) AdaBoost: Focused on "n_estimators" and "learning_rate" to balance learning speed and accuracy.
- #### 5. Model Evaluation

Objective: Model evaluation ensures that the chosen models perform effectively on unseen data and are robust enough to handle real-world scenarios.

Metrics for Evaluation:

- 1) Confusion Matrix: Provides a detailed breakdown of true positives, true negatives, false positives, and false negatives for classification performance.
- 2) Classification Report: Summarizes key metrics, including precision, recall, and F1-score for each class.

- 3) Accuracy: Measures the overall correctness of predictions.
- 4) F1-Score: Balances precision and recall to provide a single measure of performance, particularly useful for imbalanced datasets.

Testing Process:

- 1) Split data into training and test sets, ensuring no data leakage.
- 2) Evaluate the models using the defined metrics to identify strengths and weaknesses.
- 3) Selection Criteria: Models are selected based on their ability to balance accuracy, interpretability, and scalability. Models such as KNN, Decision Tree, Random Forest, and AdaBoost are particularly prioritized due to their proven performance.

Tools: Use Scikit-learn for:

- 1) Model training and evaluation.
- 2) Feature extraction and preprocessing (e.g., TF-IDF vectorization).

By following this streamlined approach, the ML model achieves reliable performance in predicting HS codes, ensuring applicability for practical use cases.

3.4 Knowledge Graph Construction

This phase focuses on constructing a Knowledge Graph (KG) to represent part attributes and the relationships between HS codes, enhancing the system's ability to interpret complex classification rules and support decision-making.

Node Definition: Nodes represent the fundamental entities within the Knowledge Graph. Examples include:

1. Part Numbers: Unique identifiers for heavy equipment parts, as shown in the dataset (e.g., "08034-00414").
2. HS Codes: Harmonized System codes assigned to parts, such as "84122100," serving as classification nodes. For example, HS code 84122100 pertains to "Hydraulic power engines and motors, linear acting (cylinders)." According to the Harmonized Tariff Schedule, the Most Favored Nation (MFN) duty rate for this

code is free, meaning imports under this classification are not subject to customs duties.

3. **Commodity Codes and Groups:** Representing categories like "MJ270" and groups such as "Cab/Guard."
4. **Function and Description:** Details on the function and technical descriptions, e.g., "Hydraulic Cylinder Ass'y" and "Compressor(EC) Related Parts."
5. **Regulatory Guidelines:** Rules and hierarchical classifications based on BTKI regulations.

Edges (Relationships): Edges define the relationships between nodes and capture the interactions within the Knowledge Graph. Examples include:

1. **"Classified As":** Links part numbers (e.g., "707-G1-X7140") to their corresponding HS codes (e.g., "84122100").
2. **"Belongs To Commodity Group":** Connects part numbers to commodity groups such as "Others" or "Cab/Guard."
3. **"Regulated By":** Associates parts and HS codes with relevant regulatory guidelines for traceability and compliance.

Graph Structure: The Knowledge Graph is a network of interconnected nodes and edges, organized to reflect the domain's hierarchy and relationships.

1. **Hierarchical Representation:** The KG mirrors the hierarchical structure of the HS code system, capturing parent-child relationships among codes. For instance, "84122100" is part of a broader classification of hydraulic systems. This structure enables reasoning over relationships between product attributes, commodity details, and their HS codes.
2. **Network Representation:** The graph structure allows for complex queries and inference, making it easier to analyze and interpret classification rules and relationships.

Implementation Details:

1. **Tools and Technologies:**

- 1) Pandas Library: Used for data preprocessing tasks such as handling missing values, normalizing data formats, and cleaning inconsistencies in the dataset.
- 2) NetworkX Library: Used for building and manipulating the graph structure programmatically. The Python script leverages NetworkX to define nodes (e.g., part numbers, categories, purposes) and edges (e.g., "BELONGS_TO," "PURPOSED_FOR") to create the Knowledge Graph.
- 3) Python Programming: Core scripting language used for integrating the preprocessing pipeline and building the Knowledge Graph.

