基于多模态结构的情感识别

赵鑫玮

Overview

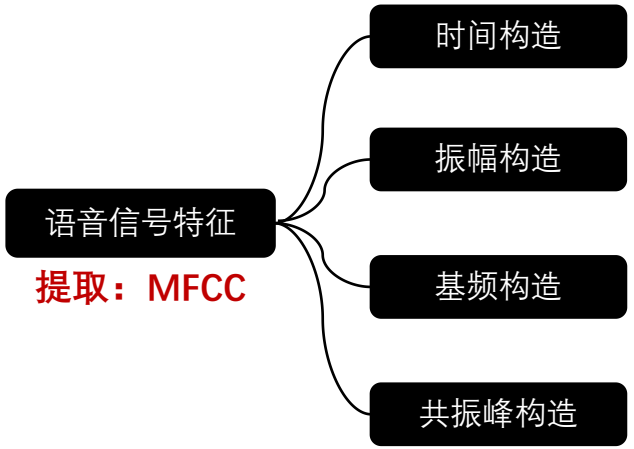
1. Background
2. Related Technologies
    a) Mel Frequency Cepstral Coefficients (MFCC)
    b) Global Vectors for Word Representation (GloVe)
    c) Attention
3. Current Work
    a) Overall Structure
    b) Text branch details
    c) Audio branch details
    d) Result
4. Dataset Intro

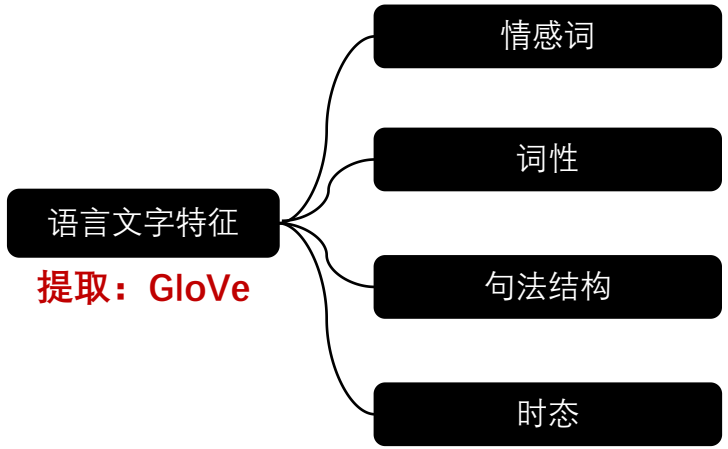
Background

计算机对从传感器采集来的信号进行分析和处理，从而得出对方正处于的情感状态，这种行为叫做情感识别（Emotion Recognition）。

情感识别的目的在于及时地、准确地通过一定的“情感表达”方式，一方面向他人展现自己的价值关系，另一方面了解和掌握对方的价值关系，才能够在此基础上，分析和判断彼此之间的价值关系，做出正确的行为决策。

目前对于情感识别有两种方式，一种是检测生理信号如呼吸、心律和体温等，另一种是检测情感行为如人脸的情感识别、语音语调的情感识别、语言文字的情感识别。





MFCC: Mel Frequency Cepstral Coefficients

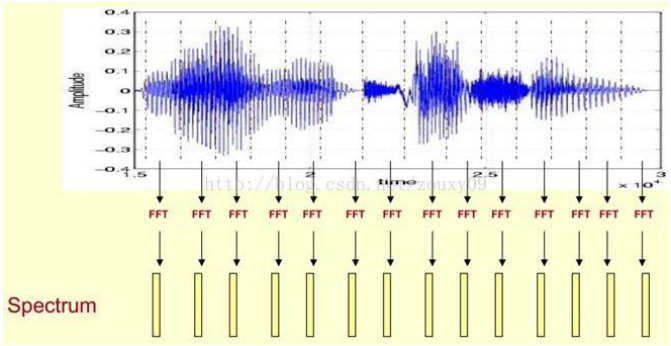
语音识别的第一步是特征提取，也就是提取语音信号中有助于理解语言内容的部分而丢弃掉其它的东西。

区分语音的关键就是声道的不同形状。不同的形状就对应不同的滤波器，从而产生了不同的语音。如果我们可以准确的知道声道的形状，那么我们就可以得到不同的语音的表示。声道的形状体现在语音信号短时功率谱的包络(envelope)中，因此好多特征提取方法需要准确的表示包络信息。

