低资源方言语音识别系统

曹松军
xmdxcsj@gmail.com

主要内容

- 一、语音识别简介
- 二、构建低资源方言识别系统

语音助手-产品应用

覆盖场景

合作客户

TCL　Skyworth 创维　GREE 格力空调

Hisense 海信电视　Haier　SHARP

优必选 UBTECH　ling 物灵

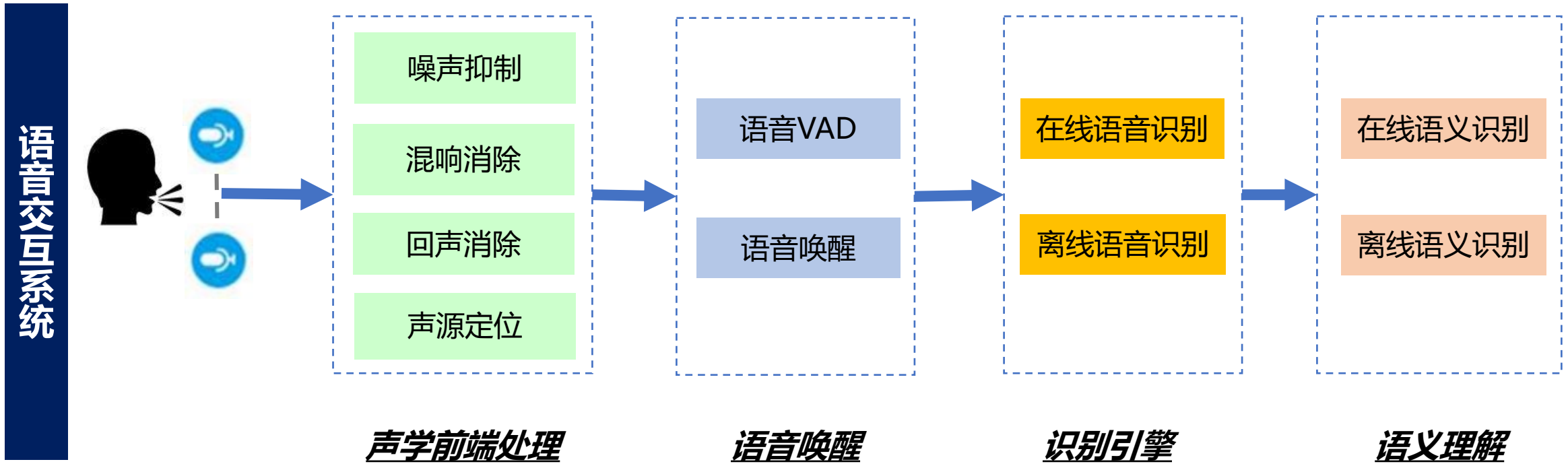
江铃汽车　广汽三菱　吉利汽车 GEELY AUTO

长安汽车 CHANGAN　MASERATI　Audi

上汽大众 SAIC VOLKSWAGEN　CHRYSLER　理想

Ford　长城汽车

语音助手-技术模块



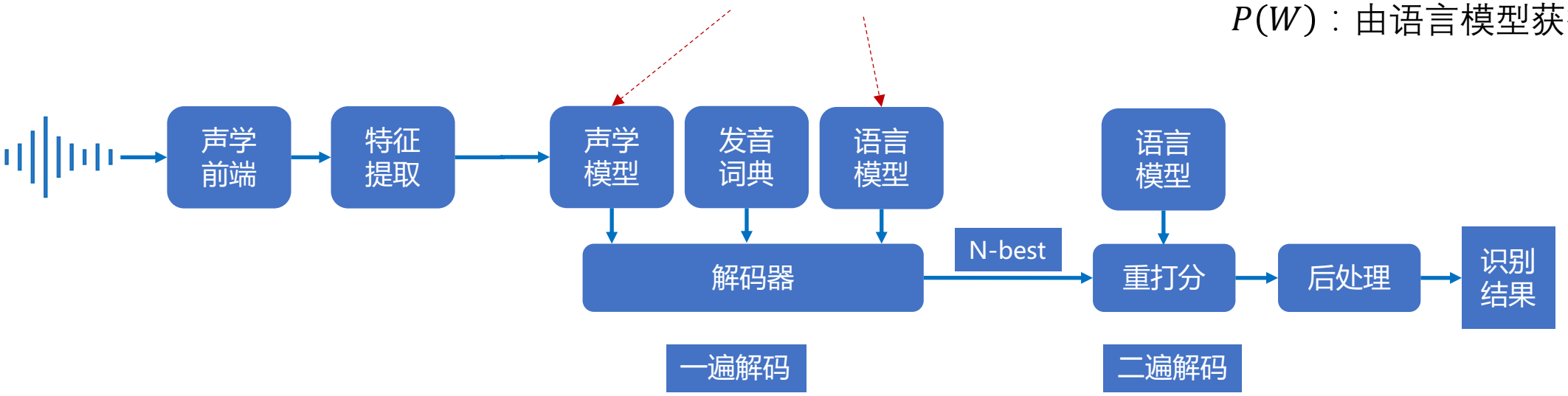
自动语音识别技术(automatic speech recognition)，可以简写为**ASR**，在语音交互过程中，ASR技术模块负责将用户的语音转化为文本，转化得到的文本会传给下游的NLP技术模块做语义理解，所以ASR模块的识别效果对整个语音交互的体验至关重要。

ASR-技术架构

系统架构

$$\widehat{W} = \arg\max P(W|O) = \arg\max(P(O|W)P(W))/(P(O))$$
$$\widehat{W} = \arg\max P(O|W)\,P(W)$$

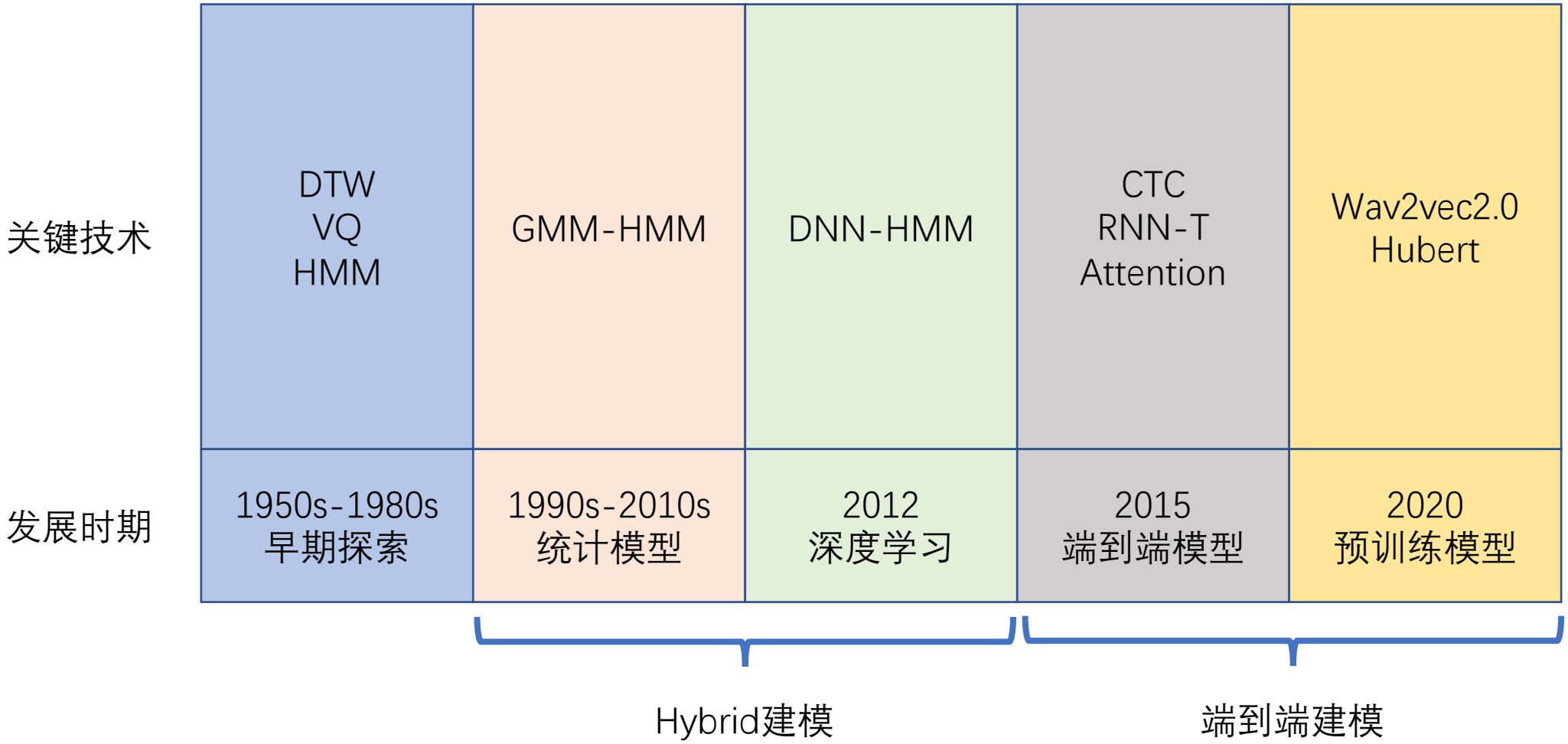
O: 观察值序列（语音特征）
$\widehat{W}$：最优词序列（解码结果）
$P(O|W)$：由声学模型获得
$P(W)$：由语言模型获得



