信也科技杯 FINVOLUTION BIG DATA COMPETITION | 信也 FINVOLUTION

数智创新 声至未来

DEEP IN DIALECTS, FOR FUTURE WAVE

第八届信也科技杯算法大赛

THE 8TH FINVOLUTION DATA SCIENCE COMPETITION

OnTheWay

陈冬雨 刘世欢 陈欣

目录

赛题理解

赛题描述

赛题背景

语音方言距离（相似性）研究，应用于汉语方言ASR转写；

根据方言距离设置一系列核心方言，并构建核心ASR模型，对未知语音进行方言鉴别，找到最相似核心方言与其ASR模型进行转写。故此次赛题的目标即为：语音方言距离识别

数据维度

1. 9种（复赛18种）10万+的point-wise已知方言语音文件
2. 已知方言间的距离矩阵
3. 测试集为 100万的pair-wise语音样本

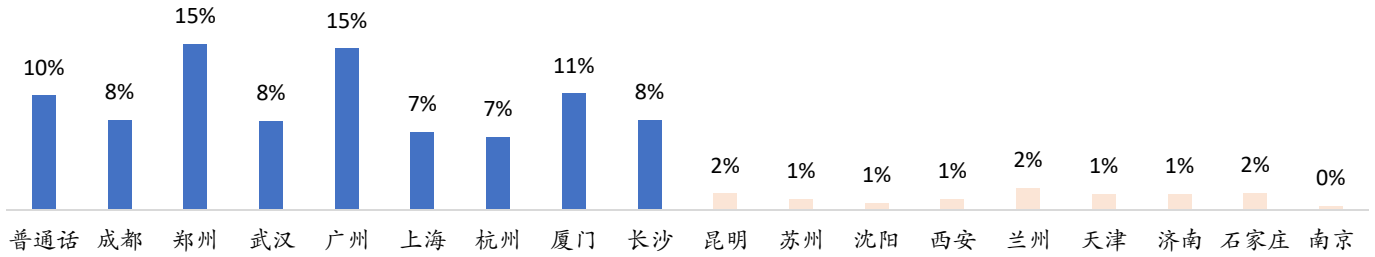
评估指标

预测推理距离和真实距离间的MAE

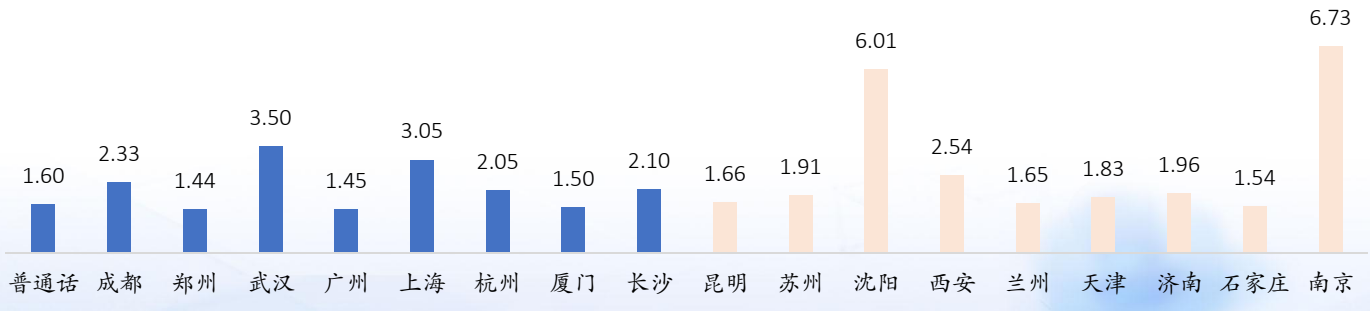
赛题理解

方言描述

方言量级占比分布



方言语音时长中位数



距离矩阵

	北京	成都	郑州	武汉	广州	上海	杭州	厦门	长沙	昆明	苏州	沈阳	西安	兰州	天津	济南	石家庄	南京
北京	0.0	32.1	23.4	34.4	68.7	67.7	57.1	79.2	51.7	32.4	68.3	14.5	33.0	29.7	15.3	15.9	17.7	35.5
成都	32.1	0.0	38.2	24.5	65.8	60.7	49.9	76.6	43.7	31.7	61.7	33.3	38.7	37.7	33.0	33.5	38.2	31.7
郑州	23.4	38.2	0.0	40.1	68.6	70.5	61.8	79.1	55.9	41.0	69.6	32.4	32.4	37.1	32.1	21.6	22.8	44.4
武汉	34.4	24.5	40.1	0.0	66.0	62.6	54.2	77.2	34.0	31.4	62.9	34.3	40.0	40.6	36.2	36.9	40.9	37.1
广州	68.7	65.8	68.6	66.0	0.0	67.1	68.3	71.0	68.4	69.1	63.8	67.3	69.9	73.4	67.8	68.6	69.0	63.9
上海	67.7	60.7	70.5	62.6	67.1	0.0	40.7	78.0	63.9	65.9	23.8	66.4	69.4	67.0	66.8	68.4	69.7	60.0
杭州	57.1	49.9	61.8	54.2	68.3	40.7	0.0	76.2	57.3	56.8	41.8	57.2	60.8	59.6	57.3	59.1	60.5	50.9