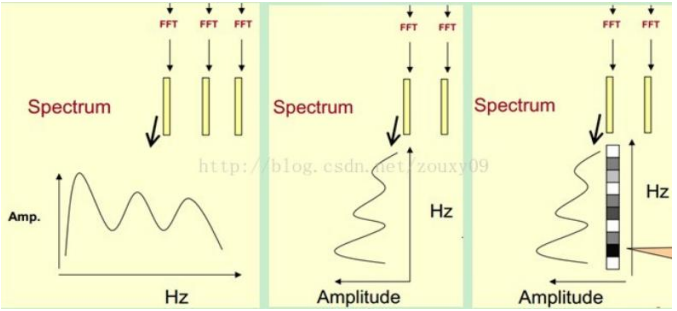
Mel频率倒谱系数就是其中的一种方式。此外，Mel频率是基于人耳听觉特性提出来的，它与Hz频率成非线性对应关系，也因此更贴合实际。由Mel频率倒谱系数计算得到的Hz频谱特征，已经广泛地应用在语音识别领域。

https://blog.csdn.net/zouxy09/article/details/9156785/

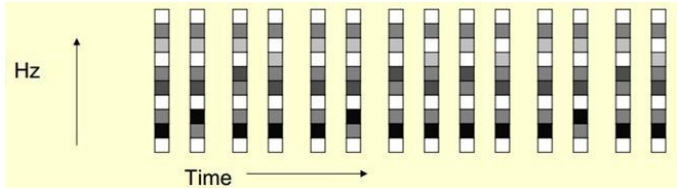
MFCC: Mel Frequency Cepstral Coefficients



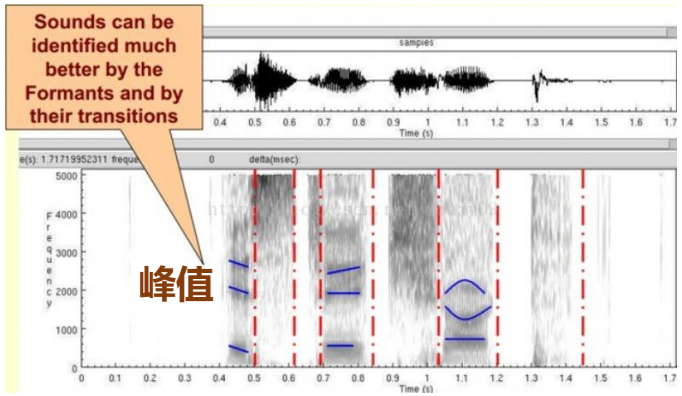
一段语音被分为很多帧（20-40ms），每帧语音都对应于一个频谱（通过短时FFT计算）



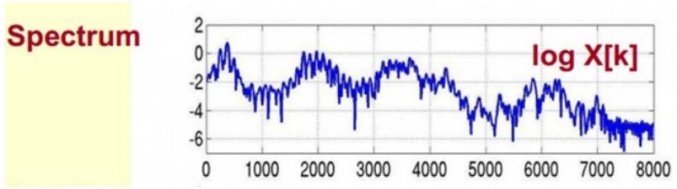
1. 先将其中一帧语音的频谱通过坐标表示出来
2. 将左边的频谱旋转90度
3. 把这些幅度映射到一个灰度级表示



