关键词唤醒和嵌入式系统

2019 年 1 月 6 日

如今在科幻电影或小说中，人们对未来世界的想象都是人和机器可以像人和人一样顺畅地交流，其实现在人机语音交互已经逐渐渗透到我们的日常生活中。设想这样的场景，当你满手油污在厨房做饭时，突然忘了一道菜式的做法，你再也不需要手忙脚乱地找手机，你只需要通过说“Hi，Siri”来启动手机，打开APP里的菜谱；当你在寒冬的周末享受温暖的被窝时，突然想享受余音绕梁的美妙，但又被寒冷束缚了手脚，别担心，现在你只需要对着语音助手说，给我放一首《快让我在雪地上撒点野》!

看，语音人机交互已经慢慢地改变了我们的生活方式，改变手动操作机器的旧习，让语音打破空间的距离。而这些技术是如何实现的呢？本章就来揭开语音交互第一步的神秘面纱——关键词唤醒。

关键词唤醒是语音交互的第一步，他就像机器的触发开关，而神奇的是这个开关将由人发出的声音来控制。机器不再是乱接话的“讨厌鬼”或是不爱搭理你的“高冷帝”，只要你说出正确的关键词，就会打开机器的话匣子，让机器像一个听话的助手一样，完成你赋予的任务。他让人类和的机器交流将变得更加灵活，更加可爱，更加人性。

1 什么是关键词唤醒

1.1 关键概念

什么是关键词唤醒，其实上面描述的两个场景已经解释了这个技术。**关键词唤醒**就是用户说出特定的语音指令——关键词，设备从休眠状态切换到工作状态，给出指定的响应。**关键词唤醒**被称为keyword spotting(简称KWS)，定义为在连续语音流中自动检测出关键词的任务[1]。应用场景涉及到生活的方方面面：手机的虚拟助手让手机从“手”中解放；智能家居让家居生活更加便利；车载语音控制让驾驶更加安全；机器人上搭配语音助手也许未来我们真的能收

获大白这样暖心的机器人伙伴。

关键词一般是系统设计时给定，比如苹果手机的“hi, siri”；天猫精灵语音助手的“天猫精灵”；亚马逊Echo 的“Alexa”；Google Home的”Hi，Google“等。当产品检测出相应指定的关键词时，该产品才能被唤醒。关键词的设计对系统的性能指标影响也较大。一般关键词设计为3到5个字，4个最佳。可以使用纯英文、纯中文或中英混合的关键词。关键词音节覆盖越多，差异越大，系统的稳定性越好。当然，未来的产品可能会设计成用户自定义关键词，这样对产品自身系统稳定性的要求将更高。

1.2 评价指标

评价一个关键词唤醒系统性能的指标主要有以下几个：

唤醒率 唤醒率即在测试过程中系统被正确唤醒的次数除以总的测试次数。被正确唤醒的意思就是当环境给出关键词时，系统响应并启动。显然，唤醒率越高，系统性能越好。

虚警率 虚警率即在测试过程中系统不该被唤醒的次数除以总的测试次数。不该被唤醒即环境未给出关键词但是系统错误地响应。虚警率越低，系统性能越好。

实时率 实时率即系统的反应速度。用户说出关键词，系统应当能快速反应，这样才能提高用户体验。如果关键词唤醒系统的反应速度像《疯狂动物城》里的那只树懒一样，估计没人会喜欢这样的产品吧。

功耗水平 低功耗是系统性能的一个重要指标。因为很多产品是在充电后使用的，因此这个指标和用户体验紧密相关。

1.3 方法流程

论文[1]中给出了关键词唤醒的具体方法并从数学角度给出了详细的阐述。可以将关键词唤醒分为两个阶段：第一个阶段是**检测阶段**。即系统收集关键词在给定的句子中的信息。定义$X \in \chi$是输入语音，其中χ 是所有语音信号的域（比如语音信号的声学特征向量）。为简化问题，我们假设最多只能有一个关键字能出现在句子中。定义$K \in \kappa$ 是我们需要检索的关键词，其中κ是关键词的域（例如关键词的文本格式），**检测阶段**需要学习一个从语音信号空间$\chi \times \kappa$到置信空间$\mathbb{R}^d$的映射，设参数为$\theta_1 \in \Theta_1$：

$$E_K = f_1(X, K; \theta_1) \tag{1}$$

其中$E_k \in \mathbb{R}^d$代表关键词K在给定语音信号X中出现的置信度。常用的向量置信度维度特征有关键词的后验概率$P(K \mid X)$，关键词的起始时刻t_{Kb}和结束时