厦门	79.2	76.6	79.1	77.2	71.0	78.0	76.2	0.0	77.0	79.1	75.6	79.0	80.3	81.4	79.0	79.3	79.0	76.0
长沙	51.7	43.7	55.9	34.0	68.4	63.9	57.3	77.0	0.0	46.8	63.7	50.5	53.0	51.6	51.8	53.1	54.9	48.6
昆明	32.4	31.7	41.0	31.4	69.1	65.9	56.8	79.1	46.8	0.0	65.9	39.2	45.7	40.2	40.0	36.5	40.7	30.5
苏州	68.3	61.7	69.6	62.9	63.8	23.8	41.8	75.6	63.7	65.9	0.0	67.7	69.7	67.9	67.5	69.0	68.5	60.9
沈阳	14.5	33.3	32.4	34.3	67.3	66.4	57.2	79.0	50.5	39.2	67.7	0.0	34.6	32.6	10.8	24.0	26.4	39.3
西安	33.0	38.7	32.4	40.0	69.9	69.4	60.8	80.3	53.0	45.7	69.7	34.6	0.0	39.8	32.7	29.4	36.3	46.1
兰州	29.7	37.7	37.1	40.6	73.4	67.0	59.6	81.4	51.6	40.2	67.9	32.6	39.8	0.0	31.9	32.3	37.2	42.3
天津	15.3	33.0	32.1	36.2	67.8	66.8	57.3	79.0	51.8	40.0	67.5	10.8	32.7	31.9	0.0	24.5	27.4	38.7
济南	15.9	33.5	21.6	36.9	68.6	68.4	59.1	79.3	53.1	36.5	69.0	24.0	29.4	32.3	24.5	0.0	18.8	39.5
石家庄	17.7	38.2	22.8	40.9	69.0	69.7	60.5	79.0	54.9	40.7	68.5	26.4	36.3	37.2	27.4	18.8	0.0	44.1
南京	35.5	31.7	44.4	37.1	63.9	60.0	50.9	76.0	48.6	30.5	60.9	39.3	46.1	42.3	38.7	39.5	44.1	0.0

测试集构成

100万样本对

集内：已知方言

集外：未知方言

wav_id1	wav_id2	distance
A	B	?
C	D	?
.....	.....	.....
E	F	?

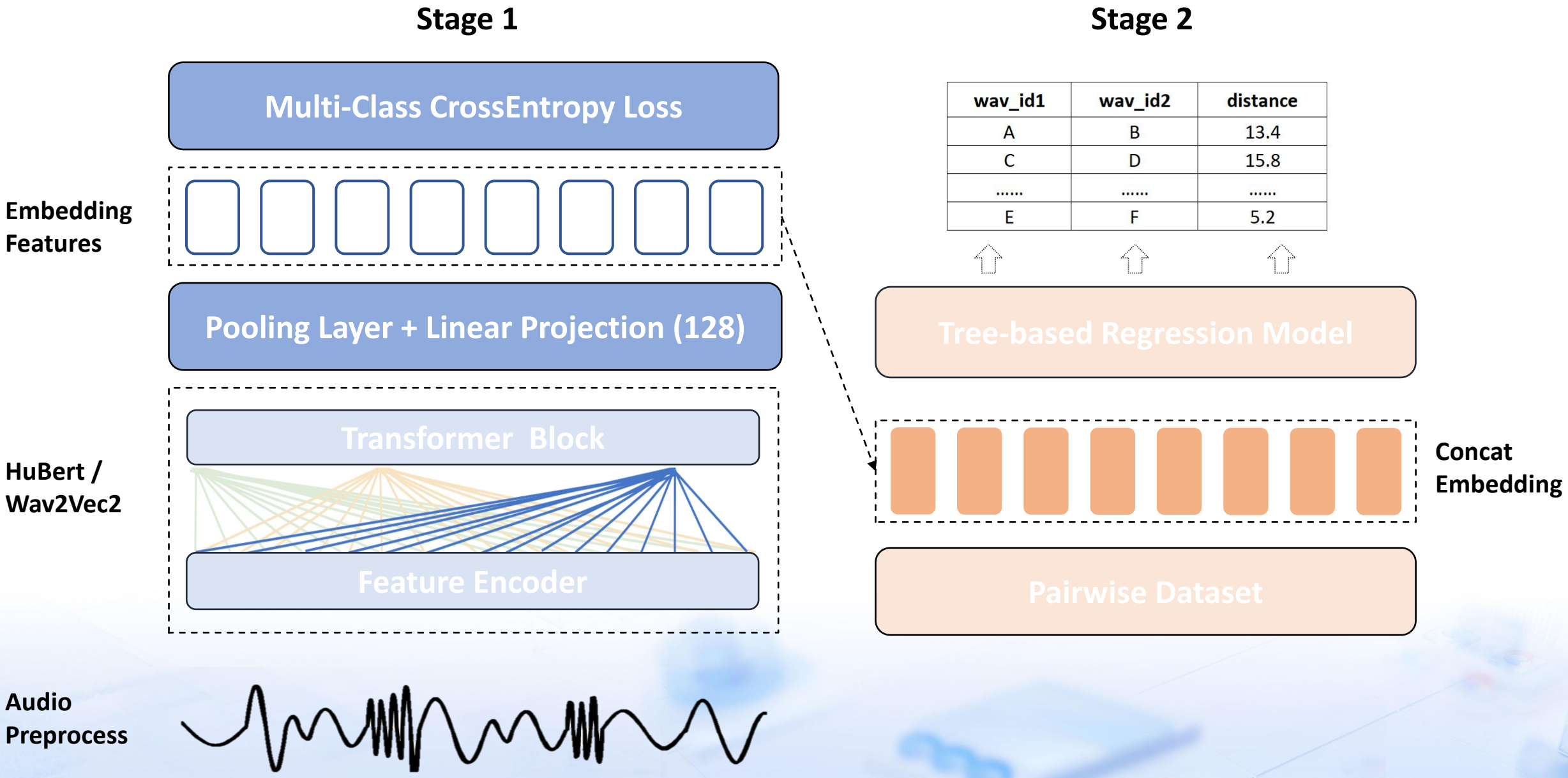
	wav_id2	训练语音集	测试语音集
wav_id1		方言1-18	方言19-N
测试语音集	方言 19-N	集内-集外	集外-集外

