Data Classification for Improved Cost Analysis

Authors

Utkarsh Raj Sutihar	1008039
Sim Puey Kun (Shen Peijun)	1007996
Brosman Lim Hao Xiang	1007812
Jesco Tan Jiong Rui	1008137
Tan Zhe Xuan	1007984

Table of Contents

Executive Summary

This report details the methodology and findings of an analysis conducted on approximately 1.2 million material description records provided by SLB, covering operations in Saudi Arabia, Mexico, and United States (Land). The primary objective was to categorise these descriptions, particularly identifying the top 100 most frequent categories, to provide SLB with actionable data for potential financial decisions regarding procurement strategies, inventory management, and supplier relationships.

The analysis employed a multi-step methodology combining data preprocessing, natural language processing (NLP), and machine learning techniques. Initial steps involved consolidating data from regional Excel files, identifying and pre-categorising specific item types ('O-RING', 'PartNumber', 'empty'), and cleaning inconsistent text descriptions. A key challenge addressed was the presence of multiple languages, necessitating translation efforts. Subsequently, the cleaned descriptions were converted into numerical representations (embeddings) using the 'all-MiniLM-L6-v2' Sentence Transformer to capture semantic meaning. These embeddings were then grouped using K-Means clustering to identify 100 distinct clusters. Finally, these clusters were manually reviewed and assigned meaningful category names, allowing for frequency analysis across the total dataset and by region.

Key findings highlight the significant prevalence of the 'ORING' category, appearing over 126,000 times, more than double the frequency of the second-highest category ('empty'), representing data that SLB withheld from us due to certain business confidentiality. Regional analysis revealed substantially higher O-ring usage in KSA and MCA compared to USL. Other prominent categories include general miscellaneous items, maintenance and fluid control components, valves, and mechanical fittings. A heatmap visualisation was also generated, illustrating the overall frequency distribution across categories.

The project faced challenges: the large data size, and the complexity introduced by multiple languages. Limitations of this analysis include reliance on translation accuracy, the absence of financial cost data which prevents direct calculation of financial impact, potential miscategorisation due to incomplete operational context, and the inherent need for some ongoing manual oversight for category refinement.

The primary recommendation is for SLB to replicate this analytical methodology using their internal dataset that **includes financial cost information**. This will enable a true understanding of high-expenditure categories and provide a robust foundation for data-driven financial decisions.

1 Introduction

SLB operates globally in the oilfield services sector, managing vast inventories and complex supply chains across numerous regions. Efficient management relies on accurate data analysis, particularly understanding the nature and frequency of procured materials and services. This project was initiated to address the challenge of categorising a large volume of material descriptions – approximately 1.2 million rows – originating from SLB's operations in the Kingdom of Saudi Arabia (KSA), Middle East and Central Asia (MCA), and United States Land (USL) regions.

The core objective was to process and sort through these descriptions, often inconsistent or unstructured, to identify and quantify the top 100 distinct categories of items or services represented. This categorisation aims to provide SLB, particularly its finance and procurement departments, with clearer visibility into its operational needs and consumption patterns. The insights derived are intended to support informed financial decision-making, potentially leading to optimised procurement strategies, reduced inventory costs, or changes in service supply arrangements.

The source data was provided as Excel files containing various fields, with the primary focus being the 'Material: Description' and associated 'Text' fields, which contained free-text descriptions of parts, materials, and services (e.g., "O-Ring", "Machining services", "Oil"). This report details the methodology employed, presents the key findings from the analysis, discusses the inherent limitations, and provides recommendations for SLB moving forward.

2 Methodology

A systematic, multi-stage approach was adopted to process the large and complex dataset, leveraging data processing techniques, Natural Language Processing (NLP), and machine learning clustering. The process, implemented via a series of Python scripts is detailed below.

2.1 Data Loading & Preprocessing (`step_1_preprocessing.py`)

Raw data from KSA, MCA, and USL Excel files were consolidated. A combined field, 'MT' (Material + text), was created using 'Material: Description' and, if missing, the 'Text' field. Each row was assigned a unique 'SerLoc' identifier (region + serial number). Rule-based categorisation was applied: 10-digit part numbers starting with '1' were labelled 'PartNumber', O-RING variations as 'ORING', and empty fields as 'empty'. All other rows remained uncategorised for further analysis.

2.2 Text Cleaning & Translation (`step_2_cleaning.py`)

Uncategorised 'MT' descriptions were standardised by converting to lowercase, removing symbols, digits, short words, and excess whitespace. This produced a cleaned text field for each item, ready for semantic analysis.

2.3 Feature Extraction (BERT Embeddings) (`step_3_BERT.py`)

Cleaned descriptions were converted into 384-dimensional vectors using the 'all-MiniLM-L6-v2' Sentence Transformer model. This model captures semantic meaning and was run with multiprocessing and GPU acceleration for efficiency. The resulting embeddings were stored with the original data.