2. Data Integration:

- 1) Data from the provided dataset is cleaned and normalized. For example, missing values in "commodity_description" and "commodity_group" are replaced with "other" to ensure consistent representation.
- 2) Nodes and edges are created based on relationships such as "Made Of" for material data (e.g., "Hydraulic Cylinder Ass'y"), and "Classified As" for HS codes.

By incorporating this data into a Knowledge Graph, the system ensures a structured, explainable, and accurate approach to HS code classification. This representation facilitates robust reasoning and supports user-friendly interactions for part categorization and compliance validation.

3.5 Integration of Machine Learning Model with Knowledge Graph

This section describes the integration of the Machine Learning (ML) model with the Knowledge Graph (KG) to improve the accuracy and interpretability of HS code predictions. The focus is on leveraging the synergy between ML and KG to enhance both the training process and the dynamic refinement of the KG itself. This integration demonstrated as a proof-of-concept, highlighting the feasibility and potential benefits of such an approach.

1. ML-KG Integration

- 1) Establishes a two-way interaction where the ML model recommends new relationships, nodes, and edges for the KG using vector similarity.

- 2) The KG provides enriched contextual and hierarchical information to improve the ML model's training and predictions, creating a feedback loop that enhances both components over time.

2. Implementation Overview

1) Vector similarity for recommendations:

- The ML model generates vector representations of part attributes.
- Compares them to existing KG nodes and edges using similarity metrics.
- Suggests additions or corrections to the KG to ensure completeness and relevance.

2) Dynamic KG updates:

- Incorporates ML recommendations iteratively to validate and add new information.
- Ensures the KG evolves alongside the ML model's learning process, demonstrating adaptability in the proof-of-concept.

3. Enhanced training set generation:

- 1) The KG enriches the ML model's training data by providing additional features derived from its relationships and hierarchies.
- 2) Supports more robust and context-aware predictions, proving the viability of integrating structured knowledge into the training pipeline.

4. Explainable Integration

1) Transparent recommendations:

- The system explains KG updates by detailing the features or patterns identified by the ML model.
- Ensures transparency and trust, critical for the proof-of-concept phase.

5. Explainability techniques:

- 1) Visualization tools highlight new nodes and edges in the KG structure.
- 2) Narrative reasoning generates human-readable explanations for recommendations, enabling validation by end-users.

6. Implementation Tools

- 3) Graph management: Tools like NetworkX are used for creating, querying, and updating the KG.

- 4) Vector processing: Libraries such as Scikit-learn or TensorFlow generate vector embeddings and compute similarities.
- 5) Programming language: Python ensures compatibility and flexibility for implementation, aligning with the proof-of-concept objectives.

By integrating the ML model and the KG in a bidirectional manner, this proof-of-concept demonstrates the potential to create a dynamically improving system. The KG benefits from the ML model's ability to identify patterns and gaps, while the ML model gains a richer, more informed training set, resulting in better HS code predictions. This integration highlights the feasibility and value of combining machine learning with structured knowledge systems in practical applications.

3.6 Controlled Evaluation for AI Risk Exposure

This section outlines the methodology for controlled evaluation of potential risks associated with the integration of artificial intelligence (AI) in the HS code classification system. The evaluation focuses on demonstrating how human-in-the-loop (HITL) interaction and the integration of the Knowledge Graph (KG), Machine Learning (ML), and Large Language Models (LLMs) ensure transparency and explainability in the proof-of-concept.

1. Risk Categories

- 1) Accuracy risks: Misclassification of HS codes leading to potential errors and regulatory consequences.
- 2) Bias risks: Imbalances in training data causing disproportionate outcomes in HS code assignments.
- 3) Explainability risks: Lack of clear reasoning behind predictions reducing user trust and acceptance.

2. Controlled Evaluation Methodology

- 1) Human-in-the-loop interaction:
- 2) Human experts validate AI predictions, focusing on edge cases and potential misclassifications.