赛题理解

任务难点

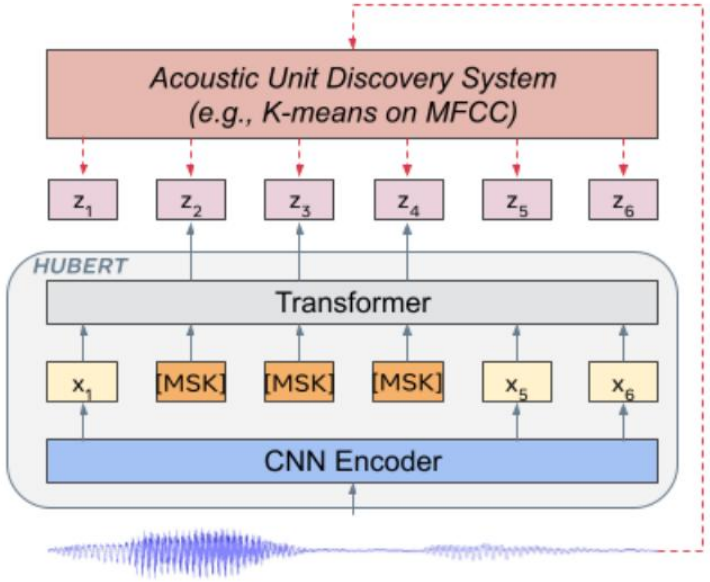
- 方言间的距离定义未知
- 测试集中存在训练集中不曾出现的未知方言
- 复赛测试集中无集内-集内类型的样本对
- 测试集距离分布未知

整体框架


Stage 1
Stage 2
Multi-Class CrossEntropy Loss
Embedding Features
Pooling Layer + Linear Projection (128)
HuBert / Wav2Vec2
Transformer Block
Feature Encoder
Audio Preprocess
wav_id1 wav_id2 distance
A B 13.4
C D 15.8
....
E F 5.2
Tree-based Regression Model
Concat Embedding
Pairwise Dataset

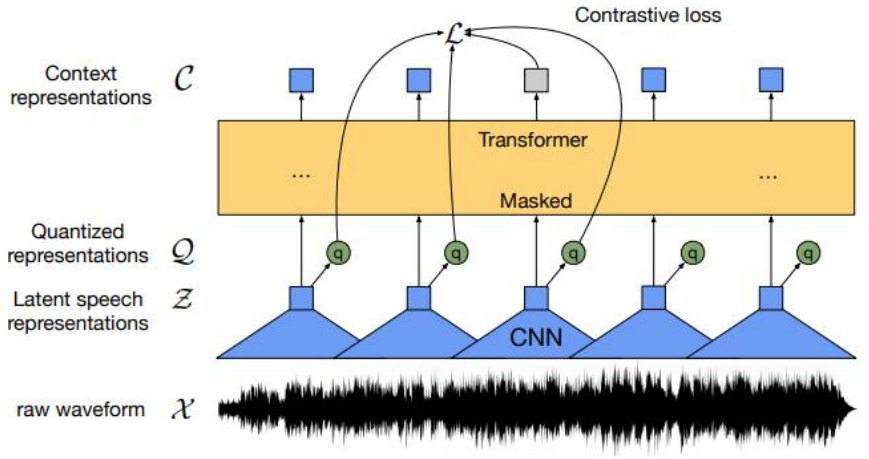
模型算法 – 预训练模型

HuBert



- cross-entropy loss
- clustering process
- re-uses embeddings from the BERT encoder