2.4 Clustering (K-Means) (`step_4_k-means.py`)

K-Means clustering grouped the BERT embeddings into 100 clusters, minimising intra-cluster distance and maximising inter-cluster distance. Each item was assigned a cluster label (0-99).

2.5 Categorisation & Output Generation (`step_5_output.py`)

Clusters were mapped to business-relevant category names via manual review. Items previously categorised as 'PartNumber', 'ORING', or 'empty' retained their labels. Frequency statistics were computed for each category and region, and compiled into a summary table. The data was split into different regions while retaining the original structure by using the 'SerLoc' field and masking in all steps.

2.6 Challenges Encountered

Several challenges were faced during this project:

- **Data Delivery Delay:** The project timeline was impacted by a five-week delay in receiving the initial dataset.
- **Data Volume:** Processing approximately 1.2 million rows required efficient methods and consideration of computational resources.
- **Multi-language Data:** The presence of multiple languages within the description fields necessitated translation efforts prior to cleaning and analysis, adding complexity and potential for inaccuracies.

3 Results & Discussion

The analysis successfully processed the ˜1.2 million material descriptions and categorised them into 100 distinct groups based on semantic similarity, supplemented by initial rule-based categories. The frequency distribution across these categories provides valuable insights into SLB's operational consumption patterns in the KSA, MCA, and USL regions.

3.1 Overview of Top Categories

The most striking finding is the dominance of the 'ORING' category, which was identified through rule-based pattern matching. This category accounts for 126,336 instances across the three regions, making it significantly more frequent than any other identified group. Its frequency is more than double that of the next largest category, 'empty' (65,995 instances), which typically represents records lacking sufficient description data. The high prevalence of O-rings suggests they are a critical and high-volume component in SLB's operations within these regions.

Beyond O-rings and empty descriptions, several other categories emerged with high frequencies. The top 10 categories (excluding 'empty' for clarity on tangible items/services) derived from the provided statistics are:

Category	Total Frequency	KSA Frequency	MCA Frequency	USL Frequency
ORING	126,336	41,198	63,240	21,898
Miscellenous	59,263	12,541	18,122	28,600
Maintainance and fluid control	30,219	836	27,524	1,859
Valve	22,395	3,429	6,666	12,300
Mechanical Fittings	22,094	2,711	6,608	12,775
Retaining Ring	21,818	5,521	12,922	3,375
Spanish (Field Machining/Site Tool)	21,197	57	21,055	85
Seal	20,706	4,961	9,570	6,175
Screw Cap	20,096	5,268	8,191	6,637
Service item	19,874	1,229	10,801	7,844

(Table derived from category_statistics.xlsx - Sheet1.csv)

The presence of categories like 'Miscellaneous', 'Maintenance and fluid control', 'Valve', 'Mechanical Fittings', and 'Seal' within the top ranks highlights key areas of operational activity and material consumption. The appearance of Spanish-language categories (e.g., 'Spanish (Field Machining and Site Tool)') confirms the multi-language nature of the data and the outcome of the clustering process grouping these items.

3.2 Regional Variations

Analysis of category frequencies by region revealed notable differences in consumption patterns. The most significant variation observed was in the 'ORING' category. While prevalent in all three regions, the frequency was considerably higher in MCA (63,240 instances) and KSA (41,198 instances) compared to USL (21,898 instances). This disparity might reflect differences in operational focus, equipment types used, maintenance schedules, or reporting practices between the regions. Further investigation by SLB personnel with regional operational knowledge would be required to determine the precise reasons for this difference.

Other regional variations exist across different categories. For instance:

- 'Maintenance and fluid control' shows a very high frequency in MCA (27,524) compared to KSA (836) and USL (1,859).
- 'Valve' and 'Mechanical Fittings' appear most frequently in USL.
- Categories identified as predominantly Spanish language content, such as 'Spanish (Field Machining and Site Tool)', are almost exclusively found in the MCA region dataset provided.

These regional differences underscore the importance of localised analysis when making procurement or inventory decisions. A one-size-fits-all approach may not be optimal across SLB's diverse operational landscape. The generated statistics provide a starting point for SLB to explore these regional nuances further.

3.3 Data Visualisation

Alongside the tabular frequency data, a heatmap visualisation was generated (as referenced in the input discussion, typically part of the accompanying Excel analysis). This heatmap serves to provide an intuitive visual representation of category frequencies across the dataset, likely highlighting the dominance of the top categories (like

'ORING') and potentially illustrating the distribution patterns across KSA, MCA, and USL, complementing the insights discussed above. Such visualisations are valuable tools for quickly grasping the overall landscape of material consumption and identifying areas that warrant closer inspection.

A PCA projection of the BERT embeddings reveals that the horizontal (x-axis) component separates data based on language. Descriptions on the left side of the plot are predominantly in Spanish (e.g., categories like "Spanish (Threading and Machine)", "mxw (mostly Spanish) - hand and power tool", "maquinar rosca (Spanish) - Threading", "Spanish (Piping and Mechanical Supplies)"), while those on the right are in English (e.g., "Bolt", "Collar", "Gasket", "Screw"). This demonstrates that the BERT model captures language differences in the embedding space. Notably, the PCA projection enables the clustering of similar words and groups of words, resulting in a visible language-based separation due to inherent linguistic differences among the data points. With further research, this approach could be leveraged for more advanced multilingual categorization and analysis of material descriptions.

