Detecting Information Codes From Data Module Codes

Data Science Final Project Report

112ZU1021 洪勻馨、112ZU1048 陳雨柔

Background

The Ship and Ocean Industries R&D Center (SOIC) hopes that we can assist them in organizing technical document examples and developing related tools based on the S1000D standard. The S1000D standard is a widely recognized framework for creating, managing, and delivering technical documentation, particularly in aerospace, defense, and marine industries.

First, we analyzed the Common Source Database (CSDB) within the standard to understand its coding logic and significance. Subsequently, we applied this standard to ship operation and maintenance processes to evaluate the efficiency and compliance of document drafting.

A key component of S1000D is the Data Module Code (DMC), which serves as a structured identifier for technical information. The DMC encodes the type, purpose, and context of a data module, allowing users to efficiently locate, retrieve, and update specific content. This standardized coding system is essential for organizing complex technical data and ensuring seamless integration within the Common Source Database (CSDB).
We have developed a query system based on the S1000D BIKE manual that allows maintenance personnel to ask questions about the difficulties they encounter. The system identifies the corresponding DMC based on their query, locates the relevant data module associated with the DMC, and finally uses an Large Language Model (LLM) to present the content of the manual in a way that is easy for humans to understand.
However, for SOIC's documents to be searchable and iqueryable in the same way, it means that these documents must strictly adhere to the S1000D standard, particularly the DMC coding rules.
The DMC is divided into three main sections: Hardware/System Identification, Information Type, and Learn Type. The first part identifies the specific piece of hardware or part of the system, while the second part specifies the type of information contained within the data module. Since the first part varies depending on the subject of the technical publications, we have decided to focus on studying the second part, Information Type, in the absence of SOIC's documents.

Problem Definition

According to the S1000D 6.0 specification PDF, there are a total of 1,145 Information Codes covering a wide range of topics (e.g., function, operation, servicing, fault reports). These codes categorize technical information into various functional areas. Given this, we aim to leverage machine learning by using XML files with correctly labeled Information Codes as a dataset to train an automated classification tool.

Data Collection

Our dataset contains four parts-the S1000D 6.0 Manual pages 3498-3546 (574 items), S1000D Bike Data Set for Release number 6 R2 (116 items), ‘s1kd_tools’(46 items) and ‘fossig’(12 items). We directly downloaded the first two (the first being the entire S1000D 6.0 Manual) from the official S1000D website and the other two from Github. Since data was still inadequate for creating a model that could identify the full information code for these files, we decided to focus on identifying the first digit (the primary code) instead. This means the model should be able to identify large categories, for example: ‘Disconnect, remove and disassemble procedures’.

Table 2 Primary codes

Primary code	Definition
000	Function, data for plans and description
100	Operation
200	Servicing
300	Examinations, tests and checks
400	Fault reports and isolation procedures
500	Disconnect, remove and disassemble procedures
600	Repairs and locally make procedures and data
700	Assemble, install and connect procedures
800	Package, handling, storage and transportation
900	Miscellaneous
C00	Computer systems, software and data

The full definitions are given in Table 3, Table 4, Table 5, Table 6, Table 7, Table 8, Table 9, Table 10, Table 11, Table 12, and Table 13.

Processing

For this project, we needed to extract information codes from each file’s DMC to label and categorize the data. For the S1000D 6.0 Manual, we only had access to a PDF version, which we first converted into a CSV file for processing. Due to structural differences in the tables containing the primary and secondary codes, we separated the two datasets and manually input the definitions for the 11 primary code labels (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, C). Using LabelEncoder, we converted the textual labels into numerical format, assigning ‘C’ to label 10.

Given that this dataset was the only one with balanced data across all 11 labels, we aimed to amplify its significance by resampling the data frame with frac=5 (increasing the dataset size by five times) and using replacement (replace=True). A factor of 5 was chosen as it provides a reasonable increase in data size without excessive duplication, which could lead to overfitting. This balance helps the model generalize better while avoiding the risk of biasing it towards overly repeated instances.As for the rest of the CSV file, we cleaned the data by removing irrelevant rows and columns, retaining only the title column for each secondary code to streamline the text. To ensure consistency, we also standardized the information codes to three digits, allowing the model to recognize numbers such as ‘14’ and ‘21’ as codes starting with ‘0’. Once the dataset was cleaned, we extracted the first digit of each information code as the ‘label’ and its corresponding description as the ‘text’.