Wave2Vec2



- contrastive loss
- quantization process
- only uses convolutional network output

https://browse.arxiv.org/pdf/2106.07447.pdf
https://browse.arxiv.org/pdf/2006.11477.pdf

模型算法 - 多分类模型

Multi-Class CrossEntropy Loss

Linear Projection (128 DIM)

Pooling Layer (Max/Mean/Last hidden states)

Transformer Block

Feature Encoder

Chinese-HuBert-Large

Chinese-Wav2Vec2-Large

~~ERes2Net-Large~~

~~ECAPA-TDNN~~

~~Lang-id-voxlingua107-ecapa~~

Resample 16kHz + Max length Padding

模型参数

- 冻结 Feature Encoder
- learning rate: 1.e-04
- optimizer: AdamW
- scheduler: CosineAnnealing
- epoch: 初赛10 / 复赛5
- batch size: 32
- duration: 3.2

- HuBert与wav2Vec2显著优于其他模型
- 增大epoch对集内-集内类型有较大提升；而复赛期间，epoch5效果最优

模型算法 - 核心代码


```python
class AudioModel(nn.Module):
    def __init__(
            self,
            args,
            model_name,
            freeze_encoder,
            num_labels,
            config_path=None,
            use_gradient_checkpointing=False,
            pooling_params={},
            hidden_size=128,
        ):

        super().__init__()
        self.window_size = window_size

        self.config = AutoConfig.from_pretrained(f'{args.cache_dir}')
        self.backbone = AutoModel.from_pretrained(f'{args.cache_dir}', config=self.config)

        if freeze_encoder:
            self.backbone.feature_extractor._freeze_parameters()

        if use_gradient_checkpointing:
            self.backbone.gradient_checkpointing_enable()

        self.newfc_hidden = nn.Linear(self.backbone.config.hidden_size, hidden_size)
        self.fc = nn.Linear(hidden_size, num_labels)
```