The combination of detailed frequency tables and visual summaries like heatmaps allows different stakeholders within SLB, from finance analysts to operations managers, to engage with the data in the manner most suited to their needs.

4 Limitations

While the analysis provided valuable insights into categorising SLB's material descriptions, it is important to acknowledge certain limitations inherent in the process and the data provided:

- **Translation Reliance:** With only 2-3 words remaining in descriptions after cleaning, translation APIs such as LangDetect and Deep Translate often struggle to accurately identify and translate the text, especially when dealing with keywords or equipment names. This limitation can result in categorization based more on

the local and global structure of words rather than their true meaning or their relationship to similar entries in other languages.

- **Absence of Cost Data:** The dataset provided for this analysis did not include financial information (e.g., unit cost, total spend) associated with the material descriptions. Consequently, while frequency analysis identifies high-volume items, it cannot directly pinpoint high-cost categories or quantify the financial impact of potential procurement decisions. The primary recommendation addresses this limitation directly.

- **Contextual Awareness:** As an external analysis, there is a lack of complete operational context that internal SLB personnel possess. Certain descriptions might be ambiguous or refer to specific internal processes or equipment types that could lead to potential misgrouping or misinterpretation of category contents without further validation by SLB subject matter experts.

- **Manual Tagging Requirement:** While machine learning significantly aids in grouping similar items, the assignment of meaningful category names to the 100 clusters requires manual review and interpretation. Furthermore, language evolves, new items are introduced, and data entry practices may change. Therefore, this is not a one-time, fully automated solution; periodic review and potential refinement of category assignments or the model itself will be necessary to maintain accuracy over time.

These limitations should be considered when interpreting the results and planning subsequent actions based on this report.

5 Recommendations & Next Steps

Based on the analysis conducted and the identified limitations, the following recommendations and next steps are proposed for SLB:

1. **Integrate Financial Data:** The most crucial next step is for SLB to **re-run this analysis methodology internally using a dataset that includes associated financial cost data**. Correlating the identified categories with actual expenditure is essential to understand the true financial significance of each category. This will enable SLB to move beyond high-volume identification (like O-rings) to pinpoint high-spend areas, providing a solid basis for strategic financial decisions regarding supplier negotiations, potential standardisation efforts, or inventory optimisation.

2. **Extend Regional Analysis:** Apply this methodology to datasets from other SLB operational regions not included in this initial analysis (KSA, MCA, USL). This will provide a more comprehensive global view of material and service consumption patterns, verify the consistency of findings, and potentially uncover further regional specificities or opportunities for cross-regional synergies.

3. **Establish Continuous Improvement Process:** Recognise that material categorisation is an ongoing process. SLB should establish a workflow for periodic review and refinement. This could involve:

- Regularly reviewing items within existing categories, particularly 'Miscellaneous' or newly emerging clusters, for potential re-categorisation or creation of new categories.
- Incorporating feedback from operational and procurement teams to improve category definitions and accuracy.
- Adjust the granularity and number of clusters to balance between detailed, high-accuracy analysis and broader, umbrella categories that may offer lower accuracy but greater generalization, depending on business needs.

4. **Validate High-Impact Categories:** Conduct deeper dives into the categories identified as high-frequency (e.g., 'ORING') and those potentially identified as high-spend (following Recommendation 1). Validate the contents of these categories with subject matter experts to ensure accuracy and fully understand the drivers behind their

prominence. By taking these steps, SLB can build upon the foundation established by this analysis to create a robust, data-driven system for material categorisation that directly supports strategic financial and operational objectives.

5. **Use Dimensionality Reduction Techniques:** Visualise the clustering in 2 or 3 dimensions using UMAP, PCA or t-SNE to better understand the relationships between clusters and identify potential clusters that may be overlapping or misclassified. A PCA projection can help separate the itens in the embedding space based on language. Further research can be done to improve the clustering and categorisation of the data.

Appendix

1. Overall Category Distribution
2. Region Specific Category Distribution
3. PCA Projection showing language as highest variance component
4. Codes
5. References