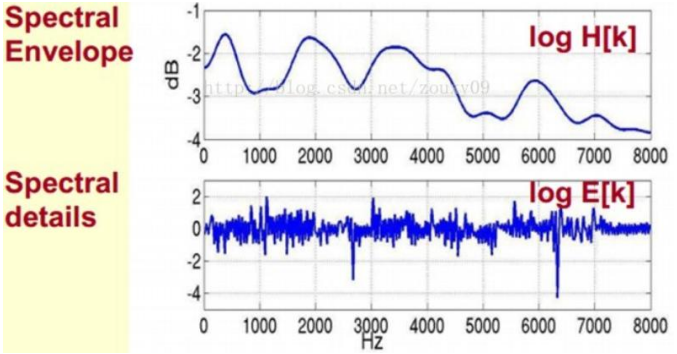
得到一个随着时间变化的频谱图



峰值（共振峰Format）表示语音的主要频率成分，携带了声音的辨识属性



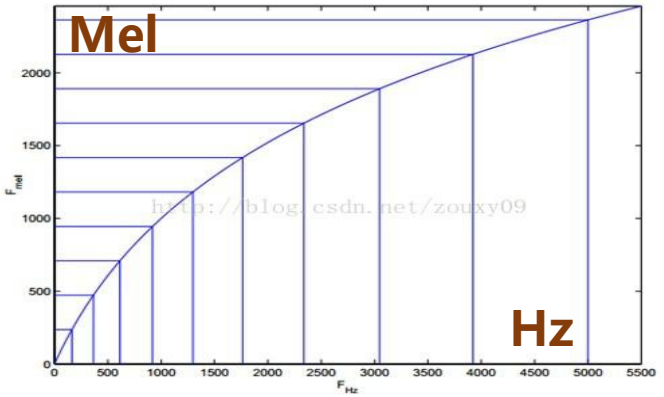
提取共振峰点得到频谱包络（Spectral Envelope）频谱曲线去掉包络曲线得到频谱细节（Spectral details）