关键步骤

- 载入预训练模型
- 冻结预训练模型Encoder层
- 使用gradient checkpointing降低显存使用率
- 定义emb降维全连接层

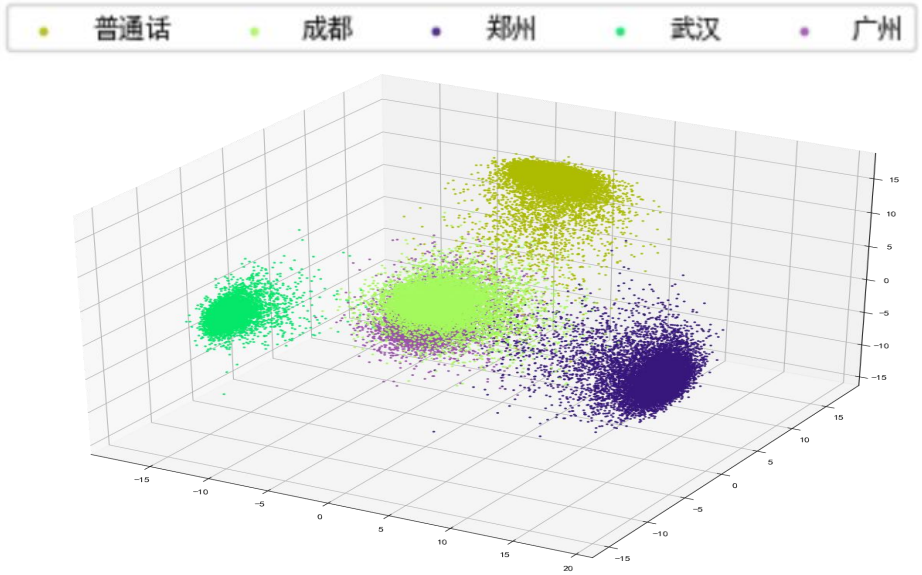
模型算法 - 核心代码

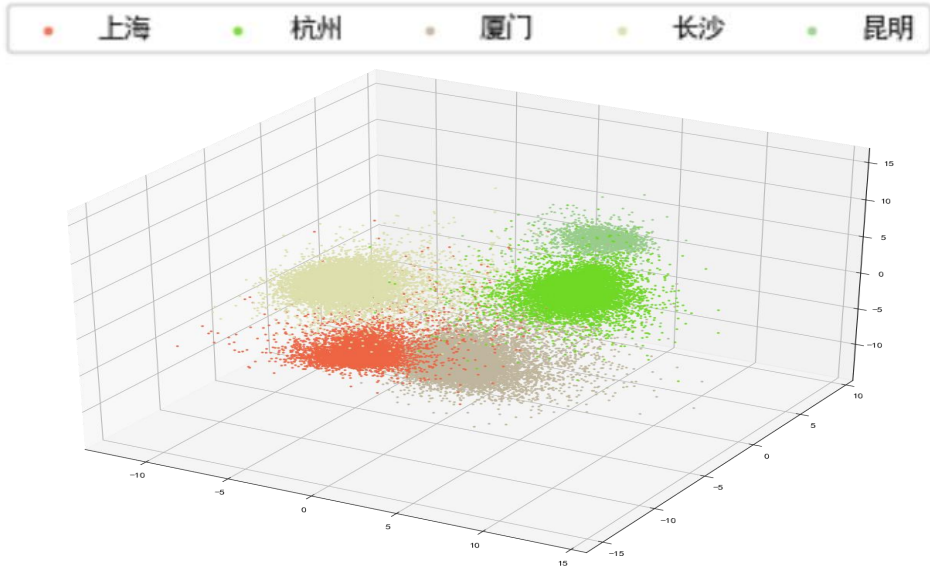

```python
def forward(self, x):

    features = self.backbone(x)

    if self.pooling_params['pooling_name'] == "MeanPooling":
        features = torch.mean(features.last_hidden_state, 1)
    elif self.pooling_params['pooling_name'] == "NoPooling":
        features = features.last_hidden_state[:, -1, :]
    elif self.pooling_params['pooling_name'] == "MaxPooling":
        features, _ = torch.max(features.last_hidden_state, 1)
    else:
        raise NotImplementedError

    embedding = self.newfc_hidden(features)

    pred = self.fc(torch.relu(embedding))

    return pred, embedding
```

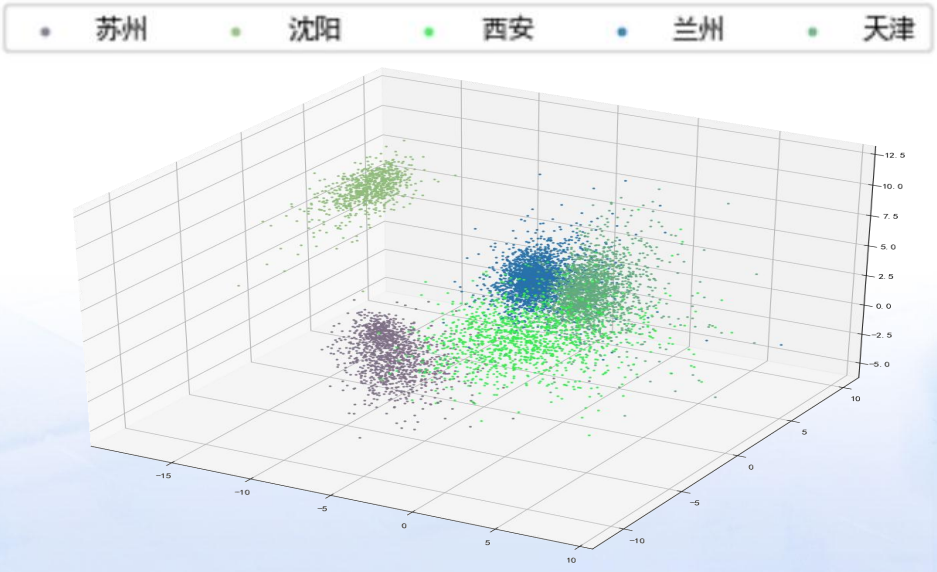
关键步骤

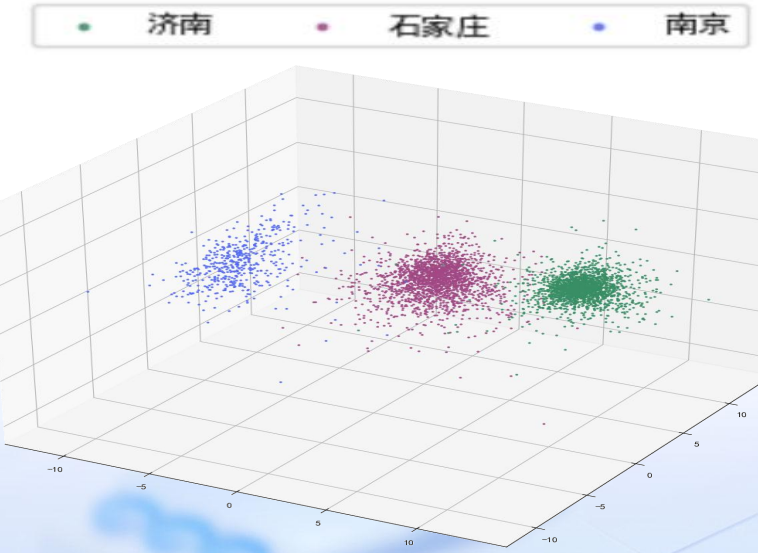
- 预训练模型抽取语音数据信息
- 定义不同的池化方法
- 将经过池化后的emb降维
- 最终输出模型结果与中间层emb

