refs

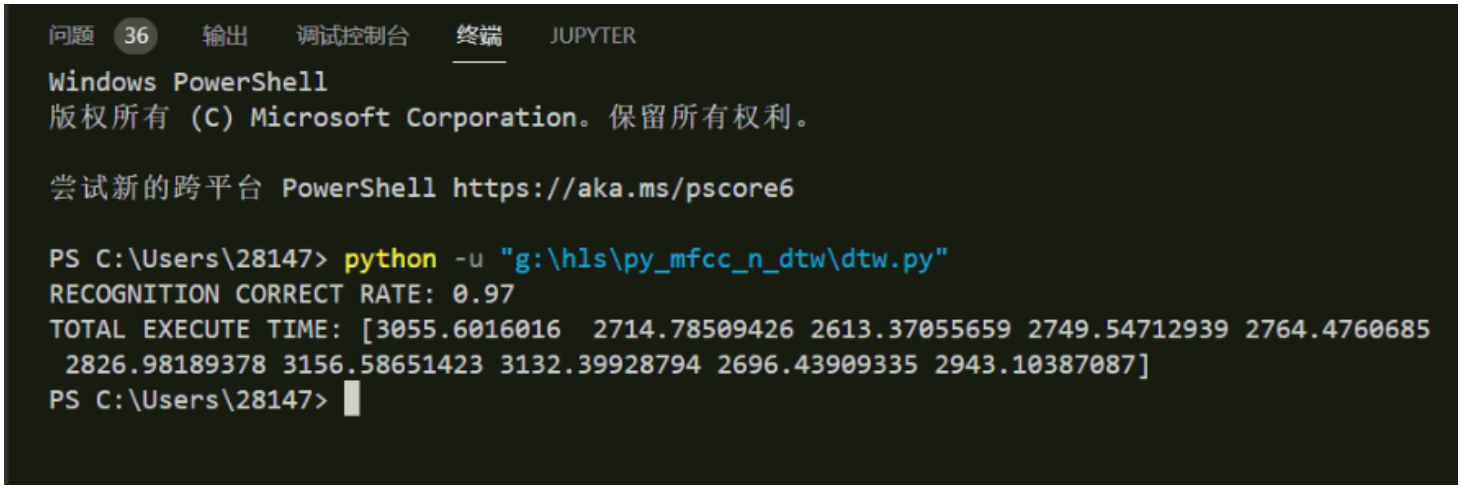
DTW core code from https://github.com/Linzecong/MFCC-DTW

Audio dataset from https://www.kaggle.com/datasets/sripaadsrinivasan/audio-mnist, using python librosa to extract MFCC feature

Python FastDTW

Accuarcy

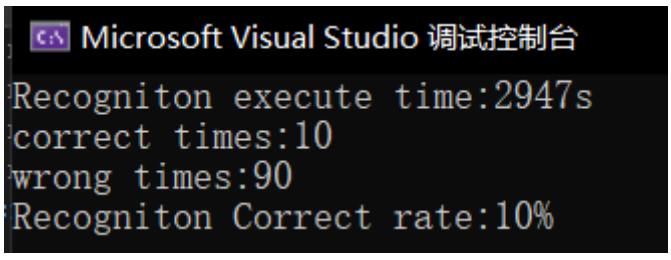
97%



C++ DTW

Accuarcy

Only 10%😅



Utilization

CPU Details

```
                    vector<vector<float> >  match_mfcc_feat;
1 (0.00%)           for (int i = 0; i < match_csv_content.size(); i++)
                    {
3 (0.01%)               if((match_csv_content[i].size() % 13) != 0){
                            cout << "WRONG READ!" << "MFCC Length is: "<< match_csv_content[i].size() << endl;
                            /* for (int j = 0; j < match_csv_content[i].size(); j++)
                            {
                                cout << match_csv_content[i][j] << endl;
                            } */
                            break;
                        }
                        else{
                            float distance = 0;
                            float normal_distance = 0;
3372 (11.55%)               match_mfcc_feat = rebuilt_mfcc_feat(match_csv_content[i]);
17208 (58.95%)              distance = ComputeDTW(query_mfcc_feat,match_mfcc_feat);
                            normal_dist = distance / (query_mfcc_feat.size() + match_mfcc_feat.size());
                            eud_distance.push_back(normal_dist);
                        }
                    }
                    //eud_distance = standard_normal(eud_distance);   //no standard
                    auto    min_pos   = std::min_element(eud_distance.begin(), eud_distance.end());
                    *min_idx = (min_pos - eud_distance.begin());
                    *min_dist = *min_pos;
```

```
                                                                                
                                                                                
    59 (0.20%)  float DTW_Function::Distance(float *ps1, float *ps2, int k1, int k2){
                    int i=0;
                    float sum=0;
    42 (0.14%)      for(i=0;i<MFCC_P;i++)
 15593 (53.42%)         sum+=(1+MFCC_Pf/2*(float)sin(PI*i/MFCC_Pf))*(ps1[k1+i]-ps2[k2+i])*(ps1[k1+i]-ps2[k2+i]);
                
    67 (0.23%)      return sum;
    35 (0.12%)  }
                
```

Function Distance used 53.42%.

Memory Details