包络

频谱细节

h[k]描述了频谱的包络，它在语音识别中被广泛用于描述特征



Mel频率分析基于人类听觉感知实验，实验观测发现人耳就像一个滤波器组一样，它只关注某些特定的频率分量，也就是说人的听觉对频率是有选择性的。

梅尔频率倒谱系数（MFCC）考虑到了人类的听觉特征，先将线性频谱映射到基于听觉感知的Mel非线性频谱中，然后转换到倒谱上。

https://blog.csdn.net/zouxy09/article/details/9156785/

MFCC: Computation Process

计算过程：

1）将原语音信号先进行预加重、分帧和加窗

2）对每一个短时分析窗，通过FFT得到对应的频谱：X[k]=H[k]E[k]；

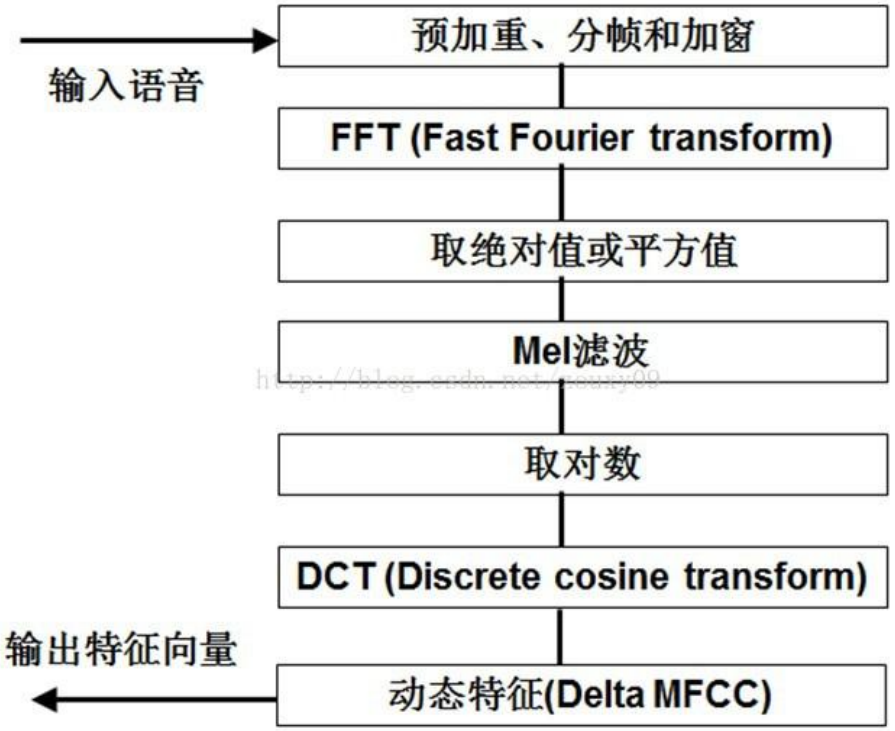
只考虑幅度：|X[k] |=|H[k]||E[k] |；

3）通过Mel滤波器组得到Mel频谱 $mel(f) = 2595 * \log_{10}(1 + f/700)$

4）在Mel频谱上面进行倒谱分析：

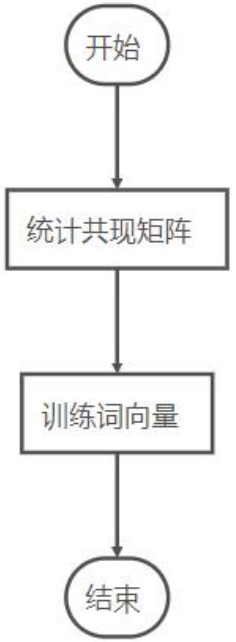
两边取对数：log||X[k] ||= log ||H[k] ||+ log ||E[k] ||→人类对于声音大小(loudness)的感受不是线性的。为了使人感知的大小变成2倍，我们需要提高8倍的能量，log这种压缩操作使得我们的特征更接近人类的听觉

两边取逆傅里叶变换：x[k]=h[k]+e[k]，通过DCT离散余弦变换来实现，取DCT后的第2个到第13个系数作为MFCC系数→因为后面的能量表示的是变化很快的高频信号，在实践中发现它们会使识别的效果变差。



https://blog.csdn.net/zouxy09/article/details/9156785/

GloVe: Global Vectors for Word Representation



模型目标：进行词的向量化表示使得向量之间尽可能多地蕴含语义和语法的信息。
输入：语料库
输出：词向量
方法概述：首先基于语料库构建词的共现矩阵，然后基于共现矩阵GloVe模型学习词向量。

设共现矩阵为 X，其元素为 $X_{i,j}$。
$X_{i,j}$的意义为：在整个语料库中，单词 i和单词 j共同出现在一个窗口中的次数(窗口为1)。

- I like deep learning.
- I like NLP.
- I enjoy flying.

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

$X_{like,I} = 2, X_{like,deep} = 1, X_{like,learning} = 0$

$X_i = \sum_{j=1}^{N} X_{i,j}$,共现矩阵中单词$i$那一行的和

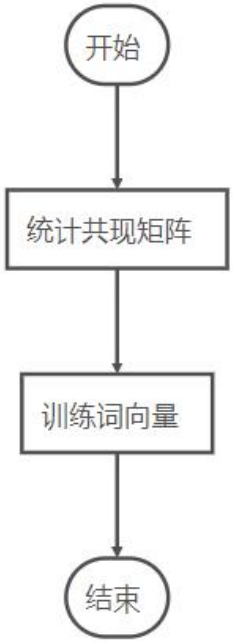
条件概率$P_{i,k} = \frac{X_{i,k}}{X_i}$，表示单词$k$出现在单词$i$语境中的概率

两个条件概率的比率$ratio_{i,j,k} = \frac{P_{i,k}}{P_{j,k}}$

一定程度上反映了单词i与单词j间语义和语法的相似性

https://zhuanlan.zhihu.com/p/58389508

GloVe: Global Vectors for Word Representation



模型目标：进行词的向量化表示使得向量之间尽可能多地蕴含语义和语法的信息。
输入：语料库
输出：词向量
方法概述：首先基于语料库构建词的共现矩阵，然后基于共现矩阵GloVe模型学习词向量。

https://zhuanlan.zhihu.com/p/58389508

$ratio_{i,j,k}$的值	单词j,k相关	单词j,k不相关
单词i,k相关	趋近1	很大
单词i,k不相关	很小	趋近1

假设我们已经得到了词向量，如果用中心词词向量u_i，u_j和语境词词向量v_k，可以通过某种函数计算出$ratio$，且计算出的$ratio$符合上表中的规律的话，那么我们得到的词向量就与共现矩阵有很好的一致性
将词向量$\mathrm{u}_i, \mathrm{u}_j, v_k$计算$ratio$的函数设为$g(\mathrm{u}_i, \mathrm{u}_j, v_k)$.

$$\frac{P_{i,k}}{P_{j,k}} = ratio_{i,j,k} = g(\mathrm{u}_i, \mathrm{u}_j, v_k).$$

如果用均方差作为代价函数，复杂度要$\mathrm{O}(N^3)$.

