TASK 3: Intent Classification

🎯 Task Overview

Intent Classification performs **multi-class classification** on spoken voice commands to determine user intent. This is critical for voice assistants and smart home devices, enabling them to understand what the user wants to do.

Task Characteristics:

- **Classes:** 20+ intent categories (voice commands)
- **Model:** WavLM-base-plus + SVM
- **Feature Extraction:** 768-dimensional embeddings
- **Dimensionality Reduction:** PCA (768 → 200)
- **Dataset:** SLURP (Spoken Language Understanding)
- **Developer:** Sahasra Ganji
- **Processing Time:** ~2-3 seconds per audio

📋 Intent Categories

Intent represents the user's goal in speaking. Examples:

```
Examples of 20+ Intents:

Smart Home Control:
├── turn_on_lights
├── turn_off_lights
├── set_brightness
├── change_color
└── set_temperature

Entertainment:
├── play_music
├── pause_media
├── skip_track
├── play_podcast
└── play_movie

Information:
├── get_weather
├── get_news
├── set_alarm
├── set_reminder
└── get_traffic

General Commands:
├── stop
├── help
```

```
├── cancel
└── repeat_last
```

Use Cases:

- Voice assistant command routing
- Smart speaker control
- IoT device management
- Hands-free operation
- Accessibility applications

🏗 Technical Architecture

Model Stack

```
Raw Audio (Voice Command)
        ↓
Audio Preprocessing (normalize, resample to 16kHz)
        ↓
WavLM-base-plus Model (Feature Extraction)
        ↓
Fixed Embedding (768-dimensional vector)
        ↓
StandardScaler (Normalize)
        ↓
PCA Dimensionality Reduction (768 → 200 dims)
        ↓
SVM Classifier (One-vs-Rest for Multi-class)
        ↓
Predicted Intent Label + Probability Scores
```

Why SVM for Intent?

Aspect	Reason
Multi-class (20 classes)	One-vs-Rest strategy effective
Non-linear	RBF kernel captures intent patterns
Efficient	Fast training/inference on 200-dim vectors
Robust	Works well with balanced intent distribution

📊 SLURP Dataset

Dataset Overview

Property	Value

Property	Value
Full Name	Spoken Language Understanding, Real and Personalized
Size	~63,000 utterances
Total Duration	~150 hours
Unique Speakers	1,300+
Language	English
Domains	Smart home, entertainment, productivity, general
Intents	20+ action categories
Entities	Device names, colors, times, etc.
Recording	Mobile devices, various conditions

Dataset Characteristics

Intent Distribution:

```
Common Intents:          | Rare Intents:
├── activate (music)     | ├── request_logs
├── deactivate           | ├── request_definition
├── turn_off (lights)    | ├── request_password_reset
├── turn_on (lights)     | ├── request_info
├── set_scene_brightness | └── alarm_info
└── audio_volume         |
```

Balanced vs Imbalanced:

- Most intents: 2,000-3,000 samples
- Some rare intents: 100-500 samples
- Requires weighted class strategy in SVM

Data Split

```
Training:   45,000 samples (71%)
Validation:  8,000 samples (13%)
Test:       10,000 samples (16%)
```

🔄 Intent Classification Pipeline

Stage 1: Audio Preprocessing

Input: Voice command utterances (1-15 seconds)

Processing:

1. Load audio at 16kHz
2. Normalize amplitude to [-1, 1]
3. Remove silence at beginning/end
4. Check duration (0.5-30 seconds)
5. Create label mapping:

```
intent_map = {
  'activate': 0,
  'deactivate': 1,
  'turn_off': 2,
  ...
  'request_info': 19
}
```

Output: Normalized audio + integer labels [0, 19]

Stage 2: Feature Extraction

WavLM-base-plus Processing:

```
voice_command → WavLM-base-plus
               → hidden_states [T, 768]
               → mean_pooling
               → embedding [768]
```

Where:

- T = sequence length (depends on audio duration)
- embedding ∈ ℝ^768

Why 768 dimensions?

- WavLM-base model size = 768
- Captures linguistic + acoustic information
- Intent requires understanding words and meanings
- 768 > 200 (emotion) because more classes (20 vs 6)

Stage 3: Scaling & Normalization

StandardScaler:

```
X_scaled = (X - mean_train) / std_train

For each of 768 dimensions:
- mean_train = compute from training embeddings
```

```
- std_train = compute from training embeddings
- Apply same transformation to val/test
```

Why Important:

- SVM RBF kernel is distance-based
- Prevents high-magnitude features dominating
- Numerical stability

Stage 4: Dimensionality Reduction

PCA Configuration:

- Input: 768 dimensions
- Output: 200 dimensions
- Variance preserved: ~94%
- Explained variance: Top 200 eigenvalues explain 94%+ of variance

Mathematical Process:

```
1. Compute covariance: Σ = (1/N) * X^T * X
2. Eigendecomposition: Σ = V * Λ * V^T
3. Sort by eigenvalues: λ1 ≥ λ2 ≥ ... ≥ λ768
4. Select top 200: W = V[:, :200]
5. Transform: X_reduced = X_scaled @ W
```

Why 200 dimensions?

