Speech Recognition

Assignment 3: Continuous Speech Recognition Based on GMM-HMM

Student Name: 赵卓冰

Student Number: 2252750

Major: 软件工程

Teacher: 沈莹

2024.11.22

1. Experiment Objectives

1. Master and reinforce knowledge of speech recognition, including the use of GMM and HMM, and understand the GMM-HMM continuous speech recognition algorithm.
2. Familiarize with Huawei's MindSpore platform and Python 3.

2. Experiment Environment

- **Programming Language**: Python
- **Libraries Used**:
  - `numpy`: Data processing
  - `scipy`: Audio signal reading
  - `python_speech_features`: MFCC feature extraction
  - `hmmlearn`: GMM-HMM model implementation
  - `pickle`: Model saving and loading
- **Running Environment**: Python 3.7, MindSpore 1.1.1

3. Experiment Steps

3.1. Data Preparation and Library Import

1. **Training Data Path**:
   - Training speech data is stored in `datas/train/speech`.
   - Label file path is `datas/labels/trainprompts_m`.
2. **Testing Data Path**:
   - Testing speech data is stored in `datas/test/speech`.
3. **Import Libraries**:

```
pip install python_speech_features
pip install hmmlearn
pip install numpy
```

3.2. Feature Extraction

The `wav2mfcc` function is designed to convert `.wav` audio files into MFCC (Mel Frequency Cepstral Coefficients) features:

- Load the audio file.
- Extract short-time MFCC feature sequences to represent the temporal dynamics of speech.

```
def wav2mfcc(labels, data_paths):
    trng_data = {}
    for label, data_path in zip(labels, data_paths):
        rate, sig = wvf.read(data_path)
        mfcc_feat = mfcc(sig, rate)
        trng_data[label] = [mfcc_feat]
    return trng_data
```

3.3. Defining the GMM-HMM Model

1. **Model Configuration**:
   - Each class's HMM uses 2 hidden states (`n_components=2`), and each state is modeled with 2 Gaussian mixture components (`n_mix=2`).

```
def obtain_config(labels):
    conf = {}
    for label in labels:
        conf[label] = {"n_components": 2, "n_mix": 2}
    return conf
```

2. **Model Construction**:
    - Train an independent GMM-HMM model for the training data of each class.
    - Use `hmmlearn.hmm.GMMHMM` for model initialization and training.

```
def get_hmm_gmm(trng_datas, GMM_configs,
model_path="hmm_gmm_model.pkl", from_file=False):
    hmm_gmm = {}
    if not from_file:
        for label, trng_data in trng_datas.items():
            GMM_config = GMM_configs[label]
            hmm_gmm[label] = GMMHMM(
                n_components=GMM_config["n_components"],
                n_mix=GMM_config["n_mix"]
            )
            hmm_gmm[label].fit(np.vstack(trng_data))
        pickle.dump(hmm_gmm, open(model_path, "wb"))
    else:
        hmm_gmm = pickle.load(open(model_path, "rb"))
    return hmm_gmm
```

3.4. Model Training

- Load training data paths and corresponding label files.
- Extract MFCC features from the training data.
- Initialize and train the GMM-HMM model.

```
def train(train_data_path, label_path, model_path):
    with open(label_path) as f:
        labels = f.readlines()
    data_paths = [train_data_path + '/' + line.split()[0] + '.wav' for
line in labels]
    labels = [' '.join(line.split()[1:]).strip() for line in labels]
    train_datas = wav2mfcc(labels, data_paths)
    GMM_configs = obtain_config(labels)
    hmm_gmm = get_hmm_gmm(train_datas, GMM_configs, model_path)
    return hmm_gmm
```

3.4.1. Training Results

- ```
  hmm_gmm = train(train_data_path,label_path,model_path)
  ```

```
关闭 风扇
打开 阀门
关 灯 九 分钟
关掉 灯 七 秒
关 阀门 五 秒
关掉 阀门 二 小时
打开 风扇 三 秒
关掉 灯 七 秒
关闭 风扇
打开 灯 五 分钟
关 灯 五 分钟
```

3.5. Model Testing and Prediction

1. **Feature Extraction and Score Calculation**:
   - Extract MFCC features from test audio.
   - Compute the log-likelihood score of the test feature sequence using GMM-HMM models.
2. **Classification Prediction**:
   - Compare scores for all classes and select the class with the highest score as the prediction.

```python
def test_file(test_file, hmm_gmm):
    rate, sig = wvf.read(test_file)
    mfcc_feat = mfcc(sig, rate)
    pred = {model: hmm_gmm[model].score(mfcc_feat) for model in hmm_gmm}
    return get_nbest(pred, 2), pred

def get_nbest(d, n):
    return heapq.nlargest(n, d, key=lambda k: d[k])
```
```

3.5.1. Testing Results