模型算法 - embedding可视化









模型算法 – 方言聚类

基于多分类模型embedding的方言聚类

cluster	Top1	Top2	Top3	Top4
0	武汉：47.3%	长沙：40.9%		
1	广州：99.2%			
2	郑州：98.8%			
3	北京：85.4%	石家庄：9.8%	济南：3.5%	
4	成都：33.9%	杭州：28.4%	兰州：8.4%	昆明：6.4%
5	厦门：58.2%	上海：38.3%		

- 经过多分类模型训练，部分方言间的相似度已与所提供距离矩阵趋于一致
- 由于距离定义未知，也仍存在部分方言间的相似度不符合认知，
  故需进一步通过回归模型优化

模型算法 - 回归模型

重要模型参数

- obejective: reg:absolute-error
- max_depth: 5
- learning_rate: 0.01
- colsample_bytree: 0.5
- subsample: 0.75

- 初赛时，当线下验证集包含集内-集内样本，模型几乎不会收敛，不断提升；
- 而当复赛去除该类型样本，模型很快就会收敛，且能使模型泛化的参数在复赛期间效果也最优。
- 特征中增加L1/L2/Cosine等其他维度，没有提升

模型算法 - 核心代码


```python
def random_choice(lst):
    idx = random.randrange(0, len(lst))
    return lst[idx]

def create_pair():
    distance_df = pd.read_excel('/xydata/复赛-距离矩阵.xlsx')
    distance_df = distance_df.melt(id_vars=['Unnamed: 0']).drop_duplicates()
    distance_df.columns = ['label1', 'label2', 'distance']
    distance_df = distance_df.replace('北京', '普通话')

    train_df = pd.read_csv('../proc/train_meta.csv')
    train_ids = train_df['id'].to_list()

    tr_list = []
    while len(tr_list) < 1000000:
        rand1 = random_choice(train_ids)
        rand2 = random_choice(train_ids)
        if rand1 == rand2:
            continue
        pair = rand1, rand2
        tr_list.append(pair)

    train_pair = pd.DataFrame(tr_list, columns=['id1', 'id2'])

    train_pair = train_pair.merge(
        train_df[['id', 'wav_path', 'label']].add_suffix('1'), how='left', on=['id1']
    )
    train_pair = train_pair.merge(
        train_df[['id', 'wav_path', 'label']].add_suffix('2'), how='left', on=['id2']
    )
    train_pair = train_pair.merge(
        distance_df, how='left', on=['label1', 'label2']
    )
    return train_pair
```

