**Speech Emotion Recognization Development Handbook**

Morindaz.Mao

**1 Dataset Description:**

**1.1 Feature Obtain**



**Figure 1.** The process of obtaining features and modeling

**1.2 Dataset**

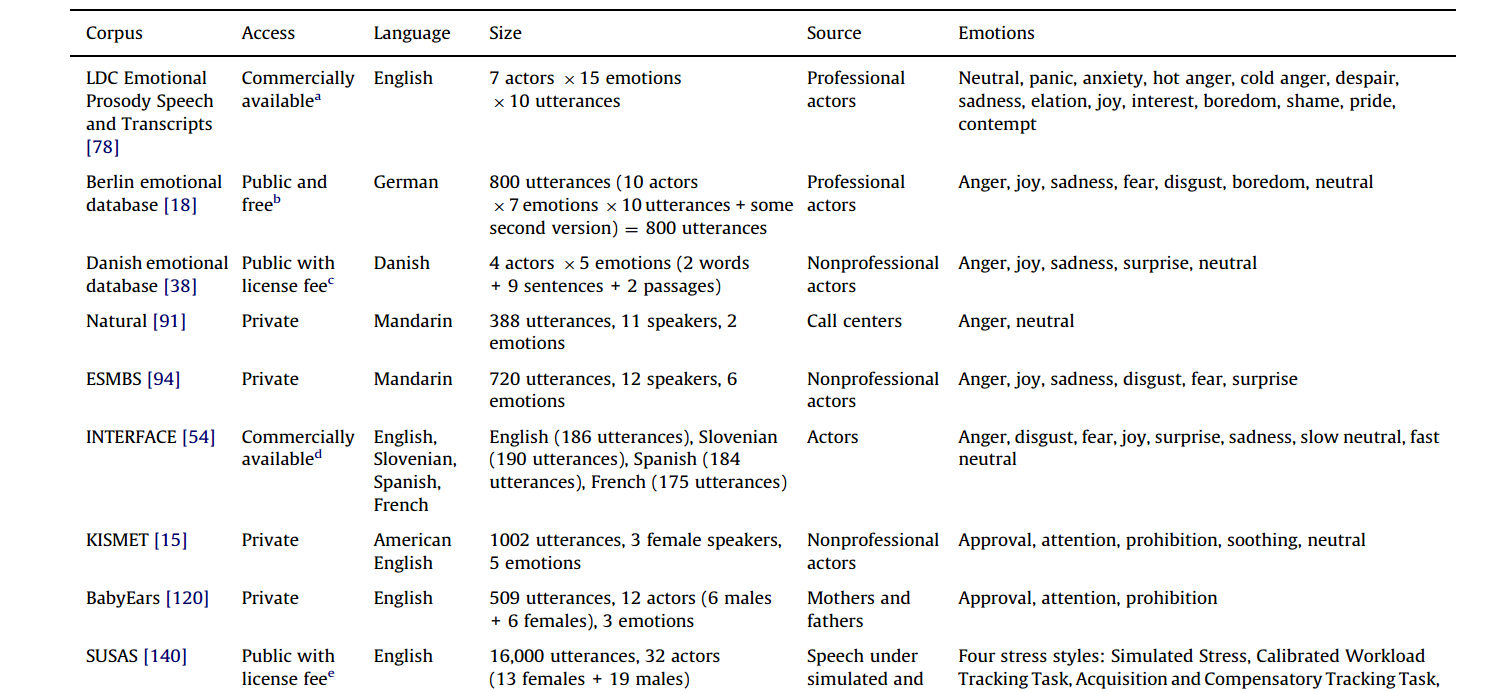
**The formal dataset with noise**

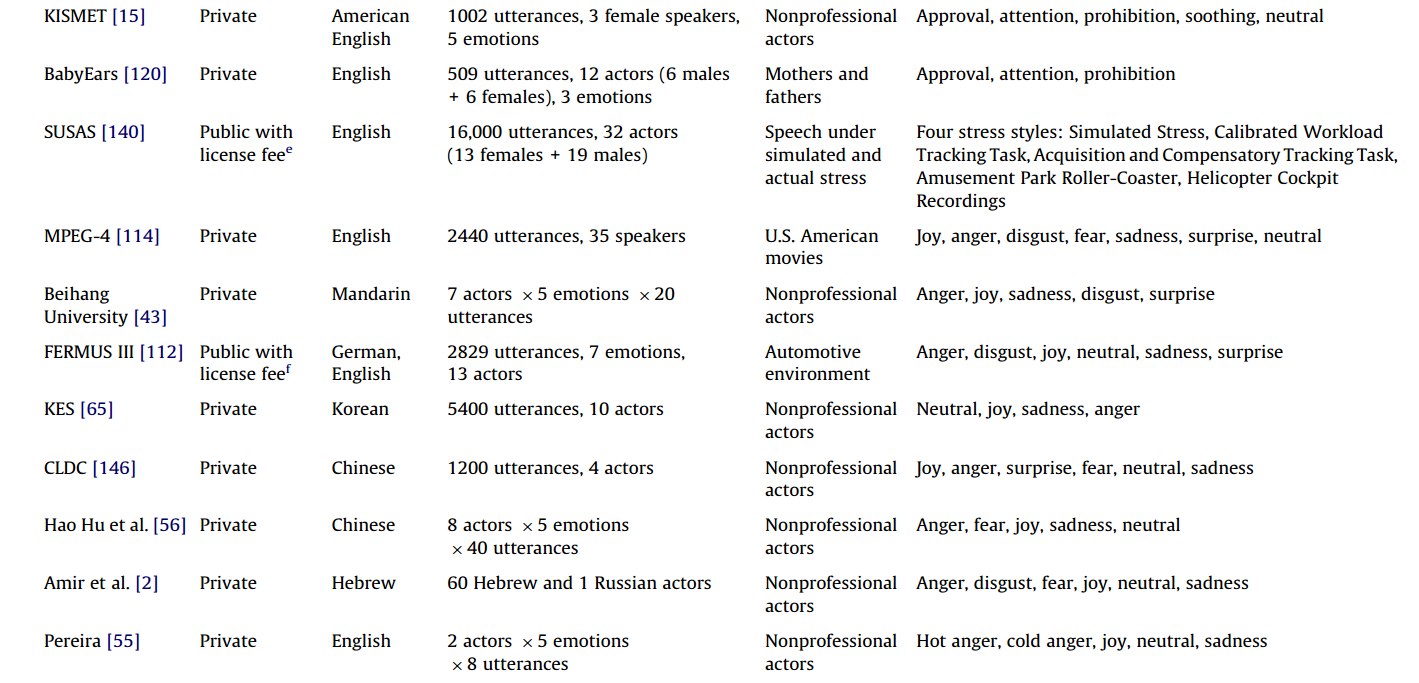
**Table 1.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Formal** | **Selected** | **Train 1** | **Train 2** | **Test** |
|  |  |  |  |  |

**1.3 Other Dataset**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 情感语音数据库 | 语言 | 说话人 | 容量/条 | 使用权 | EGG | 情感类型 |
| CASIA[12] | 汉语 | 4名非专业人员 | 9 600 | 非公开 | 否 | 高兴,悲伤,生气,惊吓,中性 |
| ACCorpus\_SR[2, 13] | 汉语 | 50名非专业人员 | 50 000 | 非公开 | 否 | 高兴,生气,恐惧,悲伤,中性 |