- 3) Feedback loops ensure that corrections are incorporated into the ML model and updates are reflected in the KG.
3. Explainability through KG and LLM:
- 1) Predictions are contextualized using the KG, which provides a structured representation of part relationships, categories, and attributes.
 - 2) LLMs, integrated with the KG, generate natural language explanations that detail the decision-making process for each HS code assignment.
 - 3) Example: "HS Code 8429 was assigned because the part is classified as a hydraulic component with steel construction and matches regulatory criteria for this category."
4. Risk Mitigation Strategies
- 1) Human oversight: Experts regularly review and validate AI outputs, particularly for critical or high-stakes classifications. HITL ensures continuous monitoring and refinement of AI performance.
 - 2) Transparency and explainability:
 - The KG provides a clear structure of part attributes and their relationships, enabling traceable and interpretable predictions.
 - LLMs enhance transparency by generating detailed, user-friendly explanations for AI decisions.
 - 3) Iterative improvement: The ML model is retrained with enriched datasets informed by human feedback and KG updates. Continuous learning ensures that the system adapts to new data and regulatory changes.

By integrating the ML model, KG, and LLMs with human-in-the-loop processes, this proof-of-concept demonstrates a robust approach to managing AI risks. The combined system ensures accurate, fair, and explainable HS code predictions, building trust and reliability in the AI-powered classification system.

3.7 System Deployment and User Interaction via Console-Based Chatbot

This section outlines the deployment of the integrated LangChain-Knowledge Graph (KG) system and its user interface through a console-based chatbot, emphasizing its

role as a proof-of-concept. The focus is on demonstrating the feasibility and potential of the system to provide an interactive and efficient medium for users to query and provide feedback while ensuring seamless integration with the backend system. The system leverages LangChain and LLama 3.0 for advanced natural language understanding and generation, enhancing the chatbot's ability to interact with users effectively.

3.7.1 Console-Based Chatbot Interface

The console-based chatbot serves as the primary user interface during the proof-of-concept phase. It allows users to input part descriptions or product information and receive HS code recommendations interactively. The chatbot supports clarifications and follow-up queries by dynamically managing conversational prompts through LangChain.

Developed as a command-line application, the chatbot utilizes Python libraries like `cmd` or `argparse` to structure user interactions. It communicates with the LangChain-KG system via APIs to process user inputs and retrieve predictions and explanations. LangChain integrates with LLama 3.0, enhancing natural language understanding and enabling context-aware responses. Through LangChain's query generation modules, natural language queries are seamlessly converted into structured commands to interact with the KG.

3.7.2 Real-Time Query and Explanation

The system processes user queries by utilizing the LangChain pipeline, which connects the ML model to the KG. HS code predictions are generated and contextualized with data from the KG to ensure accuracy and traceability. Each prediction is accompanied by transparent explanations that reference factual relationships and attributes in the KG. For example, "HS Code 8429 was assigned because the part is made of steel and used in hydraulic systems." LangChain's multi-hop reasoning capabilities ensure comprehensive data retrieval, while LLama 3.0 provides detailed and user-friendly explanations.

3.7.3 User Feedback Integration and KG Updates

The chatbot interface also acts as a feedback collection mechanism, allowing users to review and validate predictions or provide corrections. Feedback is logged and analyzed within the LangChain framework to refine system performance. Validated feedback is translated into machine-readable commands for KG updates through LangChain's custom chain modules. LLama 3.0 ensures that natural language feedback is accurately interpreted and grounded into structured updates for the KG. As a result, relationships, nodes, and edges in the KG are dynamically adjusted, keeping the data relevant and up-to-date.

By deploying the LangChain-KG system with a console-based chatbot interface and leveraging LLama 3.2, this proof-of-concept combines simplicity with advanced interaction capabilities. The system's ability to explain predictions using KG-based factual data ensures transparency and fosters user trust. Furthermore, LangChain's management of conversational context and KG updates establishes a dynamic feedback loop, enabling continuous refinement and adaptability.