- 20 classes need good separation
- 200 > 200 (gender) because more complex
- Still ~74% dimension reduction
- Prevents overfitting

Stage 5: SVM Training (One-vs-Rest)

Training Strategy for 20 Classes:

```
One-vs-Rest (OvR):
For each intent i = 1 to 20:
  1. Create binary problem:
     - Label = 1 if intent is i, else 0
  2. Train SVM: SVM_i on full training data
  3. Get decision function: f_i(x)

Prediction for new sample x:
  scores = [f_1(x), f_2(x), ..., f_20(x)]
  predicted_intent = argmax(scores)
```

SVM Hyperparameters:

```
SVC(
    kernel='rbf',            # Radial Basis Function
    C=1.0,                  # Regularization
    gamma='scale',          # 1/(n_features*var(X))
    probability=True,       # Enable predict_proba()
    class_weight='balanced' # Handle class imbalance
)
```

Why class_weight='balanced'?

- Rare intents (100 samples) vs common (3,000 samples)
- Without balancing, model ignores rare intents
- Balanced weights: w_i = n_samples / (n_classes * n_samples_i)

Mathematical SVM Formula:

```
minimize: (1/2)||w||² + C*Σ(ξ_i)

Subject to:
y_i * (w^T * ϕ(x_i) + b) ≥ 1 - ξ_i

RBF Kernel: K(x, x') = exp(-γ ||x - x'||²)
```

Stage 6: Cross-Validation

5-Fold Stratified Cross-Validation:

```
For fold = 1 to 5:
  Train:      folds 1-4 (71.2% ≈ 45,000 samples)
  Validation: fold 5 (14.3% ≈ 9,000 samples)
  Evaluate: Compute accuracy, F1, per-class metrics

Final Metrics = mean across all 5 folds
```

Stratification Ensures:

- Each fold has same intent distribution
- Rare intents represented in each fold
- Robust performance estimation

🎯 Inference Workflow

Real-time Prediction

```
User Says Voice Command
        ↓
Frontend captures audio (WebM format)
        ↓
POST to /intent_predict endpoint
        ↓
Backend receives audio file
        ↓
IntentInferenceService.predict_intent(audio_path)
  1. Load audio at 16kHz
  2. Extract 768-dim embedding (WavLM-base-plus)
  3. Scale using training scaler
  4. Apply PCA (768 → 200)
  5. SVM predict_proba → get probabilities
  6. Identify top-k intents
        ↓
Return JSON:
{
  "label": "turn_off",
  "probabilities": {
    "turn_off": 0.78,
    "deactivate": 0.12,
    "audio_volume": 0.06,
    ...
  },
  "top_k": [
    {"intent": "turn_off", "confidence": 0.78},
    {"intent": "deactivate", "confidence": 0.12},
    {"intent": "audio_volume", "confidence": 0.06}
  ]
}
        ↓
Frontend displays recognized intent + alternatives
```

Code Implementation

```
class IntentInferenceService:
    def __init__(self):
        self.extractor = load_feature_extractor('microsoft/wavlm-base-plus')
        self.classifier = joblib.load('intent_classifier.pkl')
        self.scaler = joblib.load('intent_scaler.pkl')
        self.pca = joblib.load('intent_pca.pkl')
        self.encoder = joblib.load('intent_label_encoder.pkl')  # Dict or
LabelEncoder

    def predict_intent(self, audio_path, top_k=3):
        # Extract embedding
        embedding = self.extractor.extract_from_file(audio_path)  # [768]

        # Scale
```

```
        embedding_scaled = self.scaler.transform([embedding])  # [1, 768]

        # PCA reduce
        embedding_reduced = self.pca.transform(embedding_scaled)  # [1, 200]

        # Predict probabilities (OvR combines into probabilities)
        probabilities = self.classifier.predict_proba(embedding_reduced)[0]  #
[20]

        # Get top-k predictions
        top_k_idx = np.argsort(probabilities)[::-1][:top_k]
        top_k_intents = []
        for idx in top_k_idx:
            intent_name = self.encoder.classes_[idx]
            confidence = probabilities[idx]
            top_k_intents.append({"intent": intent_name, "confidence":
float(confidence)})

        return {
            "label": top_k_intents[0]["intent"],
            "confidence": top_k_intents[0]["confidence"],
            "top_k": top_k_intents
        }
```

