IT SERVICE TICKET CLASSIFICATION

NLP PROJECT

SUPERVISOR: DR. SANJIV SINGH

SCHOOL OF BUSINESS

UPES DEHRADUN

SUBMITTED BY :- MANSI SINGH

Project Report: Ticket Classification using Machine Learning and Deep Learning Approaches

1. Introduction

This project focuses on classifying IT service tickets into their respective categories using Natural Language Processing (NLP) and Machine Learning (ML) techniques. The dataset consists of 47,837 records of ticket descriptions along with their corresponding topic groups. The goal is to build a robust classification system that accurately predicts the topic group for a given ticket description.

2. Dataset Overview

- Source: Pre-processed dataset with 47,837 rows and 2 main columns: Document: Ticket text description Topic_group: Category of the ticket (e.g., Hardware, HR Support, Access, Miscellaneous, Others) - Classes after grouping: Hardware: 13,617 HR Support: 10,915 Access: 7,125 Miscellaneous: 7,060 Others: 9,120

3. Exploratory Data Analysis (EDA)

- Checked for missing values and duplicates → None found. - Performed distribution analysis: Hardware and HR Support dominate the dataset. - Added text-based features: Number of characters (n_c), words (n_w), and sentences (n_s). - Summary statistics: Average word count per ticket: ~56 Maximum length: 7015 characters - Visualized data using pair plots and correlation heatmaps. - Detected multicollinearity between n_c and n_w, retained n_c.

4. Data Preprocessing

- Lowercased text - Removed punctuation, numbers, and special characters - Tokenized text - Removed stopwords - Applied lemmatization - Generated WordClouds for each category to visualize frequent terms

5. Feature Engineering

- Used TF-IDF Vectorizer with max_features=300. - Created additional feature: n_c (character count). - Scaled features using MinMaxScaler. - Final feature set: 301 features.

6. Model Development

Model	Precision	Recall	F1 Score
GaussianNB	0.69	0.71	0.69
MultinomialNB	0.80	0.75	0.77

BernoulliNB	0.66	0.65	0.64
Random Forest	0.84	0.83	0.83
XGBoost	0.84	0.83	0.83

Classical ML Models: - Naive Bayes Models: GaussianNB → F1 Score: 0.69 MultinomialNB → F1 Score: 0.77 (Best among NB models) - Tree-Based Models: Random Forest → Accuracy: 83%, F1 Score: 0.83 XGBoost → Accuracy: 83%, F1 Score: 0.83

Performance Summary

7. Deep Learning Approach

Word2Vec + Bi-LSTM Model: - Used Word2Vec (CBOW) for embedding (dimension = 100). - Model Architecture: Embedding layer (pre-trained with Word2Vec) Bidirectional LSTM (128 units) Dropout (0.5) LSTM (64 units) Dense layer (ReLU) Output layer with softmax - EarlyStopping applied to prevent overfitting. - Achieved 90% accuracy on validation set.

8. Observations

- Deep Learning significantly outperformed traditional ML models due to its ability to capture contextual dependencies. - Among ML models, XGBoost and Random Forest gave the best results (~83% accuracy). - Naive Bayes models were simple but less effective compared to ensemble methods and deep learning.

9. Key Challenges

- Class imbalance among categories. - Handling high-dimensional TF-IDF vectors. - Large vocabulary size requiring efficient embedding representation.

10. Conclusion

- Word2Vec + Bi-LSTM proved to be the most effective model, achieving 90% accuracy. - Classical models like Random Forest and XGBoost performed well with 83% accuracy. - This project demonstrates the advantage of deep learning methods over classical models for NLP classification tasks.

11. Future Scope

- Implement BERT-based transformers for better semantic understanding. - Use data augmentation to handle class imbalance. - Deploy the model as an API for real-time ticket classification.

12. Code Implementation

Below are the essential code snippets used in the project, organized by steps.

Importing Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import precision_score, recall_score, f1_score, classification_report
```

Data Loading & Initial Exploration

```
df = pd.read_csv('all_tickets_processed_improved_v3.csv.zip')
print(df.shape)
print(df.head())
```

Text Preprocessing

```
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

def preprocess_text(text):
    text = text.lower()
    text = re.sub(r'[^a-z\s]', '', text)
    words = nltk.word_tokenize(text)
    words = [word for word in words if word not in stop_words]
    words = [lemmatizer.lemmatize(word) for word in words]
    return ' '.join(words)

df['Document'] = df['Document'].apply(preprocess_text)
```

TF-IDF Feature Engineering

```
tfidf = TfidfVectorizer(max_features=300)
X = tfidf.fit_transform(df['Document']).toarray()
y = df['Topic_group']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Model Training & Evaluation (Example: MultinomialNB)

```
from sklearn.naive_bayes import MultinomialNB

mnb = MultinomialNB()
mnb.fit(X_train, y_train)
y_pred = mnb.predict(X_test)

print(classification_report(y_test, y_pred))
```

Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
```

```
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)

print(classification_report(y_test, y_pred_rf))
```

Deep Learning Model (Word2Vec + Bi-LSTM)

```
from gensim.models import Word2Vec
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout, Bidirectional

# Train Word2Vec
sentences = [nltk.word_tokenize(text) for text in df['Document']]
w2v_model = Word2Vec(sentences, vector_size=100, window=5, min_count=1)

# Build LSTM Model
model = Sequential()
model.add(Embedding(input_dim=vocab_size, output_dim=100, input_length=maxlen, weights=[embedding_ma
model.add(Bidirectional(LSTM(128, return_sequences=True)))
model.add(Dropout(0.5))
model.add(LSTM(64))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dense(len(label_encoder.classes_), activation='softmax'))

model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(x1_train, y1_train, epochs=5, batch_size=64, validation_split=0.2)
```