3.8 Evaluation Method

The evaluation method combines quantitative and qualitative approaches to assess the proof-of-concept's effectiveness in HS code classification. As a proof-of-concept, the focus is on demonstrating the feasibility and potential of the proposed system while validating its performance, usability, and compliance with regulatory requirements.

3.8.1 Quantitative and Qualitative Evaluation.

1. Quantitative Evaluation

- 1) Machine Learning Model Evaluation assesses the ML model's performance in predicting HS codes.
- 2) Metrics:
 - Accuracy: Proportion of correct predictions.
 - Precision: Ratio of true positive predictions among all positive predictions.
 - Recall: Ratio of true positive predictions among actual positives.

- F1-Score: Harmonic mean of precision and recall.

3) Tools: Scikit-learn for calculating evaluation metrics and visualizations (e.g., confusion matrices).

2. Qualitative Evaluation

1) System Usability Testing:

- Evaluates the usability of the system interface and functionality.
- Method: Observational studies where users perform specific tasks using the system.
- Data Collection: Task completion time and user feedback through surveys or interviews.

2) Controlled Use Cases:

- Tests the system against real-world scenarios and edge cases.
- Method: Develop scenarios simulating ambiguous part descriptions or parts with similar descriptions but different uses.
- Outcome: Assess classification accuracy and robustness in handling complex scenarios.

3. Success Criteria

The success of the proof-of-concept is determined by the following indicators:

- 1) High Classification Accuracy: Achieves above 90% accuracy for simple cases and above 80% for complex cases.
- 2) Explainability and Transparency: Provides clear and understandable justifications for classifications, promoting trust and regulatory compliance.
- 3) Regulatory Compliance: Ensures all classifications align with BTKI and relevant regulations.
- 4) User Acceptance and Usability: Users find the system intuitive and helpful, indicating a positive experience and adoption.
- 5) Adaptability to New Data: Incorporates new rules, part numbers, and materials seamlessly without major system retraining or disruptions.

This evaluation method ensures a focused and balanced assessment of the proof-of-concept's ability to meet HS code classification challenges. By concentrating on essential metrics and scenarios, the evaluation validates the feasibility and potential of the solution while highlighting its accuracy, usability, and compliance.

3.8.2 Evaluation Using the RAGAS Framework

As part of this proof-of-concept study, the Retrieval-Augmented Generation Assessment (RAGAS) framework is employed to evaluate the system's performance, aligning with the overarching goal of exploring the feasibility and iterative potential of the proposed evaluation system within a controlled and experimental setting. This approach uses a 0 to 1 scoring scale for each metric: Faithfulness, Answer Relevance, Context Relevance, and Context Recall. The proof-of-concept goal is to demonstrate the feasibility and potential effectiveness of the framework in assessing retrieval-augmented generation systems. Higher scores indicate better alignment with desired outcomes, such as precise retrieval, relevant and faithful outputs, and comprehensive context coverage.

Metrics and Scoring Process:

1. Faithfulness:

Measures factual consistency of the generated output with the retrieved context.

1) Scoring Process:

- Decompose the output into individual statements.
- Use an automated Large Language Model (LLM) with predefined prompts to validate whether each statement is supported by the retrieved context.
- The score is calculated as:

$$\text{Faithfulness Score} = \frac{\text{Number of Supported Statements}}{\text{Total Statements in Output}}$$

- ##### 2) Interpretation:
- A score close to 1 indicates high factual grounding, while lower scores highlight hallucinations or unsupported information.

2. Answer Relevance:

Evaluates the semantic alignment between the user's query and the generated response.

1) Scoring Process

- LLMs or embedding-based models generate alternative queries from the output.
- Calculate semantic similarity between the original query and these alternatives using vector embeddings (e.g., cosine similarity).
- The average similarity score represents the Answer Relevance score.

2) Interpretation: A score closer to 1 indicates high relevance of the generated response to the query.

3. Context Relevance

Assesses the proportion of meaningful and task-relevant information in the retrieved context.