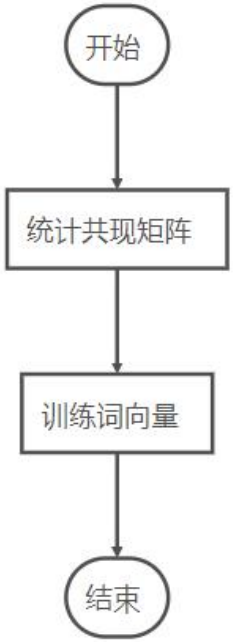
重新构建：$g(\mathrm{u}_i, \mathrm{u}_j, v_k) = e^{((\mathrm{u}_i - \mathrm{u}_j)^T v_k)}$.

即：$\frac{P_{i,k}}{P_{j,k}} = \frac{e^{(\mathrm{u}_i{}^T v_k)}}{e^{(\mathrm{u}_j{}^T v_k)}}$　　$P_{i,j} = e^{(\mathrm{u}_i{}^T v_j)}$.

代价函数：$J = \sum_{i,j}^{N} (log(P_{i,j}) - \mathrm{u}_i{}^T v_j)^2$.

复杂度$\mathrm{O}(N^2)$.

GloVe: Global Vectors for Word Representation



模型目标：进行词的向量化表示使得向量之间尽可能多地蕴含语义和语法的信息。
输入：语料库
输出：词向量
方法概述：首先基于语料库构建词的共现矩阵，然后基于共现矩阵GloVe模型学习词向量。

共现频率较低的词对，其相对重要性应该较低
因此需要为代价函数添加权重项

$$f(P_{i,j}) = \begin{cases} {P_{i,j}}^{\alpha} & if\ P_{i,j} < P_{max} \\ 1 & otherwise \end{cases}$$

代价函数：$J(\theta) = \frac{1}{2}\sum_{i,j}^{N} f(P_{i,j})(log(P_{i,j}) - u_i{}^T v_j)^2$

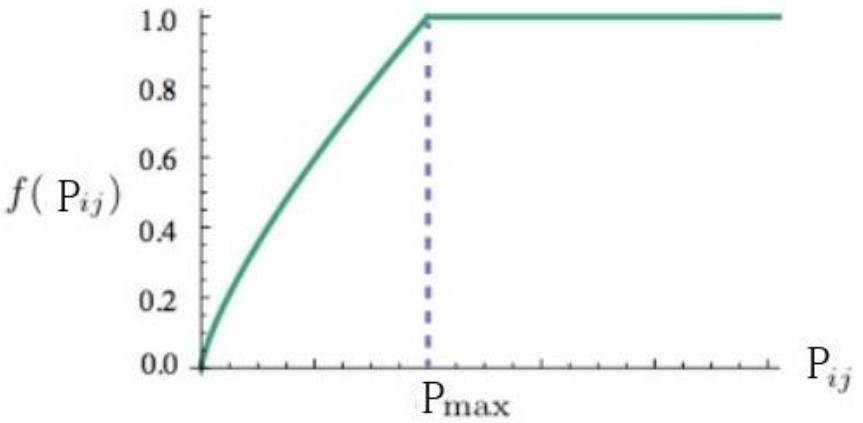
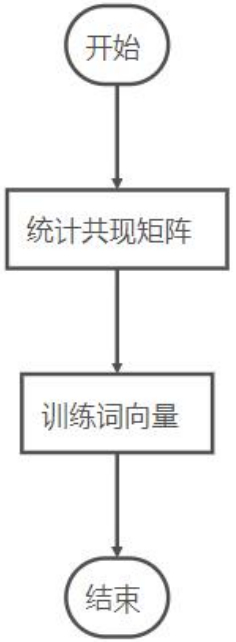


Figure 1: Weighting function f with $\alpha = 3/4$

https://zhuanlan.zhihu.com/p/58389508

GloVe: Global Vectors for Word Representation



模型目标：进行词的向量化表示使得向量之间尽可能多地蕴含语义和语法的信息。
输入：语料库
输出：词向量
方法概述：首先基于语料库构建词的共现矩阵，然后基于共现矩阵GloVe模型学习词向量。