For the BIKE, s1kd_tools, and fossig datasets, we worked with folders containing XML files and other related data. We extracted all XML files, retrieved the first digit of each information code under the <infoCode> tag, and their corresponding descriptions from the <infoName> tag. As the XML structures were consistent across files, we were able to combine them into a single data frame.

After compiling the data, we appended and shuffled the entries. The final dataset contained 792 records, with 80% allocated for training and the remaining 20% used for testing.

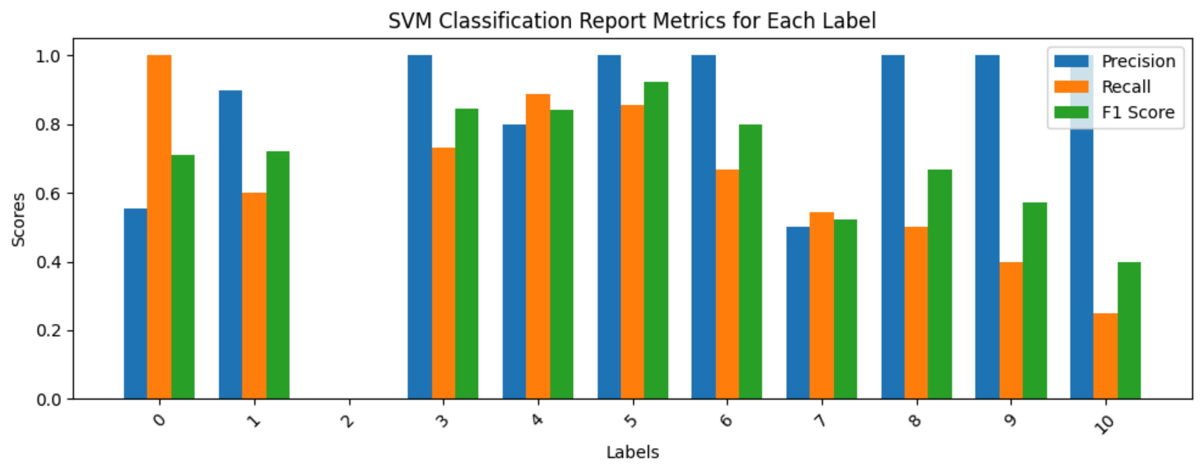
Black Box Design and Evaluation Metrics

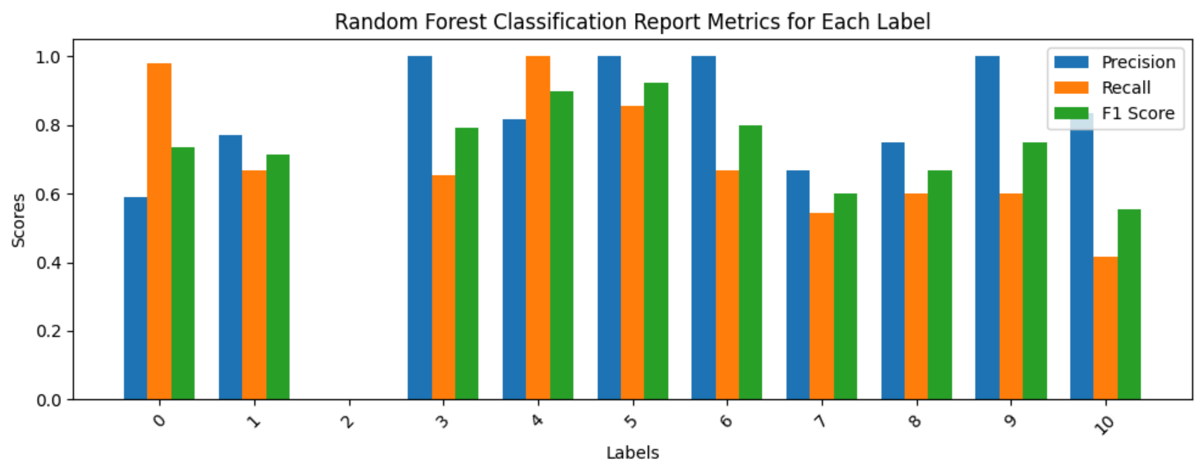
The model training process begins with raw text data and information codes as inputs. The data is then split, cleaned, and encoded. For feature extraction, the text is transformed into numerical vectors using the TF-IDF method. The machine learning model is trained using the preprocessed data, learning the relationship between text descriptions and information codes without manual input. After training, the model is evaluated on a testing set, using accuracy, precision, recall, and F1 score as evaluation metrics to assess its performance.