| ESMBS[14] | 汉语和缅甸语 | 6名中国人和6名缅甸人 | 720 | 非公开 | 否 | 高兴,生气,伤心,嫌恶,恐惧,惊奇 |
| NATURAL[15] | 汉语 | 11名人员呼叫中心对话 | 388 | 非公开 | 否 | 生气,中性 |
| Beihang University[16] | 汉语 | 7名非专业人员 | 2 100 | 部分公开 | 否 | 生气,高兴,伤心,沮丧,惊奇 |
| CLDC[17] | 汉语 | 4名非专业人员 | 1 200 | 非公开 | 否 | 高兴,生气,惊奇,恐惧,中性,伤心 |
| Hu等[18] | 汉语 | 8名非专业人员 | 1 600 | 非公开 | 否 | 高兴,生气,伤心,恐惧,中性 |
| Berlin emotional database[19] | 德语 | 10名专业人员 | 800 | 公开 | 是 | 高兴,生气,伤心,恐惧,嫌恶,厌烦,中性 |
| FERMUS Ⅲ[20] | 德语和英语 | 13名人员;汽车噪音环境 | 2 829 | 公开 | 否 | 高兴,生气,伤心,嫌恶,惊奇,中性 |
| Danish emotional database[21] | 丹麦语 | 4名专业人员 | 500多 | 公开 | 否 | 高兴,生气,伤心,惊奇,中性 |
| SUSAS[22] | 英语 | 32名非专业人员 | 16 000 | 公开 | 否 | 生气,恐惧,沮丧,焦虑,害怕等情绪 |
| KISMET[23] | 英语和美语 | 3名非专业人员 | 1 002 | 非公开 | 否 | 赞成,关心,禁止,吮吸,中性 |
| BabyEars[24] | 英语 | 12名非专业人员 | 509 | 非公开 | 否 | 赞成,关心,禁止 |