评价指标

- ✓ 字错误率（Character Error Rate，CER）= 识别错误字数 / 测试集中总字数
- ✓ 句错误率（Sentence Error Rate，SER）= 识别错误的句子数 / 测试集中总句子数
- ✓ 句正确率（Sentence Accuracy）= 识别正确的句子数 / 测试集中总句子数 = 1 – SER

ASR-发展阶段


关键技术
DTW
VQ
HMM
GMM-HMM
DNN-HMM
CTC
RNN-T
Attention
Wav2vec2.0
Hubert
发展时期
1950s-1980s
早期探索
1990s-2010s
统计模型
2012
深度学习
2015
端到端模型
2020
预训练模型
Hybrid建模
端到端建模

ASR-hybrid建模

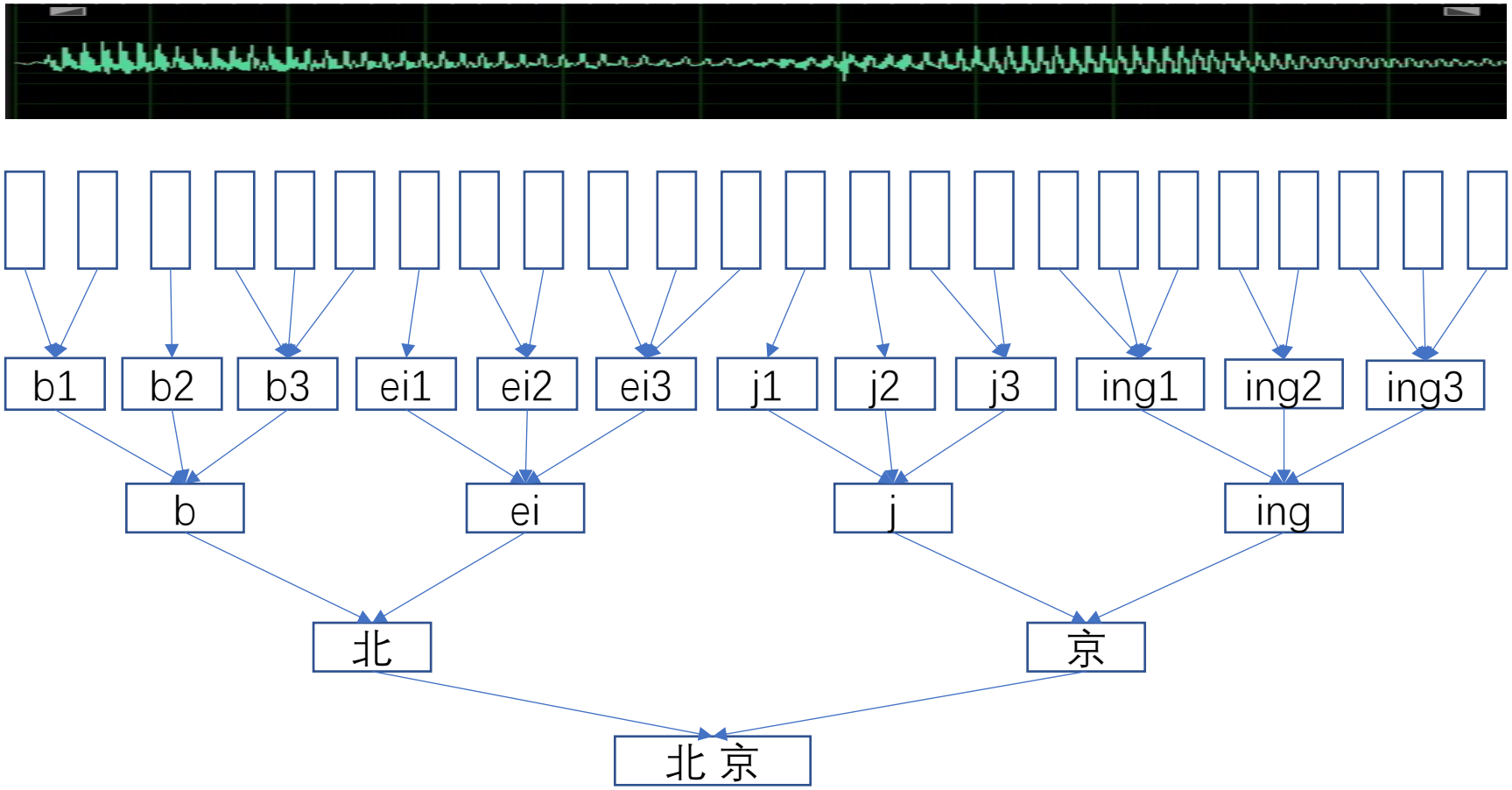
1.特征抽取
时域点->频谱特征

2.1声学模型-DNN
计算p(状态|特征)