关键步骤

对训练集语音抽样

随机抽取100万的样本语音对

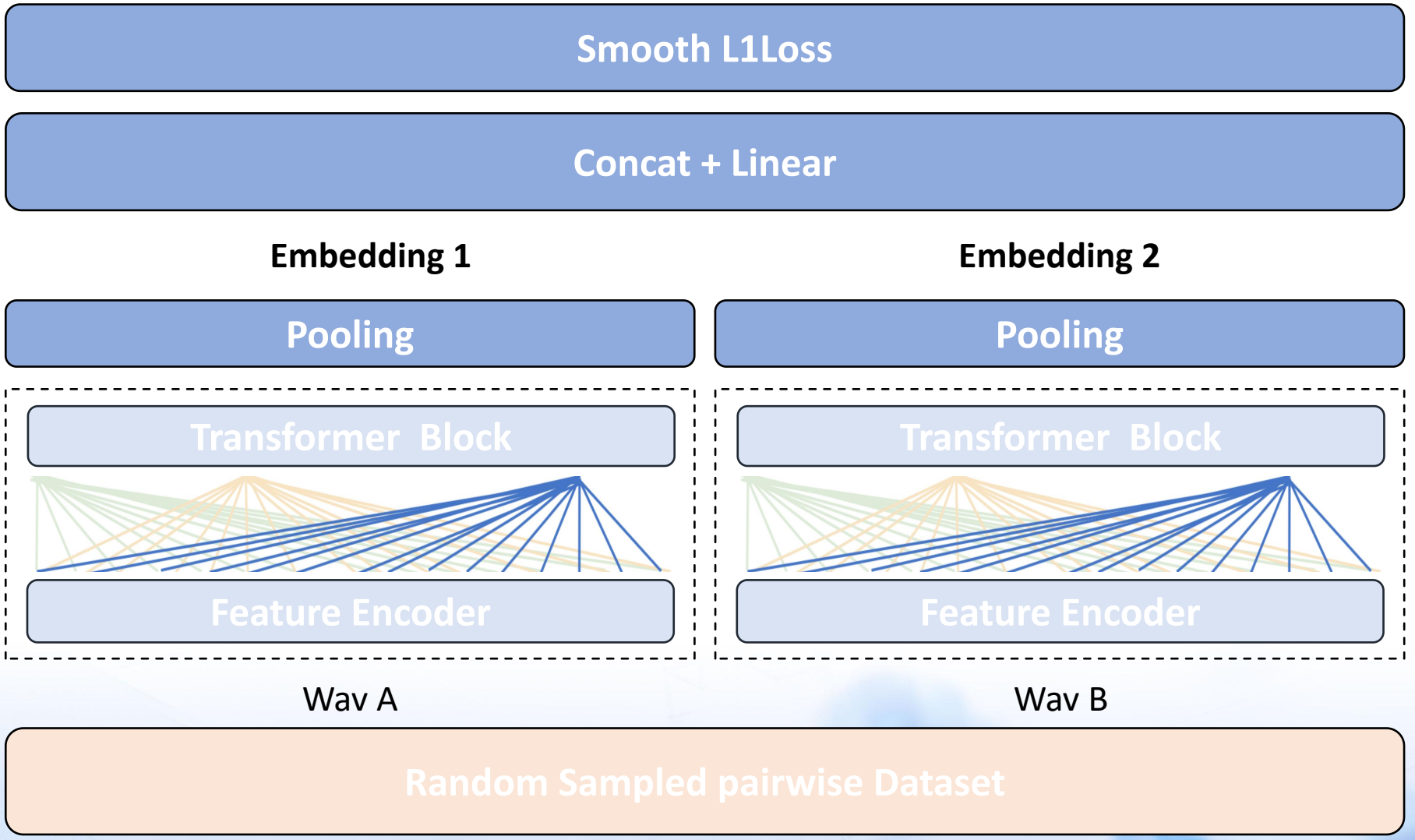
模型算法 - 核心代码


```python
def create_features(emb_query_dict, pair_df, emb_size=None):
    input1 = []
    input2 = []
    for id1, id2 in list(zip(pair_df.id1, pair_df.id2)):
        input1.append(emb_query_dict[id1])
        input2.append(emb_query_dict[id2])

    df0 = pd.DataFrame(np.vstack(input1), columns=[f'id1_emb{i}' for i in range(emb_size)])
    df1 = pd.DataFrame(np.vstack(input2), columns=[f'id2_emb{i}' for i in range(emb_size)])

    df = pd.concat([df0, df1], axis=1)

#     input1 = torch.tensor(np.vstack(input1)).to('cuda')
#     input2 = torch.tensor(np.vstack(input2)).to('cuda')

#     pdist_l1 = torch.nn.PairwiseDistance(p=1)
#     pdist_l2 = torch.nn.PairwiseDistance(p=2)

#     pdist_l1_value = pdist_l1(input1, input2)
#     pdist_l2_value = pdist_l2(input1, input2)

#     df['l1_distance'] = pdist_l1_value.cpu().numpy()
#     df['l2_distance'] = pdist_l2_value.cpu().numpy()

#     cos_sim = torch.nn.functional.cosine_similarity(input1, input2)
#     df['cos_sim'] = cos_sim.cpu().numpy()

    return df
```