**2 Preprocessing**

**Table 2.1.** The Rule of classifying the 54 sub emotions into 6 categories

|  |  |  |
| --- | --- | --- |
| **Label** | **Emotion** | **SubClasses** |
| **0** | sadness | 6, 9, 17, 21, 24, 27, 35, 37, 42, 43, 45, 47, 50, 52 |
| **1** | fear | 7, 14, 22, 25, 41 |
| **2** | surprise | 5, 32, 48 |
| **3** | joy | 0, 1, 8, 11, 18, 26, 28, 33, 34, 36, 40, 44, 46, 51, 53 |
| **4** | anger | 2, 3, 10, 13, 15, 16, 29 |
| **5** | disgust | 4, 12, 19, 20, 23, 30, 31, 38, 39, 49 |

**2.1 Using Confusion Matrix to classify**

**2.2**

**2 Experiments**

**2.1 Model 1: SVM**

2.1.1 Baseline

2.1.2 GridSearch for SVM

**2.2 Model 2: GBDT**

2.2.1 Baseline

2.2.2 GridSearch for GBDT

**2.3 Model 3: XGBoost**

2.3.1 Baseline

2.3.2 GridSearch for XGBoost

2.3.3 HyperOpt Parameters Tune

2.4 Model4 Deep Learning Methods

Deep learning model uses keras

2.4.1 Baseline

This model uses

Results for six categories

Use the selected dataset

|  |  |  |
| --- | --- | --- |
| Model | Features | Accuracy |
| Dense(10, input\_dim=30, activation='relu')  Dense(6, activation='softmax') | 30(Feature) | Baseline: 33.19% |
| Dense(10, input\_dim=30, activation='relu')  Dense(8, activation='relu')  Dense(6, activation='softmax') | 30(Feature) | 28.49% |
| model.add(Dense(10,input\_dim=980,activation='relu')) model.add(Dense(10)) model.add(Dense(6, activation='softmax')) | 0:980 | 29.85% 、 |
| model.add(Dense(10,input\_dim=980,activation='relu')) model.add(Dense(6, activation='softmax')) | 0:980 | 29.28% |
| model.add(Dense(10,input\_dim=980, activation='relu')) model.add(Dense(10)) model.add(Dense(10)) model.add(Dense(6, activation='softmax')) | 0:980 | 28.86% |
| model.add(Dense(564, activation='relu', input\_dim=980)) model.add(Dropout(0.5)) model.add(Dense(82, activation='relu')) model.add(Dropout(0.5) model.add(Dense(6, activation='softmax')) | 0:980 | 37.09% |
| model.add(Dense(464, activation='relu', input\_dim=980)) model.add(Dropout(0.5)) model.add(Dense(82, activation='relu')) model.add(Dropout(0.5)) model.add(Dense(6, activation='softmax')) | 0:980 | 37.94% |
| model.add(Dense(464, activation='relu', input\_dim=980)) model.add(Dropout(0.5)) model.add(Dense(82, activation='relu')) model.add(Dropout(0.3)) model.add(Dense(6, activation='softmax')) | 0:980 | 39.49% |

Result for 54 categories

Use the selected dataset

|  |  |  |
| --- | --- | --- |
| Model | Feature | Accuracy |
| model.add(Dense(664, activation='relu', input\_dim=978)) model.add(Dropout(0.5)) model.add(Dense(192, activation='relu')) model.add(Dropout(0.3)) model.add(Dense(54, activation='softmax')) | 0:978 | acc: 10.58% |
| model.add(Dense(664, activation='relu', input\_dim=978)) model.add(Dropout(0.4)) model.add(Dense(192, activation='relu')) model.add(Dropout(0.3)) model.add(Dense(54, activation='softmax')) |  | 10.44% |
|  |  |  |

Use the combined dataset

|  |  |  |
| --- | --- | --- |
| Model | Feature | Accuracy |
| model.add(Dense(464, activation='relu', input\_dim=978)) model.add(Dropout(0.5)) model.add(Dense(82, activation='relu')) model.add(Dropout(0.5)) model.add(Dense(54, activation='softmax')) |  | 23.41% |
| model.add(Dense(464, activation='relu', input\_dim=978)) model.add(Dense(82, activation='relu')) model.add(Dropout(0.5)) model.add(Dense(54, activation='softmax')) |  | 68.83% |

