**Multilingual Text Classification Model**

**Executive Summary**

This report documents the development and evaluation of a multilingual text classification model capable of handling both English and Bengali text inputs. The model aims to categorize texts into various training topics based on their contexts. Despite challenges with language diversity and class imbalance, the final model achieves a test accuracy of 46-50%, representing a significant improvement over initial attempts with more complex architectures like BERT (29-32%).

Git Link: <https://github.com/ALIF-AL-RAZI/TechTalents>

**Problem Statement**

The task involved classifying text data ("Context") into appropriate "Training Topics" while handling:

* Multilingual content (primarily English and Bengali)
* Class imbalance
* Limited samples for some classes

**Methodology**

**Data Preparation**

1. **Initial Dataset**: The data was loaded from "Data & Topics.xlsx"
2. **Preprocessing**:
   * Removed duplicates
   * Filtered classes with fewer than 15 samples
   * Removed empty documents after cleaning

**Text Preprocessing Pipeline**

A robust multilingual preprocessing pipeline was implemented to handle text in multiple languages:

1. **Text Cleaning**:
   * URL, hashtag, and mention removal
   * Emoji removal using Unicode pattern matching
   * Punctuation removal (including custom Bangla, Urdu, and Arabic punctuation)
2. **Language-Specific Processing**:
   * **Language Detection**: Using langdetect to determine the text language
   * **Bengali Text**:
     + Word normalization with SBNLTK preprocessor
     + Tokenization with SBNLTK word tokenizer
     + Stemming with SBNLTK stemmer
     + Dust removal with SBNLTK preprocessor
   * **English Text**:
     + Spelling correction with TextBlob (with fallback)
     + Tokenization with NLTK
     + Stopword removal
     + Lemmatization with WordNet
3. **Final Processing**:
   * Character normalization
   * Token filtering
   * Token joining to create cleaned text

**Feature Engineering**

After considering various techniques:

1. **BERT** (initially tested):
   * Achieved only 29-32% test accuracy
   * Too complex for the dataset size
2. **Word2Vec** (also tested):
   * Showed some improvement but still suboptimal
3. **TF-IDF Vectorization** (final choice):
   * Parameters:
     + max\_features=5000 (to prevent overfitting)
     + min\_df=2 (ignore terms in fewer than 2 documents)
     + max\_df=0.8 (ignore terms in more than 80% of documents)
     + ngram\_range=(1, 2) (unigrams and bigrams)

**Class Imbalance Handling**

SMOTE (Synthetic Minority Over-sampling Technique) was applied to address class imbalance:

* Dynamically calculated k\_neighbors parameter
* Generated synthetic samples for minority classes
* Created a more balanced training dataset

**Model Selection**

Multiple algorithms were considered:

1. **Logistic Regression** (chosen final model):
   * Configuration:
     + C=1.0
     + max\_iter=1000
     + multi\_class='multinomial'
     + solver='lbfgs'
2. Other tested models:
   * Support Vector Machine (SVM)
   * Random Forest
   * Naive Bayes

**Results and Analysis**

**Performance Metrics**

|  |  |  |
| --- | --- | --- |
| **Metric** | **Training Set** | **Test Set** |
| Accuracy | ~95% | 46-50% |

**Key Observations**

1. **High Training vs. Lower Test Accuracy**:
   * The substantial gap (95% vs. 46-50%) indicates overfitting
   * Despite SMOTE and feature selection, the model still memorizes training data to some extent
2. **Improvement Over Baseline**:
   * Significant improvement over initial BERT attempts (29-32%)
   * Shows that simpler models can outperform complex ones on this dataset
3. **Feature Importance Analysis**:
   * Most important features were identified for each class
   * Visualized the top features for the most prevalent classes

**Confusion Matrix Analysis**

A normalized confusion matrix was generated to identify:

* Classes with highest prediction accuracy
* Common misclassification patterns
* Areas for potential improvement

**Deployment and Usage**

The model and vectorizer were saved for production use:

* text\_classification\_model\_T2.pkl (model)
* tfidf\_vectorizer.pkl (vectorizer)

A prediction function was implemented to:

1. Preprocess incoming text
2. Vectorize using the same TF-IDF vectorizer
3. Predict the most likely class
4. Return top 3 predictions with probabilities

**Sample Prediction**

Text: "Police arrested three individuals involved in corruption at the ministry"

The model provides:

* Primary predicted topic
* Alternative predictions with confidence scores

**Challenges and Limitations**

1. **Multilingual Processing**:
   * Handling both English and Bengali adds complexity
   * Language detection sometimes fails on short texts
2. **Class Imbalance**:
   * Despite SMOTE, some classes remain underrepresented
   * Classes with fewer than 15 samples were removed
3. **Overfitting**:
   * High train accuracy (95%) vs. lower test accuracy (46-50%)
   * Additional regularization might help reduce this gap
4. **Limited Dataset Size**:
   * More data would likely improve performance
   * Some topics may not have enough examples for effective learning

**Recommendations for Improvement**

1. **Data Collection**:
   * Gather more samples, especially for underrepresented classes
   * Ensure balanced representation across languages
2. **Feature Engineering**:
   * Try domain-specific feature extraction
   * Consider combining TF-IDF with word embeddings
3. **Model Tuning**:
   * Implement cross-validation for hyperparameter optimization
   * Explore regularization to reduce overfitting
4. **Ensemble Methods**:
   * Create language-specific models and combine predictions
   * Use voting or stacking of multiple classifiers
5. **Advanced Techniques**:
   * Consider distilled BERT or smaller transformer models
   * Explore transfer learning from pre-trained multilingual models

**Conclusion**

The multilingual text classification model demonstrates satisfactory performance given the complexity of the task. The final TF-IDF + Logistic Regression approach outperformed more complex models, achieving 46-50% test accuracy across multiple topics. While there is room for improvement, the current implementation provides a solid foundation for classifying texts into relevant training topics in a multilingual context.