Grouping and Clustering Invalid Application Reasons

Overview

This script processes and categorises **invalid application reasons** by grouping them into predefined themes, creating a structured Word document, and performing **clustering** with **TF-IDF vectorisation** and **TSNE visualisation**. The final output includes:

1. A **Word document** summarising grouped reasons.
2. A **CSV file** containing structured reasons.
3. A **TSNE visualisation** of clustered reasons.

The purpose of this script was to automate the process of grouping similar invalid application reasons. As there isn't a obvious matching between the different reasons (apart from vague groupings such as missing a document, or not paid total amount), a simple matching procedure would not be sufficient. That is why we are using cluster analysis, which is a method of unsupervised machine learning. It is worth noting, this script only aims to provide a rough grouping of the data. If this was to be generalised for further datasets, it would be ideal to properly train and score a clustering model, to enusre high accuracy.

Step 1: Load and Preprocess Data

The script starts by **loading the dataset** containing invalid application reasons from a CSV file.

Theme-based classification:

- Reasons are **classified into predefined themes** using regular expressions (e.g., "Missing Reports", "Incorrect Fee", "Missing Drawings").
- Any reason that does not fit a theme is placed in the `"Other"` category.

Subcategorisation of "Incorrect Fee":

- The `"Incorrect Fee"` category is further split into:
  - `"Incorrect Fee - Underpayment"` (if the reason contains words like `"insufficient"` or `"further fee"`).
  - `"Incorrect Fee - Other"` (all remaining incorrect fee reasons).

Step 2: Create a Structured Word Document

A **Word document** is created to store grouped reasons.

- Each theme is added as a **section heading** with the **number of occurrences**.

- The `"Incorrect Fee - Other"` category includes additional **counts for underpayments**.
- The document is saved as `Grouped_Invalid_Reason_Details.docx`.

Step 3: Perform Text Clustering

The script **clusters the reasons using machine learning techniques**:

1. **TF-IDF Vectorisation**:
    - Converts text into numerical representations.
    - Removes common stop words.
    - Limits features to **500** for efficiency.
2. **Dimensionality Reduction with TSNE**:
    - Reduces **TF-IDF features to 2D** for visualisation.
3. **K-Means Clustering**:
    - Groups reasons into **10 clusters**.

Step 4: Visualise Clusters with TSNE

The clusters are **plotted using TSNE**:

- The visualisation **color-codes clusters**.
- The plot is **saved as** `TSNE_Clusters.png`.

Step 5: Generate a CSV File with Grouped Reasons

The script **creates a CSV file** where:

- Each **theme becomes a column**.
- Reasons **are stored in rows**, ensuring alignment by filling empty spaces with `None`.

The final **CSV file** `Grouped_Invalid_Reason_Details.csv` contains structured data.

Summary of Outputs

✅ `Grouped_Invalid_Reason_Details.docx` → A structured Word document grouping invalid reasons.
✅ `TSNE_Clusters.png` → A visualisation of clustered reasons.
✅ `Grouped_Invalid_Reason_Details.csv` → A CSV file storing grouped reasons for analysis.

This script **automates text classification and clustering** to better understand invalid application reasons. 🚀