2.2声学模型-HMM
状态->音素

3.发音词典
音素->词

4.语言模型
词->句子



建模到HMM的状态

ASR-hybrid建模

问题1：每一帧对应一个HMM状态，拆分科学吗？

- 语音过渡地带

问题2：训练流程复杂？

- 第一步，训练hmm-gmm
- 第二步，使用hmm-gmm来做对齐获取每一帧特征的label
- 第三步，训练帧到label的鉴别模型

问题3：独立性假设？

- 每一帧的输出只跟当前帧有关，跟上一帧的输出无关

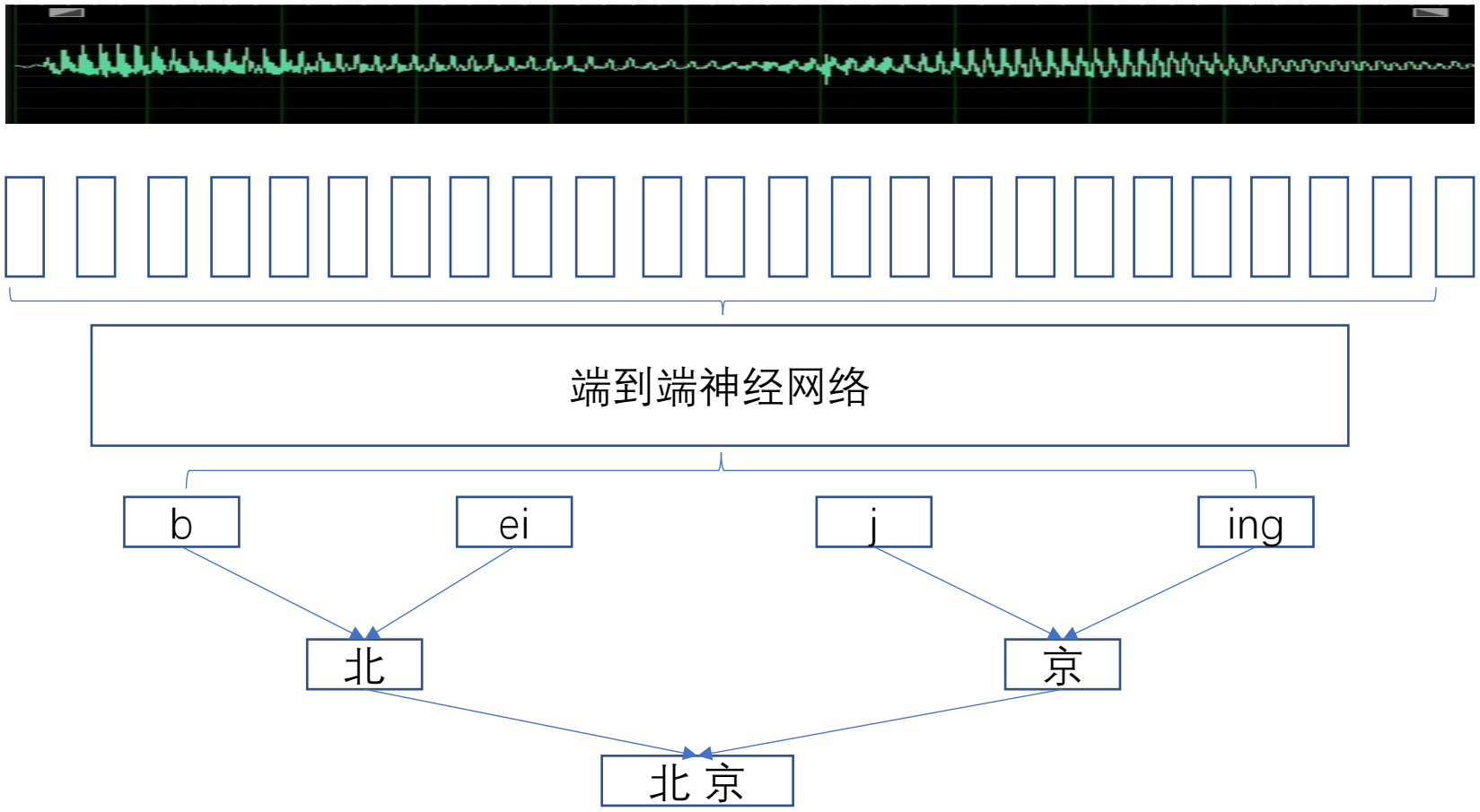
ASR-端到端建模

1.特征抽取
时域点->频谱特征

2.端到端建模
计算p(音素|特征)

3.发音词典
音素->词

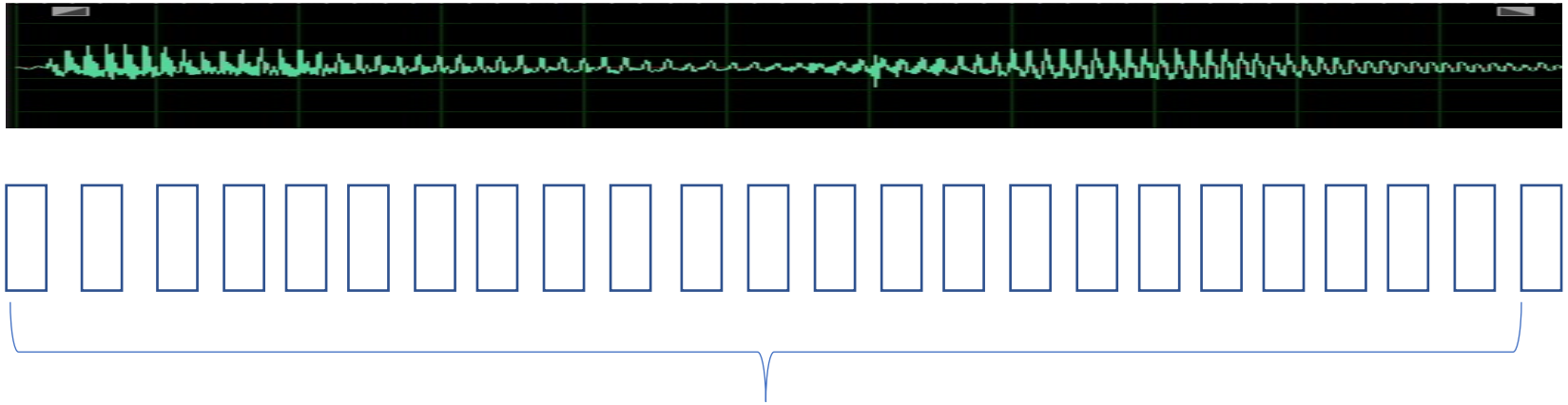
4.语言模型
词->句子



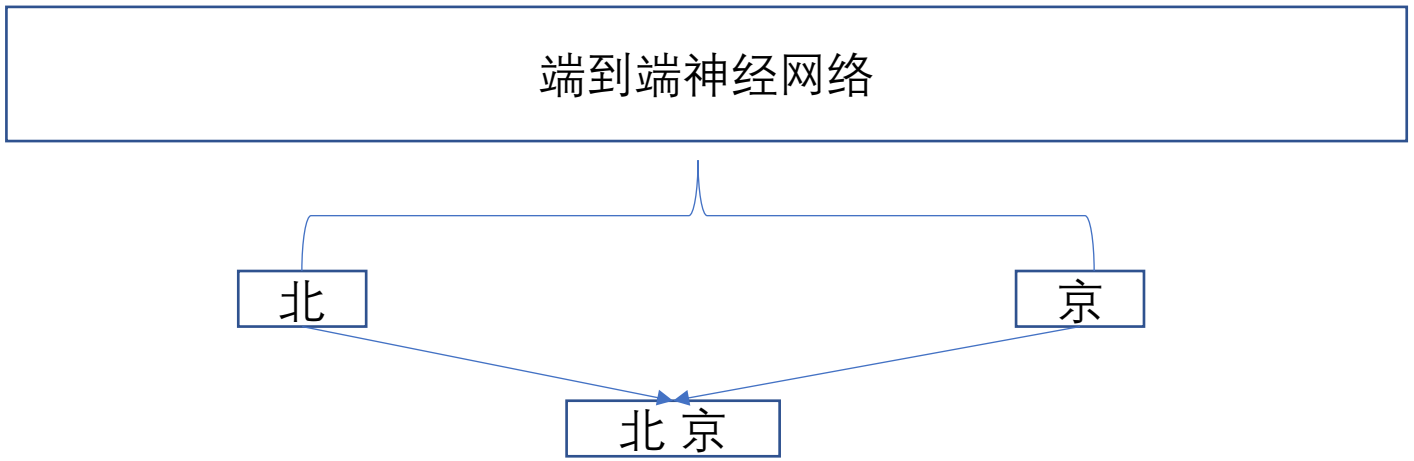
没有HMM，直接建模到音素，需要词典映射为汉字

ASR-端到端建模

1.特征抽取
时域点->频谱特征



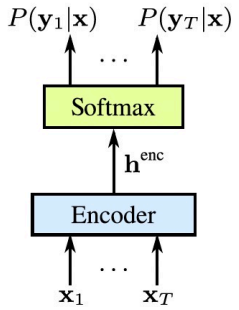
2.端到端建模
计算p(词|特征)