- ```
  wave_path = os.path.join(test_data_path,"T0001.wav")
  #wave_path = os.path.join(train_data_path,"S0001.wav")
  predicted, probs = predict_label(wave_path,hmm_gmm)
  print("PREDICTED: %s" % predicted[0])
  ```

  ```
  PREDICTED: 关掉 风扇 二 秒
  ```

3.6. Using Features Extracted from the Assignment 1

- First, save the features extracted in the first task into a `features.npy` file:

```
    file_path = 'input.wav'
    signal, sr = librosa.load(file_path, sr=None)
    frame_length = int(0.025 * sr)
    frame_step = int(0.01 * sr)
    features = mfcc(signal, sr, frame_length, frame_step)
    output_file = 'features.npy'
    np.save(output_file, features)
    print(f"Features saved to {output_file}")
```

- Use the extracted features for prediction:

```
feature_file_path = 'features.npy'
predicted, probs = predict_label(feature_file_path, hmm_gmm)
print("PREDICTED: %s" % predicted[0])
```

3.6.1. Prediction Results

```
# 调用预测
predicted, probs = predict_label(feature_file_path, hmm_gmm)

# 输出结果
print("PREDICTED: %s" % predicted[0])
```

```
PREDICTED: 关 灯 五 分钟
```
```

4. Analysis and Discussion

4.1. Analysis of Differences Between Recognition Results and Actual Speech

Despite the effectiveness of the GMM-HMM model in this experiment, the recognition results may still differ from actual speech. This discrepancy can be attributed to several factors, which can be analyzed from the perspectives of the dataset, testing conditions, model accuracy, and other aspects:

4.1.1. Dataset Issues

- **Distribution Mismatch Between Training and Test Data**: The training and test datasets may come from different distributions, meaning that the test data may contain speech patterns or accents that were not present in the training data. This can lead to poor generalization of the model. For example, if the training data is mostly from speakers with a certain accent or from a specific region, while the test data contains speech from speakers with different accents, the model's performance may suffer.
- **Noisy Annotations**: Inaccurate or noisy annotations in the training data can also affect the model's performance. If the labels for the speech recordings are incorrect, the model learns to recognize incorrect patterns, leading to poor recognition accuracy. This is particularly important in speech recognition, where data accuracy and quality are essential for successful model training.

4.1.2. Feature Extraction Issues

- **Precision of Feature Extraction**: MFCC is a widely used feature in traditional speech recognition tasks, but it is not always the most robust in capturing all nuances of speech. While MFCC captures the time-frequency characteristics of speech, it may not perform well in noisy environments or when faced with variations in accent, speaking speed, or phonetic details. In such cases, the feature extraction may not capture all the relevant information, leading to lower recognition accuracy.
- **Feature Selection and Dimensionality**: MFCC features are relatively low-dimensional and may not capture all the complexities of speech. If more sophisticated features are not used, the model may fail to distinguish subtle differences between speech patterns, leading to a higher rate of misclassification.

4.2. Advantages

1. **Ease of Implementation**:
    - The GMM-HMM model combines the advantages of temporal modeling (HMM) and probabilistic distribution modeling (GMM).
2. **Systematic Design**:
    - The experiment follows a modular approach, covering data loading, feature extraction, model training, and testing.

4.3. Limitations

1. **Limited Model Performance**:
    - GMM-HMM has limited capacity to represent high-dimensional features, which may constrain recognition accuracy.
2. **Noise Sensitivity**:
    - Recognition performance can degrade significantly if the test audio contains background noise.
3. **Dependence on Annotated Data**:
    - The training process requires a large amount of labeled data and is computationally intensive.

5. Suggestions for Improvement

1. **Model Enhancement**:
    - Replace GMM with a deep neural network (DNN) to build a DNN-HMM system.
    - Alternatively, adopt end-to-end deep learning models, such as RNN or Transformer.
2. **Feature Processing**:
    - Use more robust features, such as log-mel spectrograms or spectrogram features.
3. **Noise Reduction**:
    - Apply noise reduction techniques before feature extraction.

6. Conclusions

This experiment successfully implemented the GMM-HMM model, demonstrating its effectiveness in speech recognition tasks. The results show that GMM-HMM can model and classify speech signals but may struggle with complex features and noisy inputs. Future work can incorporate deep learning techniques to further enhance the accuracy and robustness of speech recognition systems.