刻t_{Ke}。因此

$$E_K = [P(K|X), t_{Kb}, t_{Ke}] \in \mathbb{R}^3 \tag{2}$$

其他附加的维度信息也包括关键词的先验概率$P(K)$，关键词的持续时长T_K等。第二个阶段是**决策阶段**。即系统根据检测阶段得到的置信向量E_K判断关键词k是否出现在语音X中，并给出关键词出现的位置。定义输出O_k 为三元组$O_k = (Yes/No, t_{Kb}, t_{Ke})$。**决策阶段**需要学习一个$E_K$到$O_k$的函数映射，设参数为$\theta_2 \in \Theta_2$：

$$O_K = f_2(E_K; \theta_2) \tag{3}$$

将上述两个阶段结合，关键词唤醒任务其实就是学习一个复合函数$f_{KWS} = f_2 \circ f_1$，参数为$\theta = (\theta_1, \theta_2) \in \Theta = (\Theta_1, \Theta_2)$:

$$O_K = f_{KWS}(X, K; \theta) \tag{4}$$

2 关键词唤醒和LVCSR

在前面的章节已经介绍了LVCSR（大词汇量语音识别任务）的定义和解决方法，我们来分析KWS任务和LVCSR任务的相关性。LVCSR任务是将一段连续语音识别成文字。该任务的HMM-GMM 系统的原理是为每个音素进行建模，将输入语音的每帧识别成音素的状态。LVCSR任务的DNN方法输入是每帧语音信号特征向量，输出是该帧对应的音素，本质上是一个多分类问题。KWS任务可以被视为LVCSR任务的子问题，不同的是KWS任务只需要在一段连续语音中提取关键字的信息。而实际上关键词唤醒问题的解决方法和LVCSR在一定程度上是相通的，这将在后文进行详细阐明。

3 关键词唤醒的难点

在主观感受中，关键词唤醒任务需要识别的只是几个关键词而已，为什么会划分为一个单独的问题进行研究。因为关键词唤醒问题在实际产品落地中有很大的挑战，其中最主要的问题是低功耗和高计算需求的不平衡。现实应用关键词唤醒任务的产品如智能手机、智能音箱等都是计算量不大的低端芯片，且大都是电池供电。这就需要系统尽可能减小运算量和降低功耗，但同时不能降低系统的稳定性。这就是关键词唤醒系统设计中的难点所在。

4 模型方法

4.1 Query-by-Example方法

Query-by-Example方法[2]是解决关键词唤醒问题最早尝试的方法之一。顾名思义，query-by-example方法是一种将关键词当作example，将听到的语音（query）和关键词（example）进行比对的方法，这是一种基于模板匹配的方法。具体实现包括两个步骤：

(1)训练步骤

先选择合适的方法对将语音格式的关键词表示为易于比较的模板格式（例如特征向量或网格形式等），完成对关键词的建模。

(2)测试步骤

将待检测的目标语音转化成和模板相同的格式，然后和关键词模板进行匹配计算相似度。

在过去几十年对query-by-example方法的探索中，研究热点是如何选择合适的特征表示方法[3]对关键词进行建模，而模板比对主要采用动态时间规整（dynamic time warping）[4]的算法，它采用了动态规划的思想。在自动语音识别的孤立词识别任务中，也采用了类似query-by-example 的方法。将给定的词汇创建模板，将待识别语音和模板比对，相似度最大的即为识别结果。DTW方法在孤立词识别任务中也能取得较好的效果。query-by-example方法不再是KWS问题的主流方法，因此此处不再做过多展开。

4.2 Keyword/Filler隐马尔科夫系统

基于隐马尔可夫模型的keyword/filler系统[5]是目前的主流方法之一，它和语音识别中早期使用的HMM-GMM系统原理相似。其思想都是用HMM 模型对待识别的语音的子字单元进行建模。最大的不同是在解码器中，LVCSR方法的解码器中包含了词典中所有的单词，而关键词唤醒任务的keyword/filler系统则将语音分为关键词（keyword）和非关键词（filler）两类进行建模。如图1所示，keyword建模采用精细建模方法，在词级、音素级或状态级上对关键词进行建模；filler建模采用粗放建模的方法，对除关键词之外的任意词语和噪音进行建模。这样建模的好处是大大缩小了解码空间，使解码速度变快，提高系统的实时率。