Train2.csv: the train dataset includes the test dataset

Train3All: the train dataset includes the test dataset and there is some repeat data

Train.csv:

Code

**import** numpy  
**import** pandas  
**from** keras.models **import** Sequential  
**from** keras.layers **import** Dense  
**from** keras.wrappers.scikit\_learn **import** KerasClassifier  
**from** keras.utils **import** np\_utils  
**from** sklearn.model\_selection **import** cross\_val\_score  
**from** sklearn.model\_selection **import** KFold  
**from** sklearn.preprocessing **import** LabelEncoder  
**from** sklearn.pipeline **import** Pipeline  
  
**from** sklearn **import** datasets  
seed = 7  
numpy.random.seed(seed)  
  
encoder = LabelEncoder()  
encoder.fit(y\_train)  
encoded\_Y = encoder.transform(y\_train)  
# convert integers to dummy variables (i.e. one hot encoded)  
dummy\_y = np\_utils.to\_categorical(encoded\_Y)  
  
# define baseline model  
**def baseline\_model**():  
 # create model  
 model = Sequential()  
 model.add(Dense(8, input\_dim=30, activation='relu'))  
 model.add(Dense(6, activation='softmax'))  
 # Compile model  
 model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])  
 **return** model  
  
estimator = KerasClassifier(build\_fn=baseline\_model, epochs=200, batch\_size=5, verbose=0)  
kfold = KFold(n\_splits=10, shuffle=True, random\_state=seed)  
results = cross\_val\_score(estimator, X\_train, dummy\_y, cv=kfold)  
**print**("Baseline: %.2f%% (%.2f%%)" % (results.mean()\*100, results.std()\*100))

Reference:

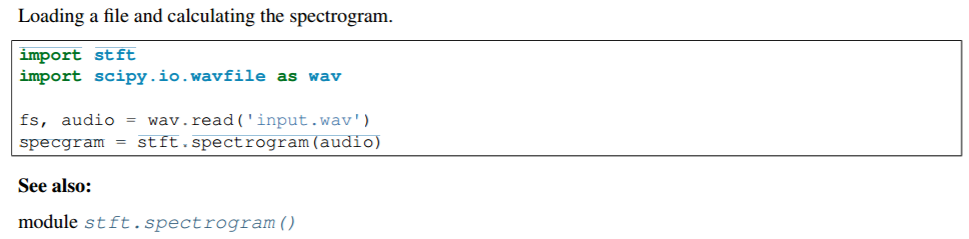
https://machinelearningmastery.com/multi-class-classification-tutorial-keras-deep-learning-library/

2.4.2 An implementation of the essay

Essay: Speech Emotion Recognition using Convolutional and Recurrent Neural Networks

1. Extract STFT features

Installation: pip install stft



1. sd

Reference:

https://media.readthedocs.org/pdf/stft/latest/stft.pdf

This hasndbook records the development process of the SNR.

References  
1、bobo给的 https://github.com/morindaz/Kaggle\_CrowdFlower  
2、整合几种方法的例子：  
http://scikit-learn.org/stable/auto\_examples/ensemble/plot\_feature\_transformation.html#example-ensemble-plot-feature-transformation-py  
3、GBDT使用方法 https://github.com/morindaz/GBDT  
4、GBDT调参 http://chuansong.me/n/296022746725  
5、XGBoost调参  
https://www.dataiku.com/learn/guide/code/python/advanced-xgboost-tuning.html  
https://github.com/bamine/Kaggle-stuff/blob/master/otto/hyperopt\_xgboost.py  
http://blog.csdn.net/qq\_34139222/article/details/60322995  
6、Hyperopt+randomSearch  
https://stats.stackexchange.com/questions/183984/how-to-use-xgboost-cv-with-hyperparameters-optimization

7、<https://media.readthedocs.org/pdf/stft/latest/stft.pdf> :**This is a document of STFT**

8、 Speech Emotion Recognition using Convolutional and Recurrent Neural Networks

9、[**https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/**](https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/)

**10、**[**http://ruder.io/optimizing-gradient-descent/**](http://ruder.io/optimizing-gradient-descent/)

**11、**[**http://deeplearning.net/tutorial/**](http://deeplearning.net/tutorial/)