WEB SERVICES CLASSIFICATION PROJECT

## Complete Documentation & User Guide

Contents

[Complete Documentation & User Guide 1](#_Toc208947568)

[1. Project Overview 1](#_Toc208947569)

[2. System Architecture 1](#_Toc208947570)

[3. Project Structure 2](#_Toc208947571)

[4. Installation & Setup 3](#_Toc208947572)

[5. Execution Steps 4](#_Toc208947573)

[6. Model Implementation 4](#_Toc208947574)

[7. Evaluation System 5](#_Toc208947575)

[8. Visualization & Analysis 5](#_Toc208947576)

[9. Results & Performance 6](#_Toc208947577)

[10. Advanced Usage 6](#_Toc208947578)

[11. Conclusion 6](#_Toc208947579)

## 1. Project Overview

**Purpose**

This project implements a **comprehensive machine learning and deep learning pipeline** for web service classification. It automatically categorizes web services into predefined categories using both traditional ML algorithms and modern DL models.

**Key Features**

* **Multi-Model Approach**: Logistic Regression, Random Forest, XGBoost (ML) and BiLSTM (DL)
* **Dual Feature Extraction**: TF-IDF and SBERT embeddings
* **Scalable Categories**: Supports 10, 20, 40, and 50 category tasks
* **Comprehensive Evaluation**: Top-K accuracy, confusion matrices, classification reports
* **Rich Visualizations**: Line plots, bar charts, radar plots for in-depth analysis
* **Automated Pipeline**: End-to-end execution with minimal manual intervention

**Problem Solved**

Efficient categorization of web services based on descriptions—enabling **better discovery, organization, and recommendations**.

## 2. System Architecture

**High-Level Pipeline**

Data Input → Preprocessing → Feature Extraction → ML/DL Training → Evaluation → Visualization

**Components**

* **Data Input**: Raw datasets (CSV), web service descriptions, category labels
* **Preprocessing**: Text cleaning, tokenization, label encoding, dataset splits
* **Feature Extraction**: TF-IDF vectorization, SBERT embeddings
* **Model Training**:
  + ML: Logistic Regression, Random Forest, XGBoost
  + DL: BiLSTM with dense layers
* **Evaluation**: Metrics, top-K accuracy, confusion matrices, reports
* **Visualization**: Line plots, bar charts, radar plots, comparisons

**Data Flow**

1. Input data → 2. Preprocessing → 3. Feature Extraction →
2. Model Training → 5. Evaluation → 6. Visualization

## 3. Project Structure

web\_services\_classification/

├── main.py # Pipeline orchestrator

├── requirements.txt # Dependencies

├── README.md # Project info

│

├── src/ # Source code

│ ├── config.py # Configuration settings

│ ├── preprocessing/ # Data preprocessing

│ ├── modeling/ # ML & DL models

│ └── utils/ # Utility functions

│

├── data/ # Datasets & features

├── models/ # Trained ML/DL models

├── results/ # Analysis outputs

└── logs/ # Execution logs

## 4. Installation & Setup

**Prerequisites**

* Python 3.8+
* CUDA-compatible GPU (optional, for DL training)
* 8GB+ RAM, 10GB+ disk space

**Steps**

1. Clone repo
2. git clone <repository-url>
3. cd web\_services\_classification
4. Create & activate virtual environment
5. python -m venv venv
6. source venv/bin/activate # Mac/Linux
7. venv\Scripts\activate # Windows
8. Install dependencies
9. pip install -r requirements.txt
10. pip install sentence-transformers
11. Verify installation
12. python -c "import tensorflow; print(tensorflow.\_\_version\_\_)"
13. python -c "import sentence\_transformers; print('SBERT ready')"

**Configuration**

Customize src/config.py for data paths, model parameters, and training settings.

## 5. Execution Steps

**Quick Start (entire pipeline)**

python main.py --phase all

**Step-by-Step Execution**

* **Data Analysis**: python main.py --phase analysis
* **Preprocessing**: python main.py --phase preprocessing
* **Feature Extraction**: python main.py --phase features
* **ML Training**: python main.py --phase ml\_training
* **DL Training**: python main.py --phase dl\_training
* **Evaluation**: python main.py --phase evaluation
* **Visualization**: python main.py --phase visualize

**Advanced Usage**

* Specific categories: python main.py --phase all --categories 10 20
* Verbose logging: python main.py --phase ml\_training --verbose

## 6. Model Implementation

**Machine Learning Models**

* **Logistic Regression** (liblinear solver, C=1.0, max\_iter=1000)
* **Random Forest** (100 trees, min\_samples\_split=2)
* **XGBoost** (n\_estimators=100, lr=0.1, max\_depth=6)

**Deep Learning Model**

**BiLSTM Architecture**

* Input: TF-IDF (~10K dims) / SBERT (384 dims)
* Hidden: Dense (512 → 256 → 128) + Dropout(0.3)
* Output: Softmax over categories
* Optimizer: Adam (lr=0.001), EarlyStopping & ReduceLROnPlateau

**Feature Extraction**

* **TF-IDF**: Max features=10k, n-grams=(1,2)
* **SBERT**: Model = all-MiniLM-L6-v2, dim=384

## 7. Evaluation System

**Metrics**

* Accuracy, Precision, Recall, F1-score
* **Top-K Accuracy**: Top-1, Top-3, Top-5
* Training & Inference Time

**Outputs**

* Classification reports
* Confusion matrices
* CSV performance logs
* Visualizations: line, bar, radar plots

## 8. Visualization & Analysis

* **ML Results**: Line/Bar/Radar plots in results/ml/
* **DL Results**: Training history, radar plots in results/dl/
* **Overall Comparison**: ML vs DL trends in results/overall/

## 9. Results & Performance

* **ML Models**: 65–85% accuracy, faster training, good baseline
* **DL Models**: 70–88% accuracy, better with SBERT features
* **Top-5 Accuracy**: 90–99% across models
* **Category Scaling**: Accuracy drops as categories increase

**Recommendations**

* Use **ML models** for speed & interpretability
* Use **DL models** for maximum accuracy & complex tasks
* Use **SBERT embeddings** for production deployments

## 10. Advanced Usage

* **Custom Categories**: --categories 15 25
* **Hyperparameter Tuning**: Edit src/config.py
* **Add New Models**: Extend ml\_models.py or dl\_models.py
* **Custom Visualizations**: Use ModelEvaluator class

**Troubleshooting**

* Memory errors → reduce DL batch size
* CUDA errors → verify GPU drivers
* File not found → check config paths
* Import errors → ensure venv is activated

## 11. Conclusion

* This **Web Services Classification System** offers:
* Flexibility: ML & DL options
* Scalability: Handles multiple category sizes
* Automation: End-to-end pipeline execution
* Analysis: Rich visualizations for insights
* Reproducibility: Consistent results with logging & seeding