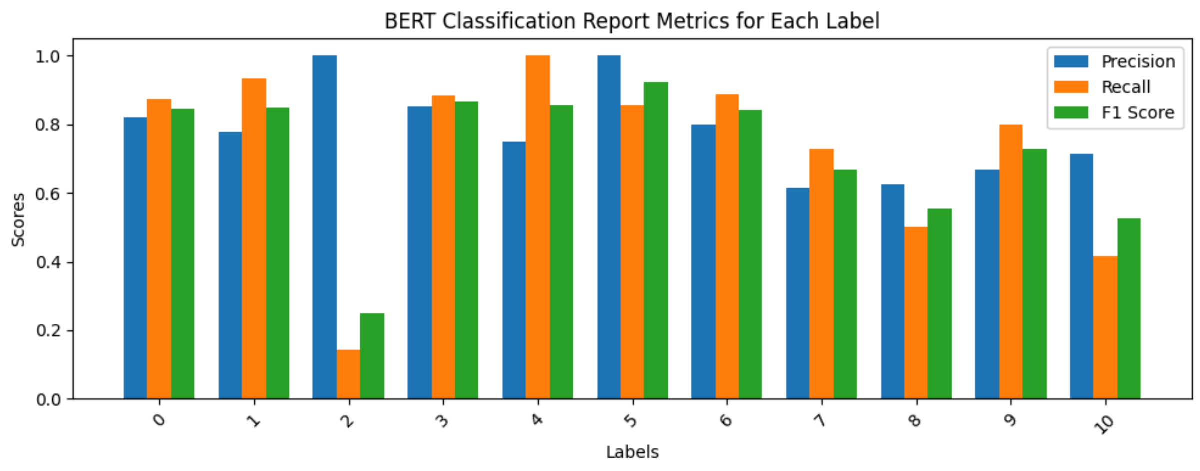
These metrics were chosen considering accuracy gives an overall sense of how well the model is doing in general and is simple to understand. However, it doesn’t give class-specific insight, only the overall performance, so it could be misleading for our model, considering our dataset is somewhat imbalanced. Precision minimizes false positives but ignores false negatives. Optimizing for precision may lead to ignoring less frequent but important classes. Recall does the opposite- it minimizes false negatives but doesn’t consider false positives. Optimizing recall could lead to a model that classifies almost everything as positive. Lastly, F1 Score balances precision and recall, making it useful for our task because both false positives and false negatives are important. This may be a better metric for our model as it works well for imbalanced datasets by taking the previous two metrics into account. However, the F1 score doesn’t provide a comprehensive understanding of a model’s performance across all possible classes. It is only a summary metric for binary classification or averaged for multi-class problems.

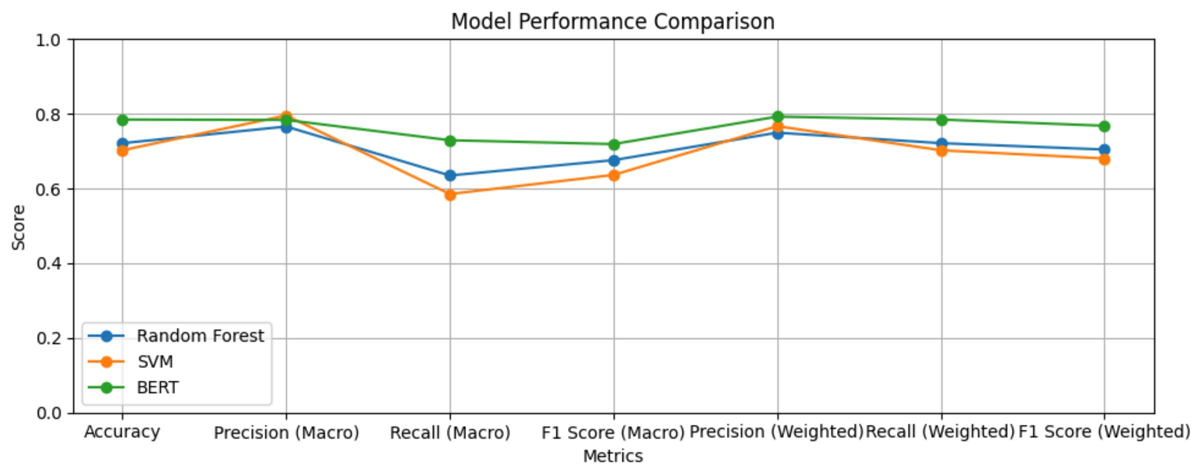
The models used in this process include Support Vector Machine (SVM), Random Forest (RF), and BERT, each chosen for their ability to handle classification tasks effectively. SVM was selected for its robustness in high-dimensional spaces (useful for text classification and featured data) and its ability to find an optimal decision boundary between classes. It can also achieve good accuracy when the dataset is not too large which suits our needs. Random Forest was chosen for its ensemble learning capabilities, which reduce overfitting and improve accuracy, and its strength in handling both numerical and categorical data. Since RF uses bootstrapped datasets to create different trees and averages their results, it is generally more robust to noisy data than a single decision tree. Although RF is used for large datasets, we wanted to see how it would perform compared to SVM on our limited dataset. Finally, BERT(a transformer-based model), was included for its deep contextual understanding of language, making it especially effective at capturing semantic relationships in text. By employing these three models, the classification task benefits from a comprehensive approach that combines the advantages of traditional machine learning methods and natural language processing techniques. Finally, the model generates predictions of information codes for unseen data, providing outputs based on the learned patterns without requiring the user to understand the complex decision-making process behind them.