1) Scoring Process:

- Analyze the retrieved context for focus and relevance.
- Identify and penalize irrelevant or extraneous content.
- The score is calculated as:

$$\text{Context Relevance Score} = \frac{\text{Relevant Information}}{\text{Total Retrieved Content}}$$

2) Interpretation:

- Higher scores indicate concise and relevant retrieval, essential for accurate generation.

4. Context Recall

Evaluates the completeness of the retrieved context with respect to the query.

1) Scoring Process:

- Identify a reference or benchmark set of ideal retrieved passages using heuristics or expert judgments.
- Compute the overlap between the system's retrieved context and the ideal set.
- The score is calculated as:

$$\text{Context Relevance Score} = \frac{\text{Relevant Information}}{\text{Total Relevant Information Expected}}$$

2) Interpretation:

- Higher scores ensure no critical information is missed, which is crucial for reliable classification.

Success Criteria:

- Scores Near 1: Indicate that the system performs exceptionally well, retrieving relevant context, ensuring completeness, and generating accurate and grounded outputs.
- Scores Below 0.5: Signal issues in the retrieval or generation process, such as incomplete retrieval, irrelevant information, or hallucinated responses. These areas require further refinement.

3.9 Validation Method Focus Group Discussion Against Experts

3.9.1 Expert Criteria

The system's HS code classifications will be compared against the classifications made by a group of certified experts. Focus Group Discussions (FGDs) with domain experts will guide the selection of controlled use cases to ensure that the AI system reflects the actual needs and challenges of the industry. These discussions will help identify potential AI risk exposures and suggest ways to integrate human-in-the-loop mechanisms. Experts invited to these FGDs will meet the following criteria:

1. **Certifications:** Participants must hold relevant certifications, such as Customs Expert Certification (*Sertifikasi Ahli Kepabeanan* or SAK) is a certificate awarded by the Directorate General of Customs and Excise (*Direktorat Jenderal Bea dan Cukai* - DJBC) to individuals who pass the customs certification exam, signifying their knowledge and expertise in the field of customs presented in figure 3.2. Additionally, holders of the Customs Legal Power of Attorney (*Izin Kuasa Hukum Kepabeanan*) are authorized to represent parties in customs-related disputes during proceedings in the Tax Court.
2. **Experience:** Have at least 3 years of experience in customs clearance, supply chain management, or heavy equipment parts classification.

3. **Domain Knowledge:** Experts should have a deep understanding of heavy equipment spare parts, their functions, and classification nuances.
4. **Openness to Technology:** Participants should be receptive to AI-driven solutions and able to provide feedback on system usability and practicality.
5. **Decision-Making Authority:** Ideally, participants should be involved in official classification decisions or have a role in the compliance department of PT XYZ (or a similar organization). This helps ensure that feedback and insights from the FGD reflect the perspectives of actual decision-makers.



Figure 3.2 The Customs Expert Certification (Sertifikasi Ahli Kepabeanaan or SAK).

3.9.2 Questionnaire Design Using the PMS Framework

The Process Modelling Success (PMS) framework, introduced by Sedera et al. (2002), provides a structured approach to evaluate the integration of Machine Learning (ML), Large Language Models (LLMs), and Knowledge Graphs (KGs) in automating HS code classification. The framework evaluates the system across five critical dimensions:

1. **Modeler Satisfaction:** Measures user satisfaction with the tools and resources utilized in the system, particularly LLM and KG integration.
2. **Process Model Quality:** Focuses on the accuracy, clarity, and compliance of the system with BTKI regulatory standards.

3. **Model Use:** Assesses usability, adaptability to regulatory changes, and its role in reducing manual classification efforts.
4. **Process Impact:** Evaluates improvements in workflows, error reduction, and alignment with operational goals.
5. **Organizational Impact:** Analyzes strategic benefits, such as cost savings and enhanced decision-making.