📈 Expected Performance

Multi-class Accuracy

Estimated Overall Accuracy: 85-89%

Per-Intent Performance (Sample):

```
Intent           | Precision | Recall | F1-Score | Support
-----------------|-----------|--------|----------|--------
turn_off         | 0.92      | 0.88   | 0.90     | 2800
turn_on          | 0.90      | 0.89   | 0.90     | 2750
activate_music   | 0.87      | 0.84   | 0.85     | 2600
set_brightness   | 0.84      | 0.86   | 0.85     | 2450
deactivate       | 0.81      | 0.79   | 0.80     | 2200
...              | ...       | ...    | ...      | ...
request_logs     | 0.68      | 0.65   | 0.66     | 120
request_info     | 0.72      | 0.70   | 0.71     | 180
-----------------|-----------|--------|----------|--------
Macro Average    | 0.83      | 0.81   | 0.82     | 50000
```

Why Not 95%+ Like Gender?

1. **More Complex Task:** 20 classes vs 2
2. **Semantic Understanding Needed:** Not just acoustics

3. **Class Imbalance:** Rare intents harder to learn
4. **Acoustic Similarity:** Some intents sound similar
5. **Ambiguity:** Same phrase could mean multiple things

🛠 File Locations

Component	File Location
Inference Service	`backend/services/intent.py`
Web Endpoint	`backend/app/app.py` (route: `/intent_predict`)
HTML Template	`backend/app/templates/intent.html`
Training Script	`ml_models/scripts/train_intent_model.py`
Model Artifacts	`ml_models/models/intent_*.pkl`

Trained Model Files

```
ml_models/models/
├── intent_classifier.pkl       # SVM model (OvR for 20 classes)
├── intent_scaler.pkl           # StandardScaler
├── intent_label_encoder.pkl    # Dict/LabelEncoder for 20 intents
└── intent_pca.pkl              # PCA transformer (768→200)
```

🔍 Error Analysis

Common Misclassifications

Confused Pair	Reason	Example
turn_on ↔ activate	Similar wording	"turn on lights" vs "activate lights"
set_brightness ↔ audio_volume	Both numeric control	Setting light brightness vs volume
deactivate ↔ turn_off	Synonyms	User phrasing differences
get_weather ↔ get_news	Both informational	Request types

Mitigation Strategies:

1. Use entity recognition alongside intent
2. Context awareness (previous commands)
3. Ensemble multiple models
4. User correction feedback

📚 Interview Talking Points

Key Concepts

1. **One-vs-Rest Strategy:**
   - Train 20 independent binary classifiers
   - Each classifier: "Is this intent i? Yes/No"
   - Final prediction: argmax of decision functions
   - Pros: Simple, parallelizable; Cons: Not optimal for imbalanced

2. **Class Imbalance Handling:**
   - Rare intents (100 samples) vs common (3000 samples)
   - Solution: `class_weight='balanced'`
   - Formula: w_i = n_samples / (n_classes * n_class_i_samples)
   - Gives more penalty for misclassifying rare classes

3. **Why 200 dimensions after PCA?**
   - More complex than gender (20 vs 2 classes)
   - Still ~74% reduction from 768
   - Trade-off: speed vs accuracy
   - Could use 300-400 for higher accuracy

4. **Performance Trade-offs:**
   - Accuracy 85-89% is good for 20-class problem
   - Audio has inherent ambiguity
   - Could improve with:
     - Larger models (HuBERT-large)
     - Fine-tuning instead of transfer learning
     - Ensemble methods
     - Multi-modal (add text transcription)

🚀 Real-World Integration

Voice Assistant Pipeline

```
User speaks: "Turn off the bedroom lights"
        ↓
Speech-to-Text (ASR) → "turn off the bedroom lights"
        ↓
Intent Classification → "turn_off"
        ↓
Entity Extraction → room: "bedroom", device: "lights"
        ↓
Action Execution → Send OFF command to bedroom lights
        ↓
Text-to-Speech → "Turning off bedroom lights"
        ↓
Speak response to user
```

Our Task: Step 3 (Intent Classification)

Why Important?

- **Routing:** Send command to appropriate module
- **Context:** Understand user goal
- **Confirmation:** "You want to turn off lights, right?"
- **Analytics:** Understand user preferences

📝 Summary

Intent Classification demonstrates multi-class ML:

- ☑ 20-class problem (vs 2 for gender, 6 for emotion)
- ☑ Class imbalance handling
- ☑ One-vs-Rest SVM strategy
- ☑ Practical voice assistant application
- ☑ 85-89% accuracy

Interview Confidence:

- Understand OvR strategy for multi-class
- Explain class_weight='balanced' rationale
- Know performance metrics for 20-class
- Can discuss real-world voice assistant pipeline
- Understand why accuracy is lower than simpler tasks

Created: December 2024
Developer: Sahasra Ganji
Expected Accuracy: 85-89% (20 classes)
Status: Production Ready ☑