1. Overall Category Distribution

Figure 1: Heatmap for overall category distribution

2. Region Specific Category Distribution

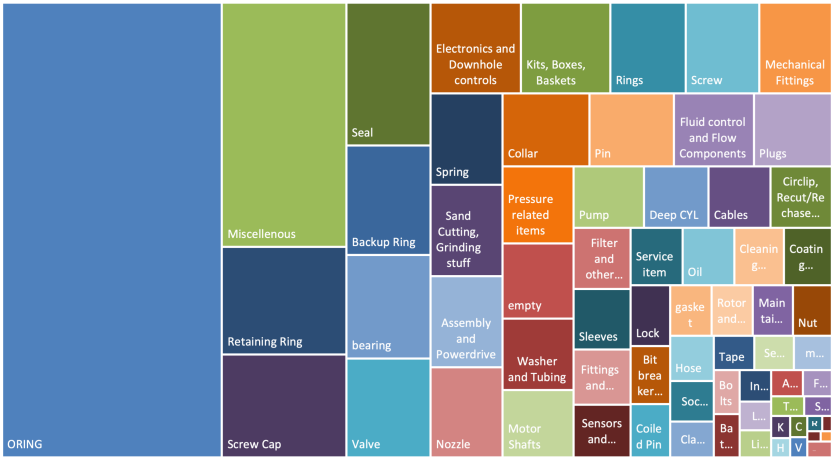


Figure 2: Heatmap for category distribution in KSA

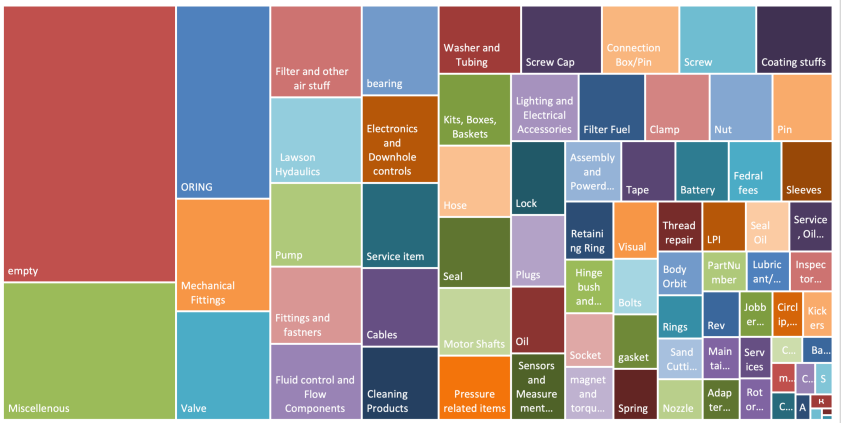


Figure 3: Heatmap for category distribution in USL

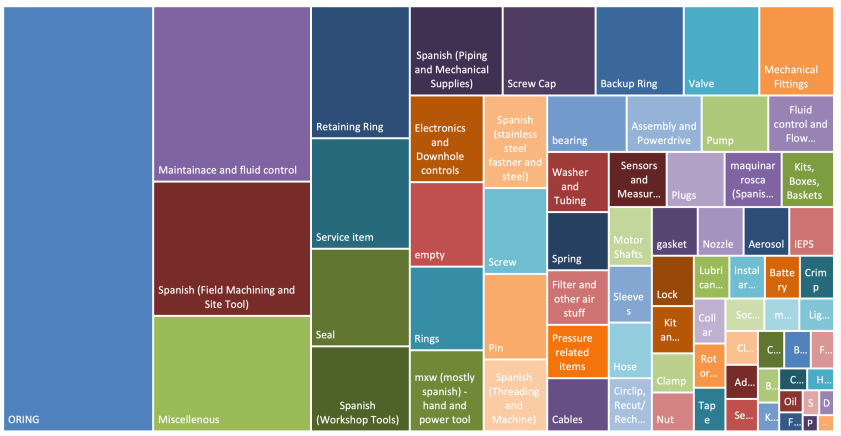


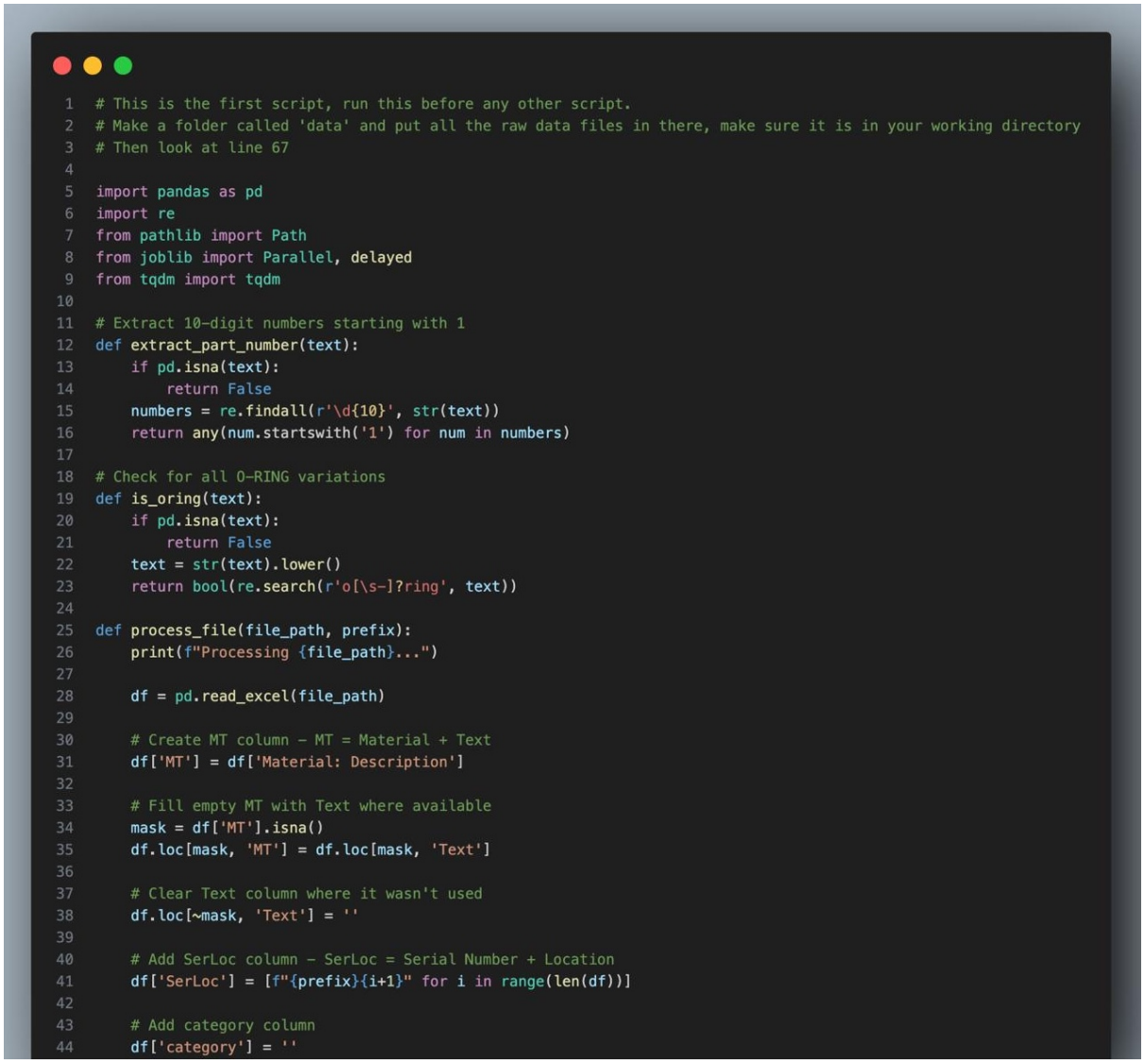
Figure 4: Heatmap for category distribution in MCA

3. PCA Projection

Figure 5: PCA projection of BERT embeddings showing language as the highest variance component.

4. Codes

step_1_preprocessing.py



```python
# This is the first script, run this before any other script.
# Make a folder called 'data' and put all the raw data files in there, make sure it is in your working directory
# Then look at line 67

import pandas as pd
import re
from pathlib import Path
from joblib import Parallel, delayed
from tqdm import tqdm

# Extract 10-digit numbers starting with 1
def extract_part_number(text):
    if pd.isna(text):
        return False
    numbers = re.findall(r'\d{10}', str(text))
    return any(num.startswith('1') for num in numbers)

# Check for all O-RING variations
def is_oring(text):
    if pd.isna(text):
        return False
    text = str(text).lower()
    return bool(re.search(r'o[\s-]?ring', text))

def process_file(file_path, prefix):
    print(f"Processing {file_path}...")

    df = pd.read_excel(file_path)

    # Create MT column - MT = Material + Text
    df['MT'] = df['Material: Description']

    # Fill empty MT with Text where available
