Greeklish-English Text Classification System Documentation

1. Data Sources and Collection

1.1 Data Source Selection

The system uses a diverse set of data sources to ensure robust classification:

English Sources

- **News Websites**: Reuters, BBC, CNN
- **Blog Posts**: Medium, WordPress blogs
- **Social Media**: Twitter, Reddit
- **Academic Texts**: Research papers, textbooks

Greeklish Sources

- **Social Media**: Greek Twitter accounts, Facebook groups
- **Online Forums**: Greek tech forums, discussion boards
- **Personal Blogs**: Greek bloggers using Latin characters
- **Chat Messages**: Greek WhatsApp groups, Telegram channels

1.2 Data Collection Methodology

Web Scraping Process

1. **URL Collection**
    - Automated URL discovery using web crawlers
    - Manual curation of high-quality sources
    - Regular updates of source list
2. **Content Extraction**
    - BeautifulSoup for HTML parsing
    - JavaScript rendering for dynamic content
    - Rate limiting and polite crawling

3. **Data Validation**
    - Language verification
    - Content quality checks
    - Duplicate detection

2. Preprocessing Pipeline

2.1 Text Cleaning

Basic Cleaning

```
def clean_text(text):
    # Remove HTML tags
    text = re.sub(r'<[^>]+>', '', text)

    # Convert to lowercase
    text = text.lower()

    # Remove special characters
    text = re.sub(r'[^\w\s]', '', text)

    # Handle whitespace
    text = ' '.join(text.split())

    return text
```

Language-Specific Processing

- Greeklish pattern normalization
- Common character substitutions
- Word boundary detection

2.2 Feature Engineering

TF-IDF Features

- Word-level features (1-3 grams)
- Character-level features (2-4 grams)
- Custom tokenization for Greeklish

Custom Features

- Character frequency ratios

- Word pattern analysis
- Language-specific markers

3. Model Selection and Training

3.1 Model Selection Rationale

Traditional ML Models

1. **Logistic Regression**
    - Fast training and inference
    - Good for linearly separable data
    - Easy to interpret
2. **Support Vector Machine (SVM)**
    - Effective for high-dimensional data
    - Good generalization
    - Robust to noise
3. **Random Forest**
    - Handles non-linear patterns
    - Feature importance analysis
    - Ensemble learning benefits

3.2 Training Process

Data Preparation

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

# Feature vectorization
vectorizer = TfidfVectorizer(
    min_df=2,
    max_features=5000,
    ngram_range=(1, 3)
)
```

Model Training

```
# Logistic Regression
lr_model = LogisticRegression(
    C=1.0,
    class_weight='balanced',
    solver='lbfgs'
)

# SVM
svm_model = LinearSVC(
    C=1.0,
    class_weight='balanced'
)

# Random Forest
rf_model = RandomForestClassifier(
    n_estimators=100,
    class_weight='balanced'
)
```

4. Evaluation and Performance

4.1 Metrics

Classification Metrics

- Accuracy: 99.78% (±0.88%)
- Precision: 100%
- Recall: 98.26%
- F1-Score: 99.11%

Cross-Validation Results

```
scores = cross_val_score(model, X, y, cv=5)
print(f"Cross-validation scores: {scores}")
print(f"Average CV score: {scores.mean():.3f} (±{scores.std() * 2:.3f})")
```

4.2 Visualizations

Performance Plots

1. Confusion Matrix
2. ROC Curve

3. Precision-Recall Curve
4. Feature Importance Plot

5. Challenges and Solutions

5.1 Data Quality Challenges

Challenge 1: Mixed Language Content

- **Problem**: Texts containing both English and Greeklish
- **Solution**:
    - Implement language ratio analysis
    - Use majority language classification
    - Add confidence scores

Challenge 2: Inconsistent Greeklish Patterns

- **Problem**: Multiple ways to write same Greek word
- **Solution**:
    - Create normalization dictionary
    - Implement pattern matching
    - Use character frequency analysis

5.2 Model Performance Challenges

Challenge 1: Overfitting

- **Problem**: Model performs well on training but poorly on new data
- **Solution**:
    - Implement cross-validation
    - Use regularization
    - Feature selection

Challenge 2: Class Imbalance

- **Problem**: Uneven distribution of English and Greeklish texts
- **Solution**:
    - Use class weights
    - Implement SMOTE
    - Balanced sampling

6. Future Improvements

6.1 Planned Enhancements

1. Deep learning models (BERT, RoBERTa)
2. Real-time classification API
3. Multi-language support
4. Improved preprocessing pipeline

6.2 Research Directions

1. Transfer learning from Greek language models
2. Context-aware classification
3. Domain adaptation techniques
4. Active learning integration

7. Maintenance and Support

7.1 Regular Tasks

- Weekly model retraining
- Daily data updates
- Performance monitoring
- Error logging

7.2 Troubleshooting Guide

1. Memory errors
    - Solution: Batch processing
2. Model loading failures
    - Solution: Version compatibility checks
3. Prediction inconsistencies
    - Solution: Input validation

License