关键步骤

- 将每个语音的emb存成query字典
- 构建100万语音样本对的emb拼接特征
- （相似性特征效果不佳）

模型算法 - 其他尝试



该模型结构效果较差

- 加载/不加载多分类模型权重，效果都不佳
- 训练、推理速度慢，实验迭代效率较低

模型算法 - 模型融合

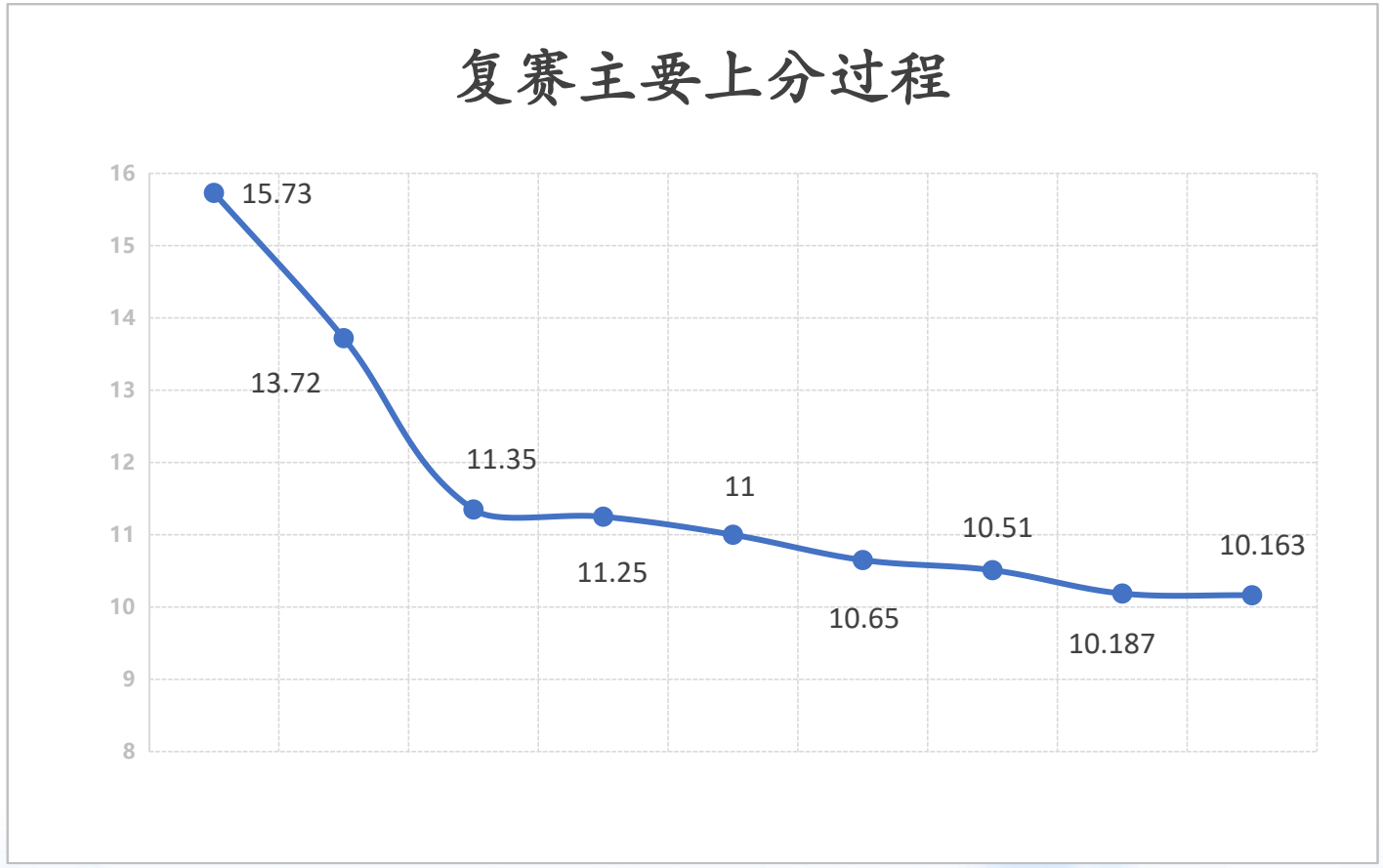
model_id	backbone	pooling	epoch	回归模型	复赛得分
1	hubert	mean pooling	10	xgboost	10.605
2	hubert	max pooling	10	xgboost	10.82
3	hubert	last hidden_states	10	xgboost	10.618
4	wave2vec2	mean pooling	10	xgboost	10.78
5	wave2vec2	max pooling	10	xgboost	10.65
6	wave2vec2	last hidden_states	10	xgboost	10.721
7	hubert	mean pooling	10	NN	11.886
8	hubert	mean pooling	5	xgboost	10.517
9	hubert	last hidden_state	5	xgboost	10.53
10	wave2vec2	max pooling	5	xgboost	10.578
11	wave2vec2	last hidden_states	5	xgboost	10.641

最优单模型复赛得分：10.517；线上推理时间：3~5分钟

加权融合模型 8、9、10，复赛得分：10.187；线上推理时间：12~15分钟；已足够获得第一

进一步加权融合模型1 、2，复赛得分提升至：10.163

模型算法 - 迭代过程



1. 多分类结果直接提交
2. 复现初赛模型 多分类+回归
3. 修改回归模型目标函数
4. 回归模型调参
5. 尝试更多Pooling方法
6. 调整回归模型轮数
7. 调整多分类模型epoch数
8. 3个模型融合
9. 5个模型融合

心得总结

有效优化点

模型层面

- ✓ 两阶段建模，多分类（NN）+ 回归（GBDT）
- ✓ HuBert/Wav2Vec2 + 多种Pooling方法，增加模型多样性
- ✓ 分别将注重拟合/泛化的模型，应用于集内-集内/涉及集外的数据集
- ✓ 一致的线上、线下趋势

工程层面

- ✓ 多分类模型训练、推理时使用AMP加速，效率提升近一倍
- ✓ 预先存储已知方言语音的多分类embedding，线上只需推理未知方言
- ✓ 基于树的回归模型在百万级样本量上，预测只需几秒，效率极高

无效/潜在优化点

- ➢ 直接使用NN，基于pairwise语音对，进行端到端训练
- ➢ 加载多分类模型权重，冻结/不冻结权重，基于pairwise语音对，添加linear层后训练
- ➢ GBDT模型中，除embedding特征外，增加相似性特征
- ➢ 多分类模型使用ArcMargin
- ➢ Soft - label / 蒸馏 （初赛阶段提升约0.1）

信也科技杯 FINVOLUTION BIG DATA COMPETITION | 信也 FINVOLUTION

THANK YOU