Visualization of Results









Conclusion

In conclusion, BERT outperformed SVM and Random Forest in terms of overall performance, with the highest accuracy (0.78), precision (0.79), recall (0.78), and F1 score (0.77). This suggests that BERT's deep learning-based architecture is particularly well-suited for handling complex text classification tasks and is better at capturing the intricate semantic relationships within the text, making it the most effective for identifying DMC codes.

On the other hand, SVM performed the worst overall, with the lowest accuracy (0.70), recall (0.70), and F1 score (0.68). Despite having relatively high precision (0.77), SVM struggled to balance precision and recall, especially with label 2, which it failed to identify altogether. Its decision boundary was unable to generalize well across all classes, contributing to its lower performance compared to BERT and Random Forest.

The main challenge in classification arose with the underrepresented labels, particularly label 2, which all models struggled to classify effectively. The small number of samples for label 2 (only 7 samples in the test dataset) contributed to its poor performance across all models. BERT, although providing high precision for label 2, had very low recall, indicating it missed many instances of this class. This imbalance in the dataset, with significantly more samples for label 0 (47 samples), made it difficult for the models to learn effective patterns for the minority classes. Interestingly, despite having few samples, labels 4, 5, and 9 were better identified, which suggests that other factors, such as more distinct patterns in these labels, may have contributed to their better performance.

Complexity in the language of DMCs, which are tailored to the S1000D setting, likely posed additional challenges for traditional models like SVM and Random Forest. These models struggled to capture deeper semantic meaning, while BERT's advanced architecture excelled in understanding the nuanced context. Feature representation also played a role, as SVM and Random Forest relied on vectorized representations that did not fully capture word dependencies, while BERT's deep learning approach was better equipped to model these relationships.

Outliers and potential mislabeled data may have further influenced the models' performance, especially in cases where the data was not completely clean. While we made efforts to eliminate irrelevant data, noise in the dataset might still have affected the models' predictions, particularly for SVM and Random Forest.

Finally, using both macro and weighted averages provided a nuanced view of the models' performance. The macro average treated all classes equally, offering a balanced assessment, while the weighted average reflected the models' overall performance, emphasizing classes with better results. The choice between these metrics depends on whether the goal is to evaluate fairness across all classes or focus on the overall performance.

Future Improvements

To address the data imbalance issue, we plan to increase the size and diversity of the dataset by incorporating additional DMC datasets from SOIC. By adding more data from a variety of sources, we can ensure that each class has enough representative samples, allowing the model to learn more effectively. This will also help mitigate the risk of overfitting to the more dominant classes and provide a more balanced distribution of information for the model to learn from.

Furthermore, we may explore techniques like data augmentation, where we artificially create more data by manipulating existing samples (e.g., by paraphrasing or adding noise to the text). This can supplement the dataset, especially for minority classes, without the need to rely solely on external data sources.

In addition to increasing the dataset size, we could also experiment with advanced techniques such as active learning. Active learning would allow the model to selectively request labels for the most uncertain samples, focusing the labeling process on data that is most likely to improve model performance.