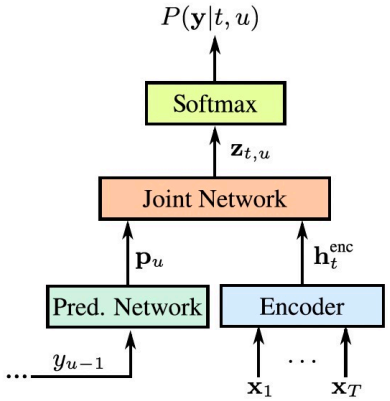
3.语言模型
词->句子

没有HMM，不需要词典，直接建模到汉字

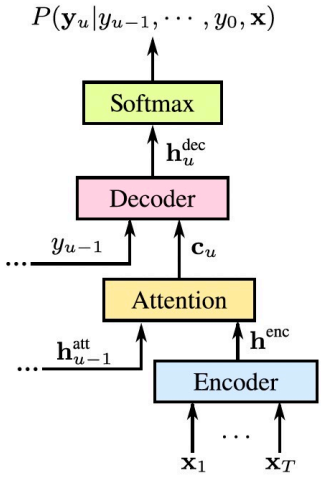
ASR-端到端建模



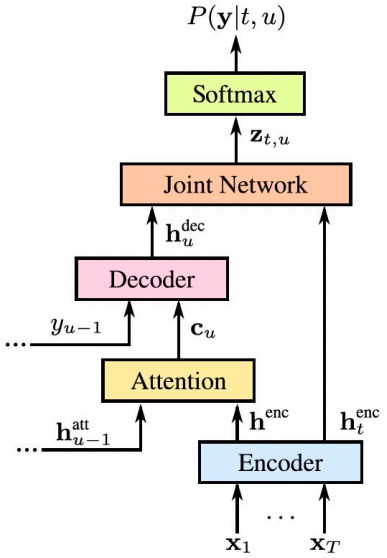
(a.) CTC



(b.) RNN-Transducer



(c.) Attention-based Model



(d.) RNN-Transducer with Attention

Table 1: *WERs (%) on various test sets for the models compared in this work. The attention-based model with two decoder layers is the single best sequence-to-sequence model.*

Model	Clean		Noisy		numeric
	dict	vs	dict	vs	
Baseline Uni. CDP	6.4	9.9	8.7	14.6	11.4
Baseline BiDi. CDP	5.4	8.6	6.9	-	11.4
End-to-end systems					
CTC-grapheme[3]	39.4	53.4	-	-	-
RNN Transducer	6.6	12.8	8.5	22.0	9.9
RNN Trans. with att.	6.5	12.5	8.4	21.5	9.7
Att. 1-layer dec.	6.6	11.7	8.7	20.6	9.0
Att. 2-layer dec.	**6.3**	**11.2**	**8.1**	**19.7**	**8.7**