如图2所示为基于HMM系统的keyword-filler的一个实例，来自于Amazon的文章[6]。该解码系统由keyword(alexa)路径和filler路径构成，filler路径又分为非语音（Non-speech,NSP）路径和语音（speech,SP）路径。对关键词采用三状

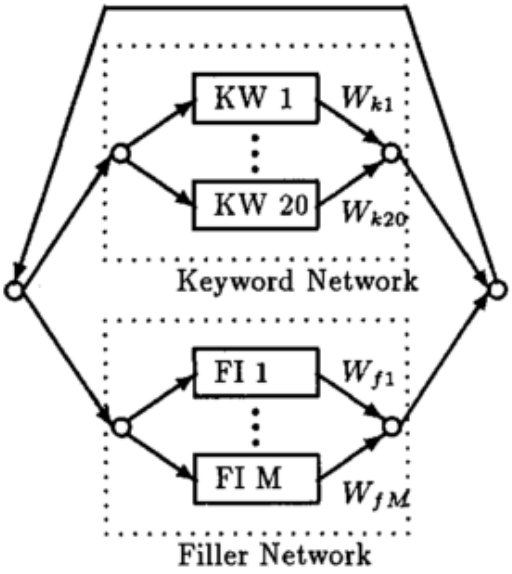

Figure 1: keyword-filler系统

态HMM对音素进行建模，而filler采用单状态HMM进行建模，有趣的是该隐马尔可夫模型的发射概率由TDNN进行建模。

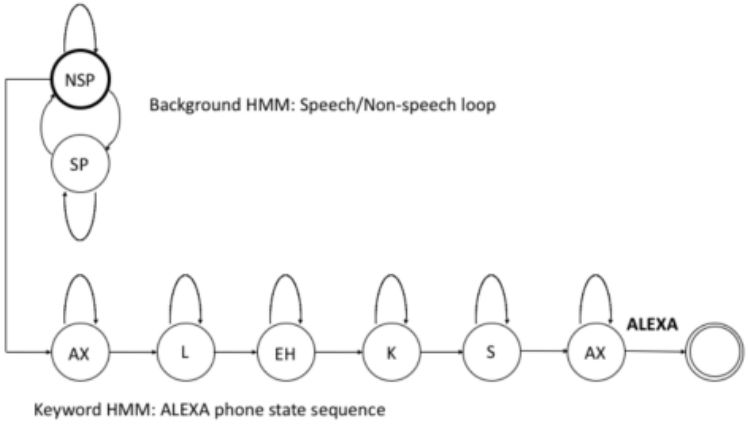

Figure 2: Alexa keyword-filler系统

4.3 LVCSR方法

一种简单粗暴的想法是直接用LVCSR语音识别系统解决关键词检索唤醒问题[7]。待识别语音通过LVCSR系统给识别为一个个单词或字，再进一步进行关键词检索。基于LVCSR系统方法的一个缺点是OOV问题。如果关键词不在词典中，那KWS问题将无法进行。基于LVCSR的关键词唤醒虽然能取得较好的效果，但是计算需求量大，在训练资源充足的时候系统稳定性较强，但是在移动设备上连续运行LVCSR系统通常是不切实际的。

4.4 Deep KWS端到端系统

目前最为流行的方法是基于神经网络的端到端关键词唤醒系统，在论文[8]中被首次提出。如图3所示为deep KWS系统的具体结构。该系统分为三个模块:

(1)特征提取模块。对原始语音信号进行特征提取，通常做法是提取每帧的特征向量。

(2)神经网络模块。输入是语音的特征向量。输出是关键词和非关键词的后验概率。

(3)后验值处理模块。由于神经网络输出的后验值是带有噪音的。在这一步骤中，对后验值以一定窗长进行平滑。平滑后的后验值超过一定的阈值，则认为被唤醒。

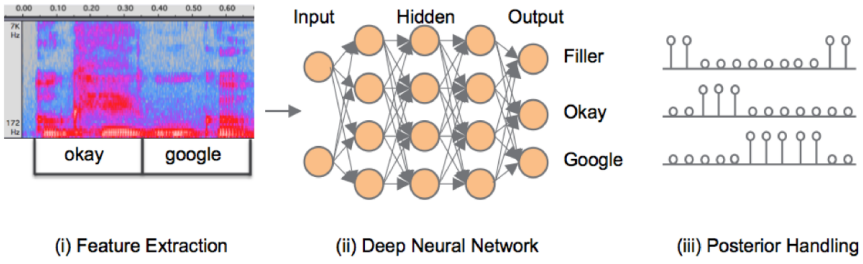

Figure 3: Deep KWS系统框架，从左到右依次为特征提取，神经网络分类器和后验值处理。

5 关键词唤醒和嵌入式系统

我们知道关键词唤醒系统的应用场景大都是计算能力不大的设备。比如可

穿戴设备，车载语音系统，家居语音助手等，在这种应用场景中，如何用深度学习的方法解决实际问题仍待完善。

目前在嵌入式设备上的关键词唤醒系统优化主要分为两类：一是通过减少神经网络参数量的方法提高系统在设备上的性能。二是针对硬件平台的优化。

5.1 减少模型参数量

5.1.1 模型裁剪

当训练好神经网络模型时，我们会发现有些神经元连接的权重值很小，在这种情况下可以认为两个神经元的关联不是很大，进行裁剪处理，从而得到较小的神经网络，减小网络模型的复杂度。

5.1.2 模型压缩

针对神经网络压缩常见的方法是SVD方法。通过奇异值分解将权重矩阵分解成两个较小的矩阵相乘，从而减小模型的计算量，提高系统在设备上的性能。

5.2 针对硬件平台的优化

目前嵌入式端设备的主流处理器有X86，ARM，MIPS等。当神经网络在服务器上训练好后，部署到嵌入式平台可以有针对性地使用运算加速库提高系统性能。常用的运算加速库有ATLAS,OpenBLAS,MKL等。

5.3 其他优化方法

在嵌入式系统工程应用当中，还采用了很多其他的优化方法。比如采用跳帧（subsampling）操作可以使速度提升2-3倍，同时准确率损失很小；在关键词唤醒系统中采用VAD过滤的方法，过滤非语音信号的音频，减少运算量的同时，还能降低系统的虚警率，提高系统准确性和灵敏度。

6 小结

本节介绍了关键词唤醒的概念、技术难点和解决方法。模型方法包括Query-by-Example通过DTW算法进行模板匹配的方法；Keyword/Filler系统基于隐马尔可夫模型对关键词和非关键词进行建模的方法；基于LVCSR的关键词唤醒方法以及Deep KWS端到端输出后验概率的方法。目前工程应用上多

采用Keyword/Filler系统和Deep KWS系统，在保证系统准确率的同时降低系统能耗并提升系统的反应速度。

关于关键词唤醒系统在嵌入式设备端的部署，主要介绍了两种方法。一是通过减少模型本身的复杂度降低运算量，二是通过运算加速库针对硬件平台进行优化，提高运算速度。

References

[1] Guoguo Chen. Low resource keyword spotting. *Department of Electrical and Computer Engineering Johns Hopkins University Baltimore, Maryland*, 2014.

[2] John S Bridle. An efficient elastic-template method for detecting given words in running speech. In *Brit. Acoust. Soc. Meeting*, pages 1–4, 1973.

[3] Petr Fousek and Hynek Hermansky. Towards asr based on hierarchical posterior-based keyword recognition. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 1, pages I–I. IEEE, 2006.

[4] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49, 1978.

[5] Richard C Rose and Douglas B Paul. A hidden markov model based keyword recognition system. In *Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on*, pages 129–132. IEEE, 1990.

[6] Ming Sun, David Snyder, Yixin Gao, Varun Nagaraja, Mike Rodehorst, Sankaran Panchapagesan, N Ström, Spyros Matsoukas, and Shiv Vitaladevuni. Compressed time delay neural network for small-footprint keyword spotting. In *Proc. Interspeech*, pages 3607–3611, 2017.

[7] John S Garofolo, Cedric GP Auzanne, and Ellen M Voorhees. The trec spoken document retrieval track: A success story. In *Content-Based Multimedia Information Access-Volume 1*, pages 1–20. LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE, 2000.

[8] Guoguo Chen, Carolina Parada, and Georg Heigold. Small-footprint keyword spotting using deep neural networks. In *ICASSP*, volume 14, pages 4087–4091. Citeseer, 2014.