    mask = df['MT'].isna()
    df.loc[mask, 'MT'] = df.loc[mask, 'Text']

    # Clear Text column where it wasn't used
    df.loc[~mask, 'Text'] = ''

    # Add SerLoc column - SerLoc = Serial Number + Location
    df['SerLoc'] = [f"{prefix}{i+1}" for i in range(len(df))]

    # Add category column
    df['category'] = ''
```

```
    # Apply category rules
    print("Identifying PartNumbers...")
    df.loc[df['MT'].apply(extract_part_number), 'category'] = 'PartNumber'

    print("Identifying O-RINGs...")
    df.loc[df['MT'].apply(is_oring), 'category'] = 'ORING'

    print("Identifying empty rows...")
    df.loc[df['MT'].isna(), 'category'] = 'empty'

    # Keep only required columns
    df = df[['Material: Description', 'Text', 'MT', 'SerLoc', 'category']]

    return df

def main():
    # Create processed directory if it doesn't exist
    processed_dir = Path('processed')
    processed_dir.mkdir(exist_ok=True)

    # Process each file -
    # Define the files to process and their prefixes
    files = {
        'data/EXPORT_KSA_2024.XLSX': 'KSA',
        'data/EXPORT_MCA_2024.XLSX': 'MCA',
        'data/EXPORT_USL_2024.XLSX': 'USL'
    }

    # Process files in parallel
    print("Starting parallel processing of files...")
    results = Parallel(n_jobs=-1)(
        delayed(process_file)(file_path, prefix)
        for file_path, prefix in tqdm(files.items(), desc="Processing files")
    )

    # Combine all dataframes
    print("Combining results...")
    all_data = pd.concat(results, ignore_index=True)