Hybrid

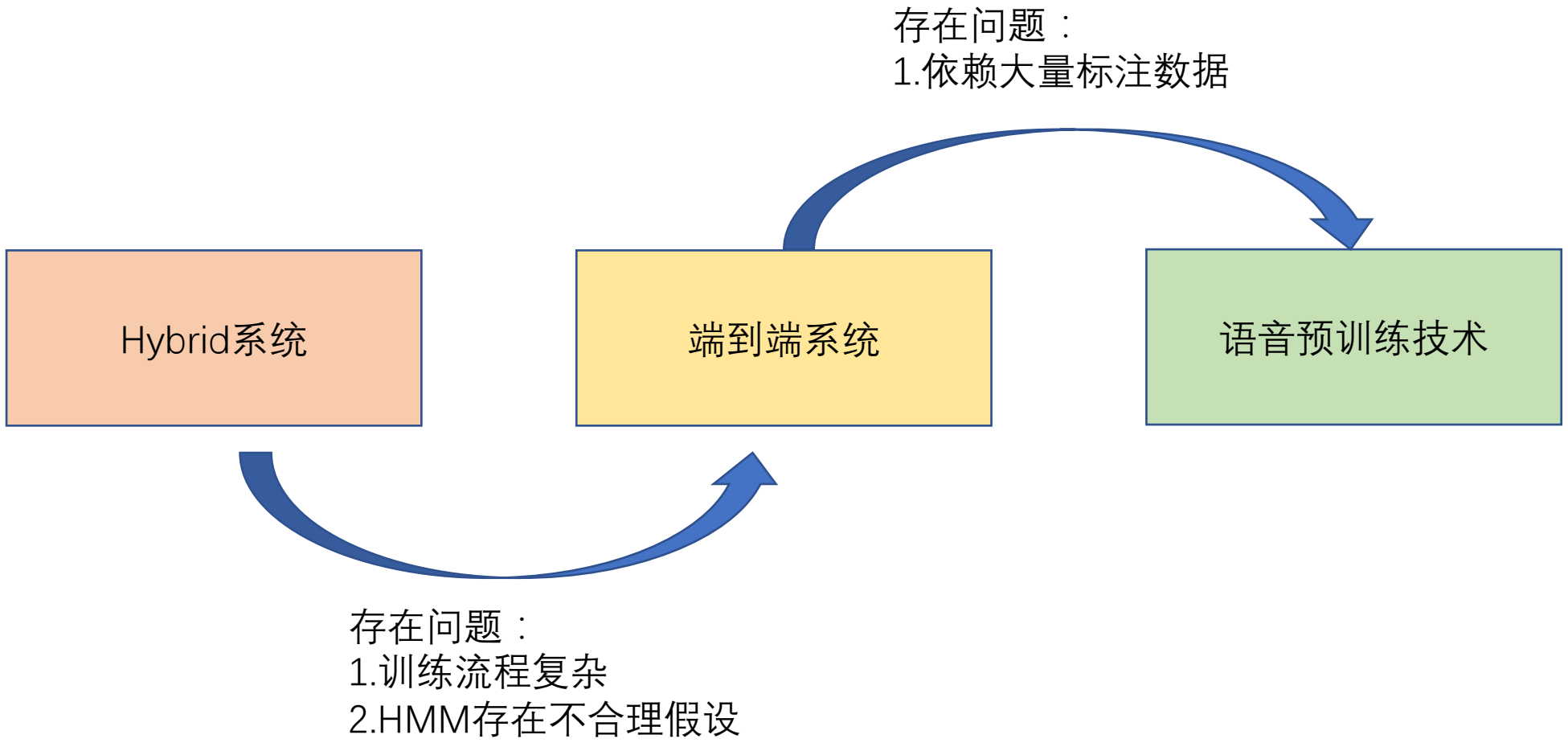
End2end

相比hybrid传统方案，端到端建模有以下特点：

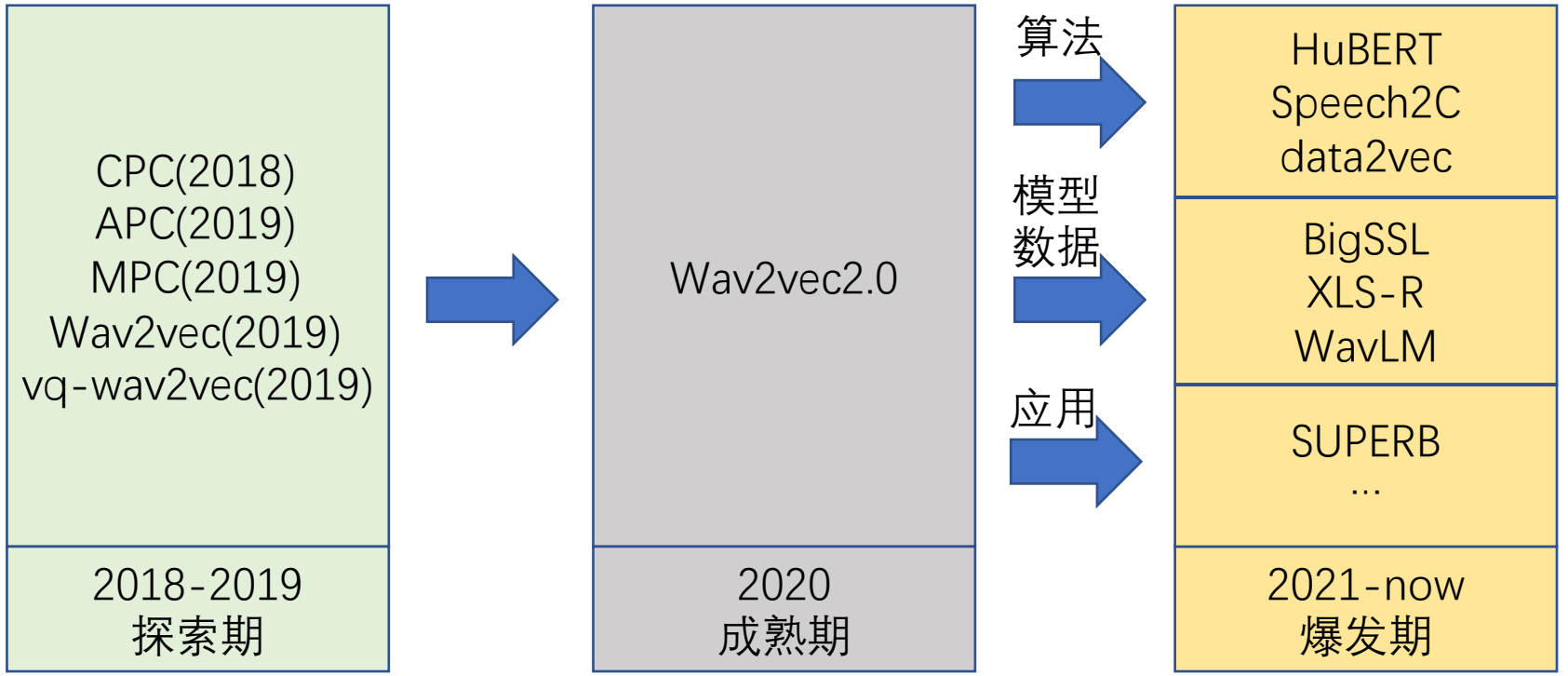
- 训练简单，直接seq2seq进行训练
- Attention/Transducer loss修正了输出独立性假设
- **对标注数据量要求比较多？？**

"A Comparison of Sequence-to-Sequence Models for Speech Recognition." *Interspeech*. 2017.

ASR-预训练模型


存在问题：
1.依赖大量标注数据
Hybrid系统
端到端系统
语音预训练技术
存在问题：
1.训练流程复杂
2.HMM存在不合理假设

ASR-预训练模型

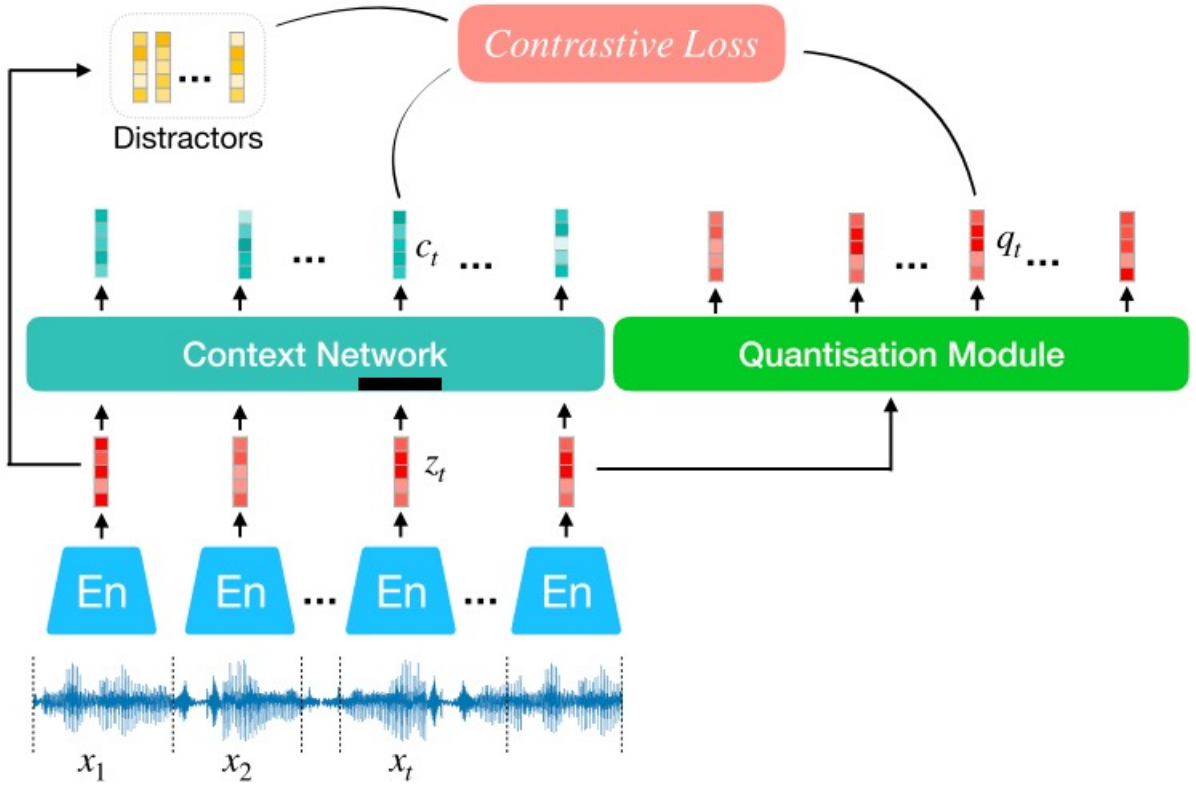


相比baseline系统，还没有取得理想的效果

在已有公开数据集取得SOTA结果，大幅优于已有baseline

成为业界主流架构在多个方向快速演进

ASR-预训练模型



网络分成三部分：
1. Encoder：将语音输入转化为特征序列
2. Context Network：将输入特征进行编码抽象
3. Quantisation Module：将目标进行离散化

Contrastive loss：
1. q_t表示正样本，q̃表示负样本
2. $sim()$表示两个向量的cos距离

$$\mathcal{L}_m = -\log \frac{\exp(sim(\mathbf{c}_t, \mathbf{q}_t))/\kappa}{\sum_{\tilde{\mathbf{q}} \sim \mathbf{Q}_t} \exp(sim(\mathbf{c}_t, \tilde{\mathbf{q}}))/\kappa}$$

Masking：
1. Encoder的输出做mask作为context网络的输入
2. 采取分段连续masking策略，整体占比49%

"wav2vec 2.0: A framework for self-supervised learning of speech representations." *Advances in Neural Information Processing Systems* 33 (2020): 12449-12460.