To gather comprehensive feedback, a structured questionnaire was developed using Likert-scale and open-ended questions. The questionnaire addressed the following themes:

1. **Accuracy and Clarity:** Measuring the system's precision in classifications and the logical presentation of results.
2. **Handling Complex Scenarios:** Evaluating performance in ambiguous or incomplete cases.
3. **Usability:** Reviewing the intuitiveness and navigability of the interface.
4. **Reliability and Scalability:** Assessing consistency across datasets and readiness for scaling.
5. **Future Enhancements:** Identifying features to improve functionality and user trust.

Sample Questionnaire Questions:

1. **Likert-Scale Questions (1: Strongly Disagree to 5: Strongly Agree):**
 - 1) "The system demonstrates accurate HS code classifications."
 - 2) "The explanations for classifications are clear and logically presented."
 - 3) "The system effectively handles ambiguous or incomplete product descriptions."
 - 4) "The classification process is more efficient compared to manual methods."
 - 5) "The outputs are consistent across similar cases."
 - 6) "The system's interface is intuitive and user-friendly."
 - 7) "The system aligns with organizational goals for regulatory compliance."
 - 8) "The system shows potential to reduce classification errors and improve operational efficiency."

2. Open-Ended Questions:

- 1) "What do you think are the system's main strengths?"
- 2) "Are there any aspects of the system that you find unclear or would like to see improved?"
- 3) "What features or improvements would make the interface more intuitive for you?"
- 4) "In what ways do you see the system contributing to cost savings or operational efficiency?"
- 5) "How well does the system align with your expectations for classification tasks?"

3.9.3 Focus Group Discussions (FGDs)

The FGDs were designed to create an interactive environment where domain experts could evaluate and discuss the HS code classification system. The FGD process was structured into the following steps:

1. **Participant Selection:** Experts with relevant certifications (e.g., Certified of Pabean Expert) and professional experience in customs or classification were invited. Participants were selected based on their familiarity with BTKI regulations and openness to technology-driven solutions.
2. **Preparation:** Before the discussions, participants were provided with briefing materials, including an overview of the system's objectives, functionalities, and expected outcomes.
3. **Session Flow:**
 - 1) **Introduction:** Facilitators outlined the purpose of the FGD and introduced the PMS framework as the evaluation model.
 - 2) **System Demonstration:** A live walkthrough of the system highlighted key features, including ML-based classifications, LLM-driven justifications, and KG integration for handling complex queries.

- 3) **Guided Evaluation:** Participants assessed the system's performance through structured discussions based on the five PMS dimensions. Facilitators posed targeted questions to elicit feedback on usability, accuracy, and scalability.
 - 4) **Open Dialogue:** Time was allocated for participants to share additional observations, challenges, and recommendations without constraints.
4. **Data Collection:** Responses were recorded and categorized for analysis. Structured feedback forms were distributed to ensure all participants could provide their input comprehensively.
 5. **Post-FGD Analysis:** The collected data was synthesized to identify recurring themes, actionable suggestions, and areas requiring further exploration or refinement.

The FGDs, guided by the PMS framework, provided a systematic approach to evaluate the HS code classification system. By engaging domain experts in structured discussions and leveraging their feedback, the process ensured a thorough assessment of the system's design, functionality, and practical applicability. This iterative approach will inform future development phases to align the system with user needs and regulatory requirements.

3.10 Conclusion

In Chapter 3, the research methodologies and framework for developing an integrated HS code classification system using Machine Learning (ML), Knowledge Graphs (KGs), and Large Language Models (LLMs) were thoroughly outlined. The chapter detailed the multi-phase approach, including data preparation, model development, KG construction, ML-KG integration, and risk evaluation, culminating in a proof-of-concept deployment via a console-based chatbot interface. This structured methodology provided a foundation for achieving accurate, explainable, and scalable HS code predictions.

Chapter 4 builds upon this framework, presenting the results and analysis derived from implementing the proposed methodology. It evaluates the effectiveness of the system in addressing the challenges and gaps identified earlier, providing insights into its performance, scalability, and feasibility. The chapter also discusses the implications of the findings in the context of HS code classification and suggests potential areas for further optimization and research.