```

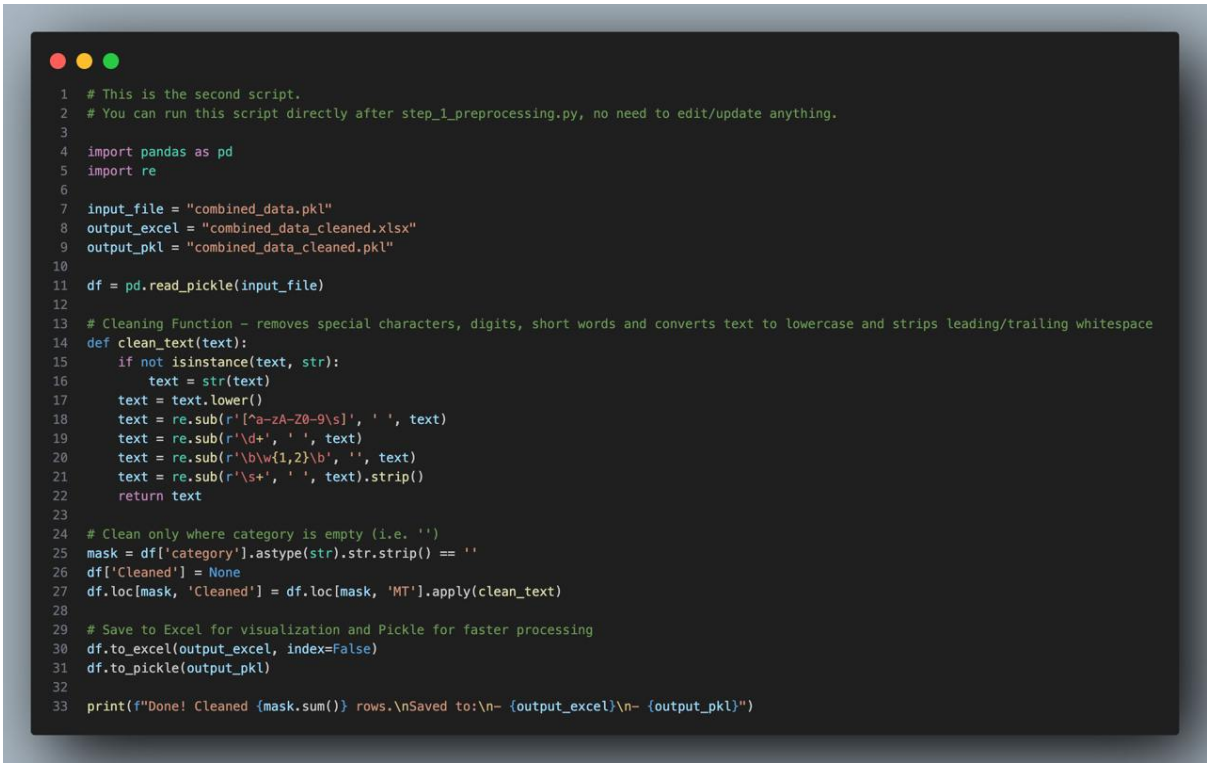
```
   # Save to Excel for visualization
    excel_output = processed_dir / 'combined_data.xlsx'
    print(f"Saving to Excel: {excel_output}")
    all_data.to_excel(excel_output, index=False)

    # Save to pickle for faster processing
    pickle_output = processed_dir / 'combined_data.pkl'
    print(f"Saving to Pickle: {pickle_output}")
    all_data.to_pickle(pickle_output)

    # Print statistics
    print("\nProcessing complete. Statistics:")
    print(f"Total rows: {len(all_data)}")
    print(f"PartNumber count: {len(all_data[all_data['category'] == 'PartNumber'])}")
    print(f"ORING count: {len(all_data[all_data['category'] == 'ORING'])}")
    print(f"Empty count: {len(all_data[all_data['category'] == 'empty'])}")
    print(f"Uncategorized count: {len(all_data[all_data['category'] == ''])}")

if __name__ == "__main__":
    main()
```

step_2_cleaning.py



```python
# This is the second script.
# You can run this script directly after step_1_preprocessing.py, no need to edit/update anything.

import pandas as pd
import re

input_file = "combined_data.pkl"
output_excel = "combined_data_cleaned.xlsx"
output_pkl = "combined_data_cleaned.pkl"

df = pd.read_pickle(input_file)

# Cleaning Function - removes special characters, digits, short words and converts text to lowercase and strips leading/trailing whitespace
def clean_text(text):
    if not isinstance(text, str):
        text = str(text)
    text = text.lower()
    text = re.sub(r'[^a-zA-Z0-9\s]', ' ', text)
    text = re.sub(r'\d+', ' ', text)
    text = re.sub(r'\b\w{1,2}\b', '', text)
    text = re.sub(r'\s+', ' ', text).strip()
    return text

# Clean only where category is empty (i.e. '')
mask = df['category'].astype(str).str.strip() == ''
df['Cleaned'] = None
df.loc[mask, 'Cleaned'] = df.loc[mask, 'MT'].apply(clean_text)

# Save to Excel for visualization and Pickle for faster processing
df.to_excel(output_excel, index=False)
df.to_pickle(output_pkl)

print(f"Done! Cleaned {mask.sum()} rows.\nSaved to:\n- {output_excel}\n- {output_pkl}")
```