ASR-预训练模型

1h labeled						
Discrete BERT [4]	LS-960	4-gram	8.5	16.4	9.0	17.6
BASE	LS-960	4-gram	5.0	10.8	5.5	11.3
		Transf.	3.8	9.0	4.0	9.3
LARGE	LS-960	Transf.	3.8	7.1	3.9	7.6
	LV-60k	Transf.	3.3	6.4	3.4	6.8
100h labeled						
Hybrid DNN/HMM [33]	-	4-gram	5.0	19.5	5.8	18.6
TTS data augm. [29]	-	LSTM			4.3	13.5
Discrete BERT [4]	LS-960	4-gram	4.0	10.9	4.5	12.1
Iter. pseudo-labeling [56]	LS-860	4-gram+Transf.	5.0	8.72	5.37	9.51
Iter. pseudo-labeling [56]	+LV-60k	4-gram+Transf.	3.19	6.14	3.72	7.11
Noisy student [41]	LS-860	LSTM	3.9	8.8	4.2	8.6
BASE	LS-960	4-gram	2.7	7.9	3.4	8.0
		Transf.	2.2	6.3	2.6	6.3
LARGE	LS-960	Transf.	2.1	4.8	2.3	5.0
	LV-60k	Transf.	2.0	4.1	2.1	4.4

1小时结果

100小时baseline

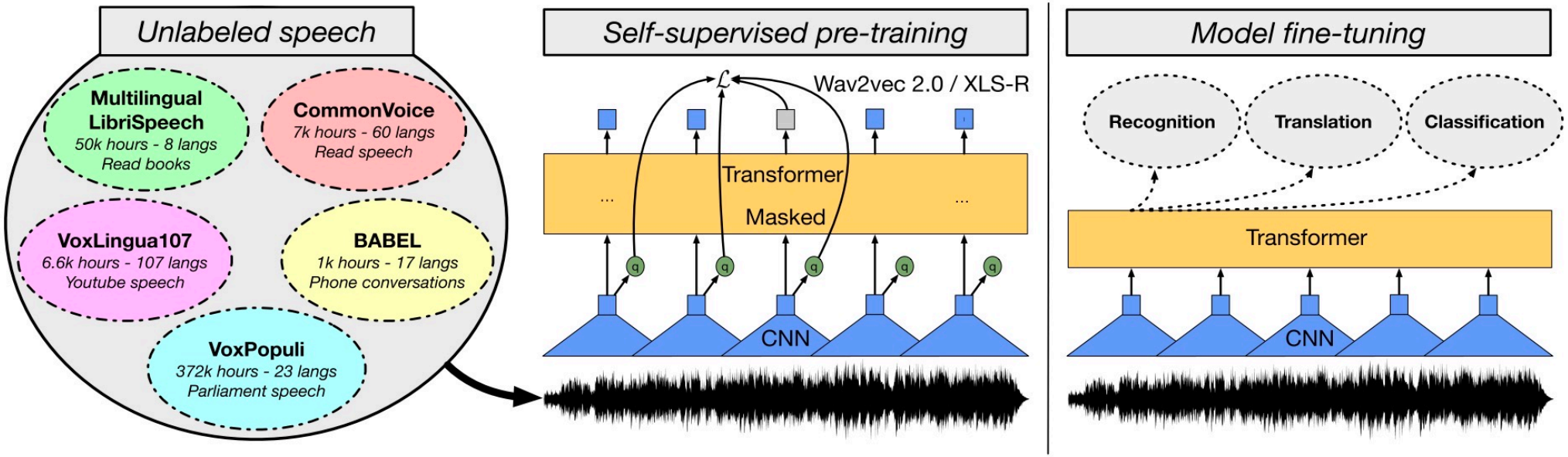
100小时结果

Wav2vec2.0 在语音识别任务上面的实验结果：

- 在Librispeech公开数据集的100小时任务上面取得了SOTA的结果，明显优于之前的工作
- 只使用了1小时的标注数据，wav2vec 2.0的结果就可以优于之前100小时标注数据的最好结果

wav2vec 2.0工作发挥了预训练在语音任务的威力，大幅降低了对标注数据的依赖

ASR-预训练模型



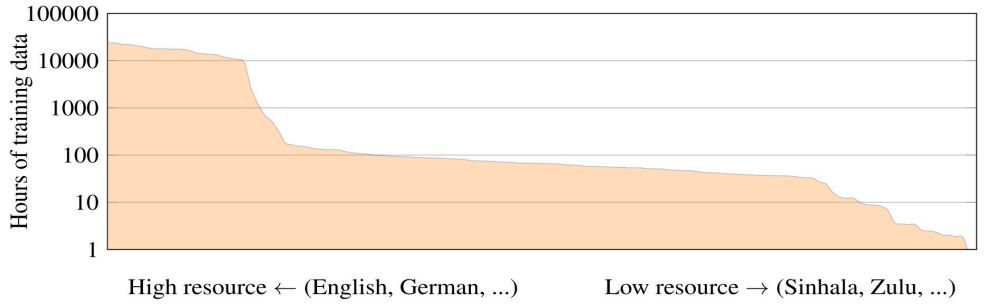


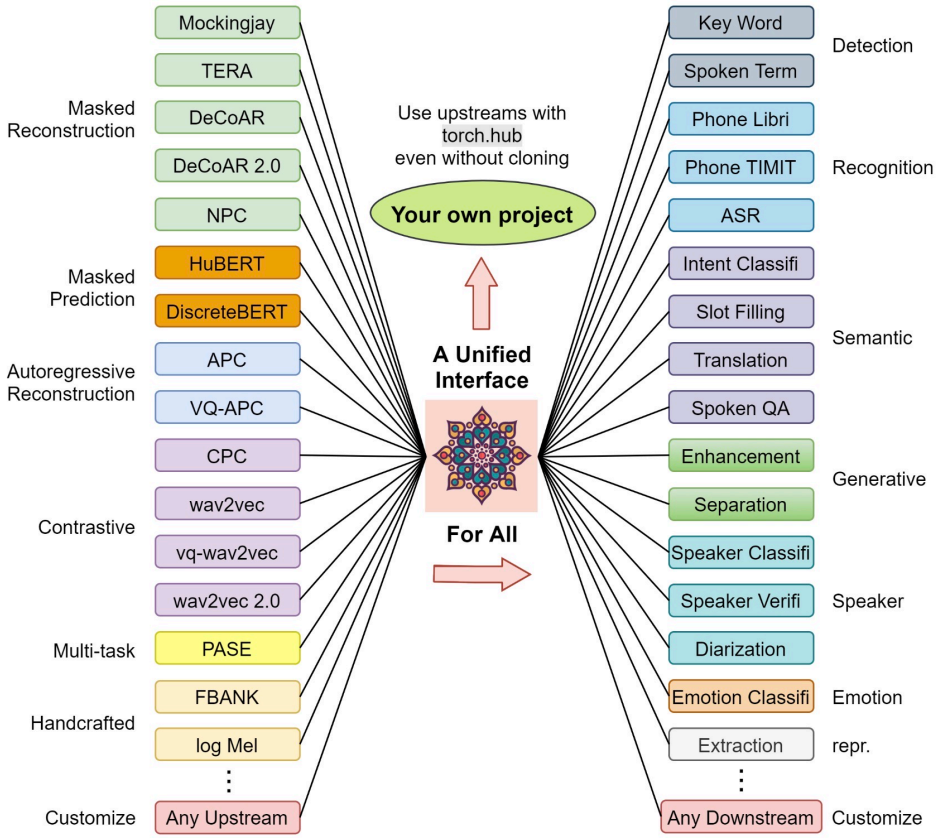
Figure 2: Illustration of the unlabeled training data distribution across the 128 languages of XLS-R.

	es	fr	it	ky	nl	ru	sv	tr	tt	zh-HK	Avg
Labeled data	1h	1h	1h	1h	1h	1h	1h	1h	1h	1h	
Previous work											
m-CPC	38.0	47.1	40.5	41.2	42.5	43.7	47.5	47.3	42.0	55.0	44.5
Fer et al. (2017)	36.6	48.3	39.0	38.7	47.9	45.2	52.6	43.4	42.5	54.3	44.9
XLSR-10	7.9	12.6	11.7	7.0	14.0	9.3	20.6	9.7	7.2	22.8	12.3
XLSR-53	2.9	5.0	5.7	6.1	5.8	8.1	12.2	7.1	5.1	18.3	7.6
This work											
XLS-R (0.3B)	3.1	5.4	4.9	5.1	5.8	6.0	7.2	6.0	4.1	17.0	6.5
XLS-R (1B)	**2.0**	**3.9**	**3.5**	4.1	**4.2**	4.1	5.5	4.4	3.4	15.7	5.1
XLS-R (2B)	2.2	4.0	**3.5**	**4.0**	4.7	**3.7**	**5.0**	**4.0**	**2.9**	**14.8**	**4.9**