In [ ]:
```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.manifold import TSNE
from sklearn.cluster import KMeans
from collections import defaultdict
from docx import Document
import re

# Load data
df = pd.read_csv('invalid-applications-sample-reasons.xlsx - invalid-application-sa

# Define themes for clustering with more granular categories for Missing Documents
themes = {
    "Incorrect Fee": r"(fee|payment|underpayment|overpayment)",
    "Missing Plans": r"(site plan|floor plan|elevation)",
    "Missing Reports": r"(report|statement|assessment|survey)",
    "Missing Forms": r"(form|certificate|ownership)",
    "Validation Checklist": r"(checklist|validation|requirement)",
    "Missing Details": r"(details|clarify|information)",
    "Missing Drawings": r"(drawing|design|sketch|diagram)",
    "Other": r".*"
}

# Initialise a dictionary to hold the themes and their corresponding rows
grouped_reasons = defaultdict(list)

# Iterate over the rows and classify based on themes
for index, row in df.iterrows():
    reason = row["Invalid Reason Details"]
    if pd.isna(reason):
        continue
    matched = False
    for theme, pattern in themes.items():
        if re.search(pattern, reason, re.IGNORECASE):
            grouped_reasons[theme].append(reason)
            matched = True
            break
    if not matched:
        grouped_reasons["Other"].append(reason)

# Separate "Incorrect Fee" into "Incorrect Fee - Underpayment" if specific words ar
incorrect_fee = grouped_reasons.get("Incorrect Fee", [])
underpayment_fee = [reason for reason in incorrect_fee if re.search(r"insufficient|
remaining_fee = [reason for reason in incorrect_fee if reason not in underpayment_f
grouped_reasons["Incorrect Fee - Underpayment"] = underpayment_fee
grouped_reasons["Incorrect Fee - Other"] = remaining_fee

# Remove the old "Incorrect Fee" key
del grouped_reasons["Incorrect Fee"]

# Reorder groups so "Incorrect Fee - Other" and "Other" appear at the end
ordered_keys = [key for key in grouped_reasons if key not in ["Incorrect Fee - Othe
ordered_keys += ["Incorrect Fee - Other", "Other"]

# Create a Word document
doc = Document()
doc.add_heading('Grouped Invalid Reason Details', level=1)

# Add each group and its count to the document
```

```
for theme in ordered_keys:
    reasons = grouped_reasons[theme]
    count = len(reasons)
    if theme == "Incorrect Fee - Other":
        underpayment_count = len(grouped_reasons.get("Incorrect Fee - Underpayment'
        doc.add_heading(f"{theme} ({count} instances, Underpayment: {underpayment_c
    else:
        doc.add_heading(f"{theme} ({count} instances):", level=2)

# Save the document
doc.save('Grouped_Invalid_Reason_Details.docx')

# Perform TF-IDF vectorisation
all_reasons = df["Invalid Reason Details"].dropna().tolist()
vectorizer = TfidfVectorizer(stop_words='english', max_features=500)
tfidf_matrix = vectorizer.fit_transform(all_reasons)

# Reduce dimensions using TSNE
tsne = TSNE(n_components=2, random_state=42, perplexity=30, n_iter=1000)
tsne_results = tsne.fit_transform(tfidf_matrix.toarray())

# Cluster the data into 10 groups using KMeans
kmeans = KMeans(n_clusters=10, random_state=42)
clusters = kmeans.fit_predict(tfidf_matrix)

# Prepare data for visualization
cluster_colors = clusters

plt.figure(figsize=(10, 8))
scatter = plt.scatter(tsne_results[:, 0], tsne_results[:, 1], c=cluster_colors, cma
plt.colorbar(scatter, ticks=range(10), label='Clusters')
plt.title('TSNE Visualization of Clusters')
plt.xlabel('TSNE Dimension 1')
plt.ylabel('TSNE Dimension 2')
plt.savefig('TSNE_Clusters.png')
plt.show()

# Create a CSV file with themes as columns and reasons as rows
max_rows = max(len(reasons) for reasons in grouped_reasons.values())
output_data = {theme: grouped_reasons[theme] + [None] * (max_rows - len(grouped_rea
output_df = pd.DataFrame(output_data)
output_df.to_csv('Grouped_Invalid_Reason_Details.csv', index=False)

print("Outputs generated:")
print("1. Document 'Grouped_Invalid_Reason_Details.docx' created with grouped reasc
print("2. Visualisation 'TSNE_Clusters.png' saved.")
print("3. CSV file 'Grouped_Invalid_Reason_Details.csv' created with grouped reasor
```