step_3_BERT.py

```python
# This is the third script, run this after step_2_cleaning.py
# Check line 22 if you have a Macbook (no need to edit), or line 30 and 36 if you have a Windows (or any other device)

import pandas as pd
import torch
from sentence_transformers import SentenceTransformer
from tqdm.auto import tqdm
import pickle
import multiprocessing
from itertools import chain

input_file = "combined_data_cleaned.pkl"
output_pkl = "bert_embeddings.pkl"

df = pd.read_pickle(input_file)

# Filter rows to encode
mask = df["Cleaned"].notna() & df["Cleaned"].astype(str).str.strip().ne("")
texts_to_encode = df.loc[mask, "Cleaned"].astype(str).tolist()

# Load BERT Model with Sentence-Transformers
''' For Mac (Apple Silicon M1,M2,M3,M4) devices with MPS (Metal Performance Shaders) support
If you are using a Mac with an apple silicon chip, the MPS backend will be used for faster computation on your GPU.
This requires PyTorch to be installed with MPS support (available in PyTorch version 1.12 or later).'''

device = "mps" if torch.backends.mps.is_available() else "cpu"  # Check if MPS is available, else use CPU
model = SentenceTransformer("all-MiniLM-L6-v2", device=device)

#--------------------------------
'''For Windows (or any device without MPS support, including Intel/AMD Macs):
If you're using Windows or a device without MPS support (e.g., Intel-based Macs or non-Apple systems),
you need to **comment the above line** and **uncomment the line below**.
This will make sure the model uses your GPU via CUDA (if available) or the CPU.

To run on CUDA-enabled GPUs (NVIDIA only), ensure you have PyTorch installed with CUDA support.
[pip install torch torchvision torchaudio]  # Make sure to install the correct CUDA version based on your GPU.
If CUDA is available, the model will run on your GPU. Otherwise, it defaults to the CPU.'''

# device = "cuda" if torch.cuda.is_available() else "cpu"  # Check if CUDA (NVIDIA GPU) is available, else use CPU
# model = SentenceTransformer("all-MiniLM-L6-v2", device=device)

# --------------------------------
```

```python

# Multiprocessing Setup
num_workers = multiprocessing.cpu_count()
batch_size = 1024

# Encode in Batches
def encode_batch(texts):
    return model.encode(texts, convert_to_tensor=False)

tqdm.pandas()

embeddings_batches = list(tqdm(
    map(encode_batch, [texts_to_encode[i:i+batch_size] for i in range(0, len(texts_to_encode), batch_size)]),
    total=(len(texts_to_encode) + batch_size - 1) // batch_size
))

flattened_embeddings = list(chain.from_iterable(embeddings_batches))

# Assign embeddings only to matching rows
bert_embeddings_column = [None] * len(df)
j = 0
for i in range(len(df)):
    if mask.iloc[i]:
        bert_embeddings_column[i] = flattened_embeddings[j]
        j += 1

df["BERT_Embedding"] = bert_embeddings_column

# Save Output - only pickle because Excel can't handle the 384 dimensions of the BERT embeddings
with open(output_pkl, "wb") as f:
    pickle.dump(df, f)

print(f"BERT Embeddings Generated & Saved:\n- {output_pkl}")
```

step_4_k-means.py

```python
# This is the fourth script, run this after step_3_BERT.py
# You can check line 25 if you want to change the number of clusters
# After running, check line 47 - very very important, don't forget to do this

import pickle
from sklearn.cluster import KMeans
import pandas as pd
from tqdm import tqdm

input_pickle = "bert_embeddings.pkl"
output_pickle = "clustered_items.pkl"
output_excel = "clustered_items.xlsx"

# Load DataFrame with embeddings
with open(input_pickle, 'rb') as file:
    df = pickle.load(file)

# Mask for rows with valid embeddings
mask = df["BERT_Embedding"].notna()

# Extract valid embeddings
embeddings = [emb for emb in df.loc[mask, "BERT_Embedding"]]

# KMeans Clustering
# Adjust 'n_clusters' to balance granularity and accuracy: more clusters increase granularity, fewer improve accuracy
print("Clustering started...")
kmeans = KMeans(n_clusters=100, random_state=42, n_init='auto')
cluster_labels = kmeans.fit_predict(embeddings)
print("Clustering complete!")

# Assign cluster numbers only to valid rows
df["cluster_number"] = None
df.loc[mask, "cluster_number"] = cluster_labels

# Save full version with embeddings as Pickle
with open(output_pickle, 'wb') as file:
    pickle.dump(df, file)

# Save Excel without BERT_Embedding column for visualization
df_excel = df.drop(columns=["BERT_Embedding"])
df_excel.to_excel(output_excel, index=False)

print(f"Clustered Excel saved to: {output_excel}")
print(f"Full clustered data (with embeddings) saved to: {output_pickle}")


# Now make an excel named "cluster_names.xlsx" by manually looking at random sample items in each cluster number from the output excel
# cluster_names.xlsx should have two columns: "DL_category" - containing category number 0-99 (0 to n_clusters-1)
# and "Name" - category name based on general item type in that cluster
```