Meta AI关于多语种预训练的工作**XLS-R**，预训练数据来自于**128个语种，合计43.6万小时**，在公开数据集common voice上面取得了大幅提升，**代码和模型已经开源：**
https://github.com/facebookresearch/fairseq/blob/main/examples/wav2vec/xlsr/README.md

"Xls-r: Self-supervised cross-lingual speech representation learning at scale." ***arXiv preprint arXiv:2111.09296*** **(2021).**

ASR-预训练模型



* The four columns (1)~(4) correspond to the macs calculated with short, medium, long, longer bucket respectively

Method	Rank ↑	Score ↑	PR ↓	KS ↑	IC ↑	SID ↑	ER ↑	ASR ↓	QbE ↑	SF-F1 ↑	SF-CER ↓	SV ↓	SD ↓
WavLM Large	19.9	1145	3.06	97.86	99.31	95.49	70.62	3.44	8.86	92.21	18.36	3.77	3.24
WavLM Base+	18.7	1106	3.92	97.37	99	89.42	68.65	5.59	9.88	90.58	21.2	4.07	3.5
WavLM Base	16.9	1019	4.84	96.79	98.63	84.51	65.94	6.21	8.7	89.38	22.86	4.69	4.55
HuBERT Large	15.8	919	3.53	95.29	98.76	90.33	67.62	3.62	3.53	89.81	21.76	5.98	5.75
wav2vec 2.0 Large	15.4	914	4.75	96.66	95.28	86.14	65.64	3.75	4.89	87.11	27.31	5.65	5.62
HuBERT Base	15.25	941	5.41	96.3	98.34	81.42	64.92	6.42	7.36	88.53	25.2	5.11	5.88
LightHuBERT Small	13.95	901	6.6	96.07	98.23	69.7	64.12	8.34	7.64	87.58	26.9	5.42	5.85
FaST-VGS+	13.15	809	7.76	97.27	98.97	41.34	62.71	8.83	5.62	88.15	27.12	5.87	6.05
wav2vec 2.0 Base	12.35	818	5.74	96.23	92.35	75.18	63.43	6.43	2.33	88.3	24.77	6.02	6.08
DistilHuBERT	11.2	717	16.27	95.98	94.99	73.54	63.02	13.37	5.11	82.57	35.59	8.55	6.19
DeCoAR 2.0	10.6	722	14.93	94.48	90.8	74.42	62.47	13.02	4.06	83.28	34.73	7.16	6.59
wav2vec	8.9	529	31.58	95.59	84.92	56.56	59.79	15.86	4.85	76.37	43.71	7.99	9.9
vq-wav2vec	7	422	33.48	93.38	85.68	38.8	58.24	17.71	4.1	77.68	41.54	10.38	9.93
APC	5.8	392	41.98	91.01	74.69	60.42	59.33	21.28	3.1	70.46	50.89	8.56	10.53
VQ-APC	5.75	377	41.08	91.11	74.48	60.15	59.66	21.2	2.51	68.53	52.91	8.72	10.45
NPC	5.4	386	43.81	88.96	69.44	55.92	59.08	20.2	2.46	72.79	48.44	9.4	9.34

Benchmark：https://github.com/s3prl/s3prl

Leaderboard：https://superbbenchmark.org/

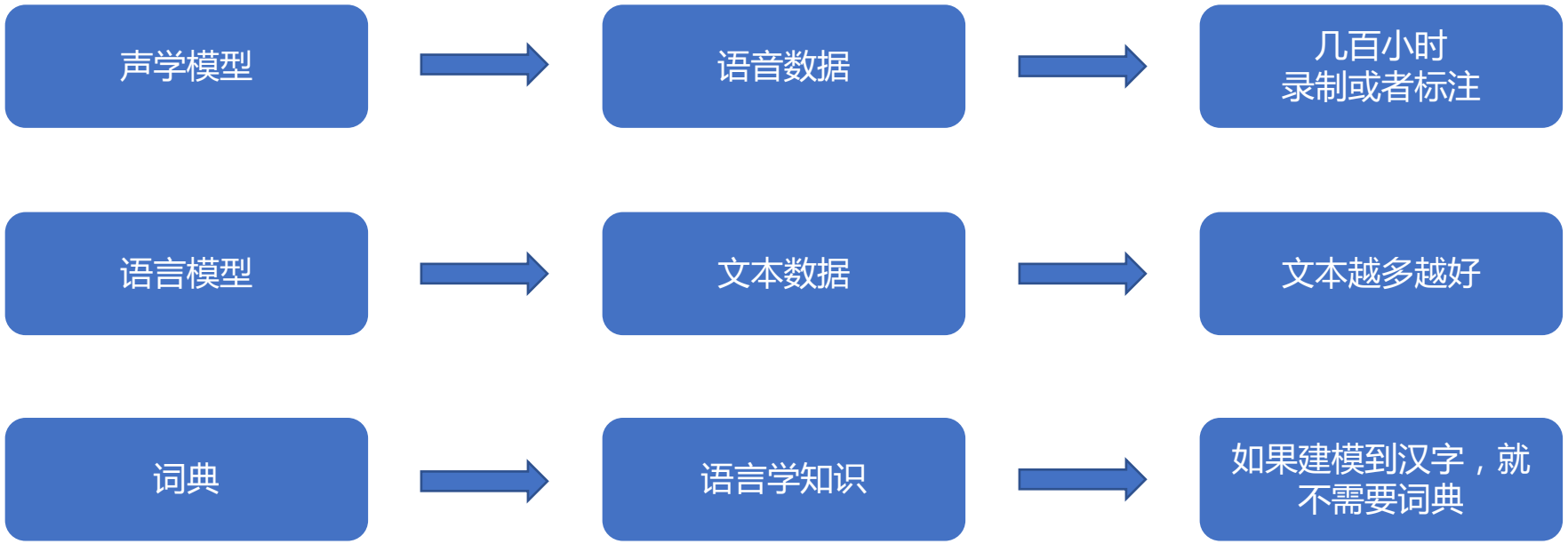
SUPERB构建了衡量语音预训练模型效果的benchmark，在语音识别、说话人识别、情感识别等十几个下游任务的公开数据集上面均有对应的结果。

"Superb: Speech processing universal performance benchmark." ***arXiv preprint arXiv:2105.01051*** **(2021).**