Moreover, tuning model parameters such as learning rates, regularization, and batch sizes, along with experimenting with different model structures, can further optimize performance. By combining these strategies—data augmentation, dataset expansion, and hyperparameter tuning—we aim to significantly improve our model's ability to classify all classes accurately and efficiently, even those with fewer samples.

Self Assessment

112ZU1021 洪勻馨

This project involved exploring and implementing several machine learning models, including SVM, Random Forest, and BERT, to classify information codes from DMCs. Through this process, I gained a deeper understanding of each model's strengths and weaknesses, particularly in handling imbalanced datasets and complex text classification tasks. By analyzing metrics such as accuracy, precision, recall, and F1 scores, I was able to identify challenges that impacted performance, including underrepresented classes and the complexity of the data.

Initially, I was skeptical about our chosen topic. While I knew it was something we would need to tackle eventually, the limited datasets and inconsistent data formats made me worry that training the models in a short amount of time would be extremely difficult. Data cleaning and attempts to improve overall performance proved to be the most challenging parts of this project. Fortunately, thanks to previous experience in extracting content from XML files, we were able to efficiently target the necessary tags, simplifying the data retrieval process for XML files. However, working with the manual required converting it into a CSV file to sort and label the content effectively. During this process, we encountered columns containing entries like “137 thru 140 = Not available for all projects,” which we decided to delete to prevent interference with text classification, but also lessened the total amount of data we had.

Despite the initial difficulties, the results for our models exceeded my expectations. While the overall performance is not outstanding, it showed significant improvement compared to our early attempts when accuracy hovered around 0.45–0.6 across multiple models. To address data imbalance, we experimented with resampling, weighting, and SMOTE; among these, only resampling showed noticeable improvement.

Even so, there remain unresolved issues. Both SVM and Random Forest consistently failed to identify label 2, and BERT only succeeded in doing so after retraining. It’s unclear whether this is due to the wording patterns being particularly difficult to trace or an error in our code. This ambiguity highlights the need for further investigation into both the dataset and model behavior.

Looking back, this project has significantly enhanced my technical skills and critical thinking in machine learning. It has also underscored the importance of balancing model complexity, data quality, and computational resources. While there are still areas for improvement, this experience has provided valuable insights into the challenges of real-world machine learning applications and the steps needed to overcome them.

112zu1048 陳雨柔

As Cindy mentioned, we encountered issues such as data imbalance, insufficient data, and low accuracy. Initially, I approached the problem by thinking about the story classifier example the teacher provided in class. I decided to use the DMCs of BIKE, Fossig, and s1kd along with the text within the <content></content> tags in their XML files as labels and text. However, I later realized that these three examples had significant differences in content, and their quantities were highly imbalanced.

	BIKE	Fossig	s1kd
xml	116	12	46

After consulting with the teacher, we discovered that the definitions of DMCs in the PDF could also be incorporated into the database. By replacing <content> with <infoName> and using the primary code of the DMC as the label instead, we increased the amount of data. This adjustment reduced the need for the model to classify the more granular secondary codes.

When evaluating model performance, the results varied significantly each time. Accuracy sometimes dropped to around 3, but after running it a few more times, it could reach as high as 7. This made me feel like the training process was a black box. We was also concerned that adjusting the frac parameter to increase the amount of specification data might have caused duplicates to appear in both the training and testing sets, which could lead to the model "cheating." Later, I realized there are many ways to address data imbalance. For example, we could ensure that duplicate data only appears in one set before splitting the dataset, generate more diverse samples instead of simply duplicating existing ones, or use cross-validation to avoid relying on a single train-test split. We plan to try these methods in the future to see if they improve performance.

Because the results varied with each run, the ranking of SVM, Random Forest, and BERT in terms of performance also kept changing, making interpretation more challenging. BERT, with its word embeddings and contextual understanding, should theoretically perform better, but sometimes its accuracy was the worst. I suspect that the improvements in accuracy during repeated training might have been due to data leakage or randomness misleading the results. It seems that improving data quality—by increasing both the diversity and quantity of data—is the only real solution to these issues.

Division of Tasks

Data Collection	洪勻馨、陳雨柔
Data Processing	洪勻馨
Black Box Design and Evaluation Metrics	洪勻馨、陳雨柔
Visualization of Results	洪勻馨
Conclusion	洪勻馨
Slides	陳雨柔
Report	洪勻馨、陳雨柔