step_5_output.py

```python
# This is the last (fifth) script
# Check line 57, then run this script if you want a combined excel as output
# If you want separate excel files for each region, you can check line 32

import pandas as pd

items_file = "clustered_items.xlsx"
freq_file = "cluster_names.xlsx"
output_base = "clustered_items_updated"
stats_file = "category_statistics.xlsx"

df_items = pd.read_excel(items_file)
df_freq = pd.read_excel(freq_file)

# Create cluster_number → category name mapping
cluster_map = dict(zip(df_freq["Cluster_Number"], df_freq["Name"]))

# Update category column only if cluster_number is present
def update_category(row):
    cluster = row.get("cluster_number")
    if pd.notna(cluster) and cluster in cluster_map:
        return cluster_map[cluster]
    return row['category']

df_items["category"] = df_items.apply(update_category, axis=1)

# Compute overall category frequency (combined)
overall_freq = df_items["category"].value_counts().reset_index()
overall_freq.columns = ["Category", "Total_Frequency"]

# --------------------------
# Split by SerLoc (optional)
# --------------------------

''' Splitting into region-specific Excel files is commented out. If you want to save separate files for regions,
uncomment the code below and add the new files names and their prefixes. This will give you both combined and region-specific files.
Comment out line 54 if you don't want to save the combined file.

# df_ksa = df_items[df_items['SerLoc'].astype(str).str.startswith("KSA")]
# df_mca = df_items[df_items['SerLoc'].astype(str).str.startswith("MCA")]
# df_usl = df_items[df_items['SerLoc'].astype(str).str.startswith("USL")]

# # Save split Excel files for each region (if splitting is enabled)
# df_ksa.to_excel(f"{output_base}_KSA.xlsx", index=False)
# df_mca.to_excel(f"{output_base}_MCA.xlsx", index=False)
# df_usl.to_excel(f"{output_base}_USL.xlsx", index=False)
'''

# --------------------------------
# Save the combined dataframe (no splitting)
# --------------------------------

# Save the whole combined dataframe into a single Excel file
df_items.to_excel(f"{output_base}_combined.xlsx", index=False)

# Region-wise category frequencies
# Add more locations with their prefixes in the same format as below to include them in the stats, then --> check line 68
ksa_freq = df_items[df_items['SerLoc'].astype(str).str.startswith("KSA")]["category"].value_counts().reset_index()
ksa_freq.columns = ["Category", "KSA_Frequency"]

mca_freq = df_items[df_items['SerLoc'].astype(str).str.startswith("MCA")]["category"].value_counts().reset_index()
mca_freq.columns = ["Category", "MCA_Frequency"]

usl_freq = df_items[df_items['SerLoc'].astype(str).str.startswith("USL")]["category"].value_counts().reset_index()
usl_freq.columns = ["Category", "USL_Frequency"]

# Merge all stats into one summary table
# Add the new locations here also in the same format as below
stats_df = overall_freq.merge(ksa_freq, on="Category", how="left") \
                       .merge(mca_freq, on="Category", how="left") \
                       .merge(usl_freq, on="Category", how="left")

# Fill NaNs with 0 and convert to int
stats_df.fillna(0, inplace=True)
stats_df[["Total_Frequency", "KSA_Frequency", "MCA_Frequency", "USL_Frequency"]] = stats_df[
    ["Total_Frequency", "KSA_Frequency", "MCA_Frequency", "USL_Frequency"]
].astype(int)

# Save statistics to Excel
stats_df.to_excel(stats_file, index=False)

# Done
print("Category updated using cluster_number.")
print("Files saved:")
print(f"- {output_base}_combined.xlsx")  # Only one combined output file now
print(f"- {stats_file}")
```

5. References

1. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of NAACL-HLT 2019 (pp. 4171-4186).

2. Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) (pp. 3982-3992).

3. MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In Proceedings of the fifth Berkeley symposium on mathematical statistics and probability (Vol. 1, No. 14, pp. 281-297).

4. Jolliffe, I. T., & Cadima, J. (2016). Principal component analysis: a review and recent developments. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 374(2065), 20150202.

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

6. McKinney, W. (2010). Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference (Vol. 445, pp. 51-56).

7. Friedl, J. E. F. (2006). Mastering Regular Expressions. O'Reilly Media, Inc.