主要内容

- 一、语音识别简介
- 二、构建低资源方言识别系统

数据需求



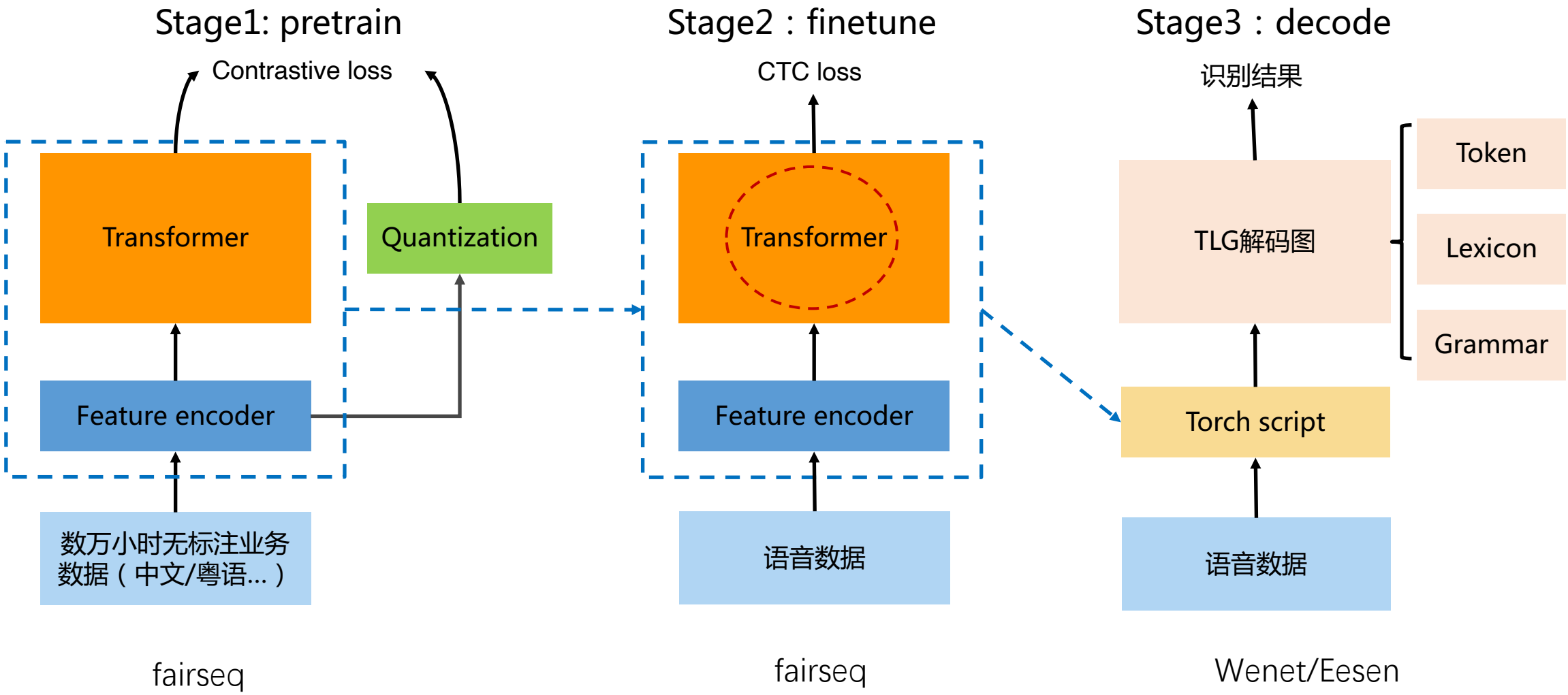
语音数据：

1. 已有语音数据，通过众包或者数据标注公司进行数据标注，成本大概几百块/小时
2. 已有文本数据，通过众包招人去做录音
3. 爬取带字幕的数据：视频/电视台/电影等

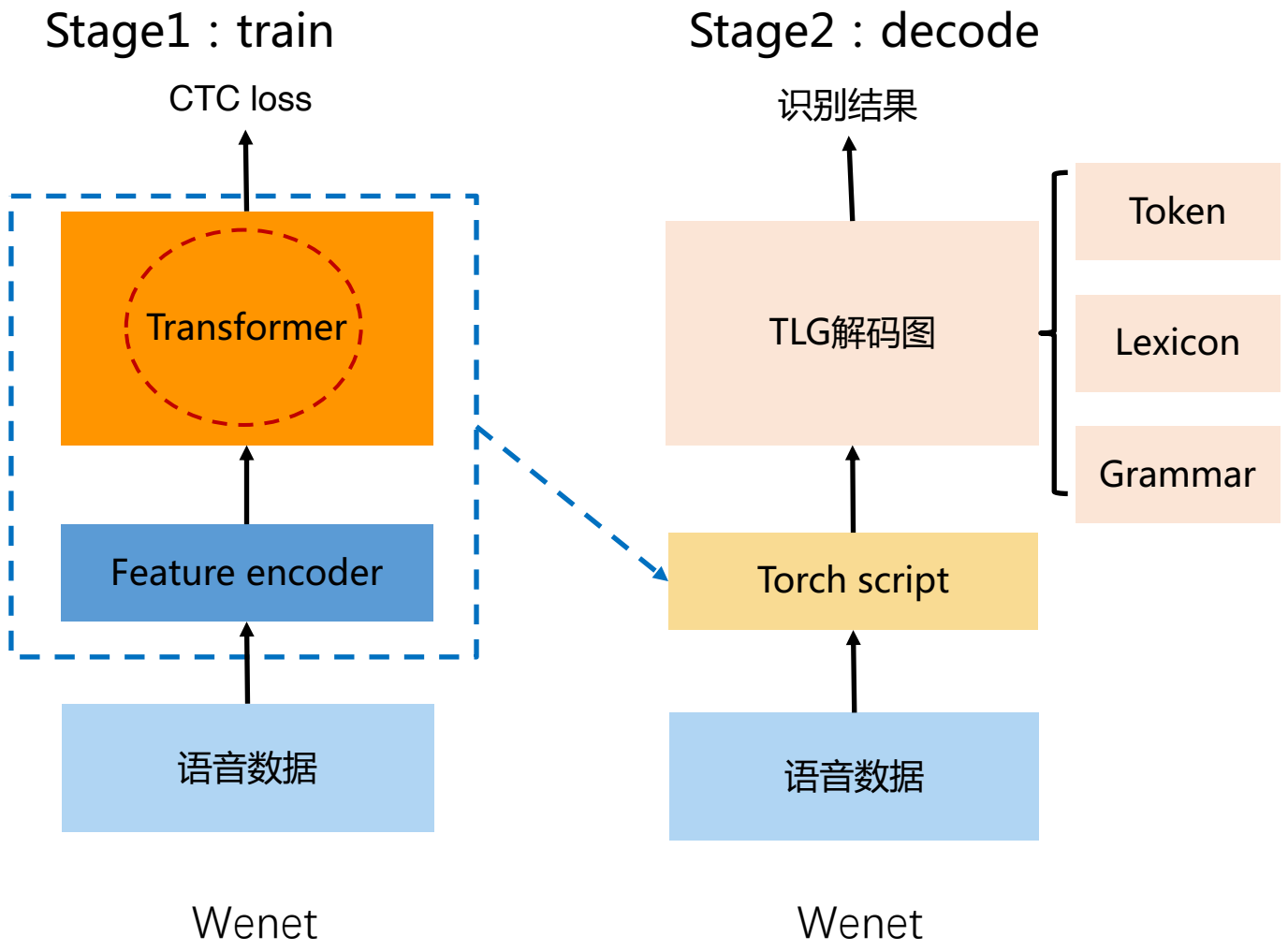
数据要求：

1. 人数尽量多，考虑年龄段和性别
2. 16k采样率，16bit

建模方案-基于预训练方法



建模方案-直接训练


Stage1：train
CTC loss
Transformer
Feature encoder
语音数据
Wenet
Stage2：decode
识别结果
TLG解码图
Token
Lexicon
Grammar
Torch script
语音数据
Wenet

建模方案-两个对比

训练方法	训练工具	特点
From-scratch	Wenet https://github.com/wenet-e2e/wenet	数据量小的话准确率不高 工程化方便，有提供客户端/服务端部署代码
预训练	Fairseq https://github.com/facebookresearch/fairseq	可以复用开源的预训练模型，准确率好 不支持工程化部署

服务部署

Runtime on WeNet

This is the runtime of WeNet.

We are going to support the following platforms:

1. Various deep learning inference engines, such as LibTorch, ONNX, OpenVINO, TVM, and so on.
2. Various OS, such as android, iOS, Harmony, and so on.
3. Various AI chips, such as GPU, Horzion BPU, and so on.
4. Various hardware platforms, such as Raspberry Pi.
5. Various language binding, such as python and go.

Feel free to volunteer yourself if you are interested in trying out some items(they do not have to be on the list).

Progress

For each platform, we will create a subdirectory in runtime. Currently, we have:

- [x] LibTorch: in c++, the default engine of WeNet.
- [x] OnnxRuntime: in c++, the official runtime for onnx model.
- [x] GPU: in python, powered by triton.
- [x] android: in java, it shows an APP demo.
- [ ] Language binding
  - [x] binding/python: python is the first class for binding.
  - [ ] binding/go: ongoing.

其他问题

是否考虑流式语音识别	→	流式网络结构 chunk transformer/lstm
是否考虑识别速度	→	模型蒸馏、模型量化
是否考虑热词修复	→	解码器改造
是否考虑普通话+方言	→	Multi-lingual Code-switch

Q&A