最优化代价函数得到词向量矩阵 → 随机梯度下降（SGD）

1. 随机初始化
2. 对共现矩阵X中非零元素进行随机抽样
3. 在一次抽样得到的词对(i,j)，计算代价函数对于 u_i和v_j的梯度（记为Δu_i和Δv_j）

$$\frac{\partial_J}{\partial_{u_i}} = v_j * f(P_{ij})(u_i^T v_j - logP_{ij})$$

$$\frac{\partial_J}{\partial_{v_j}} = u_i * f(P_{ij})(u_i^T v_j - logP_{ij})$$

4. 每一次抽样得到的词对(i,j)，对U和V矩阵中的u_i列和v_j列进行更新（设定学习率为η）

$$u_i = u_i + \boldsymbol{\eta * \Delta u_i}$$

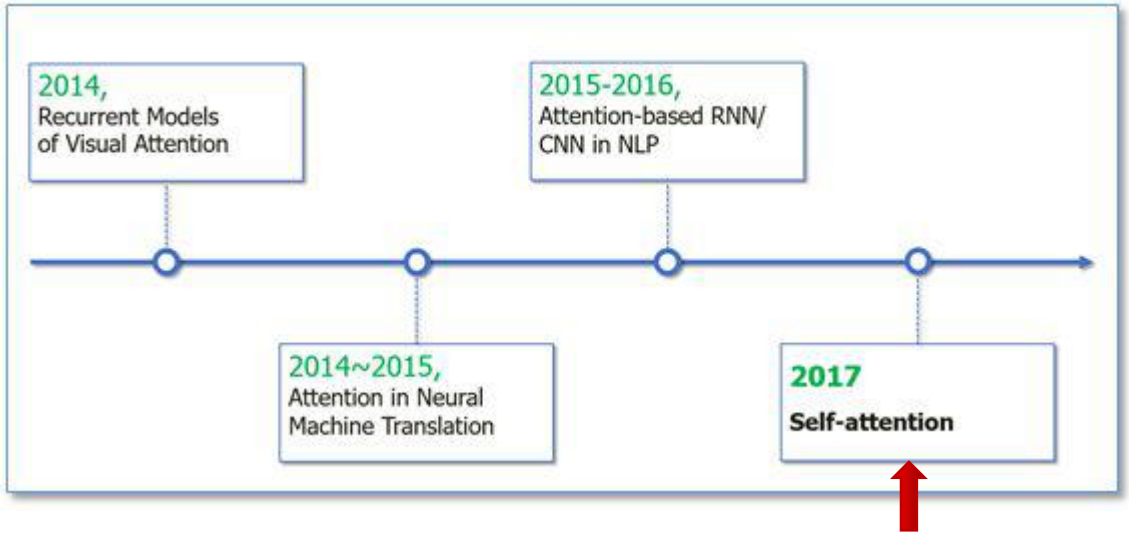
$$v_j = v_j + \boldsymbol{\eta * \Delta v_j}$$

5. 不断迭代随机抽样和U、V的更新，直至指定迭代次数（认为收敛），得到最终的词向量矩阵U和V
6. 将U和V对应位置元素相加（或求平均）得到最终的词向量矩阵

https://zhuanlan.zhihu.com/p/58389508

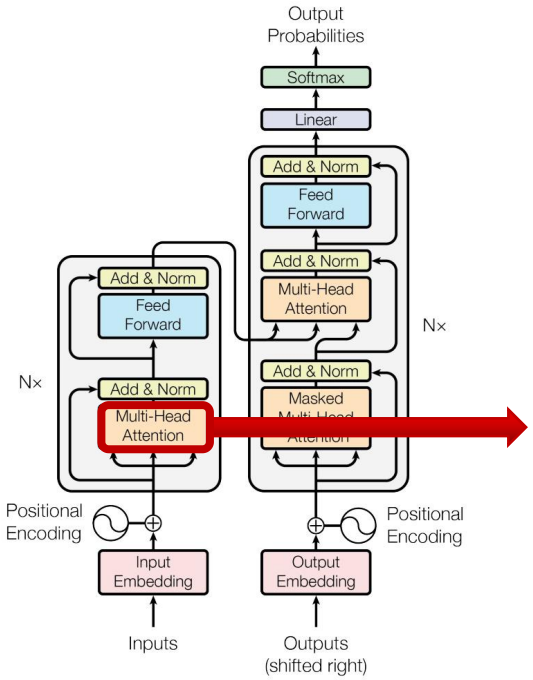
Attention: Attention is All You Need

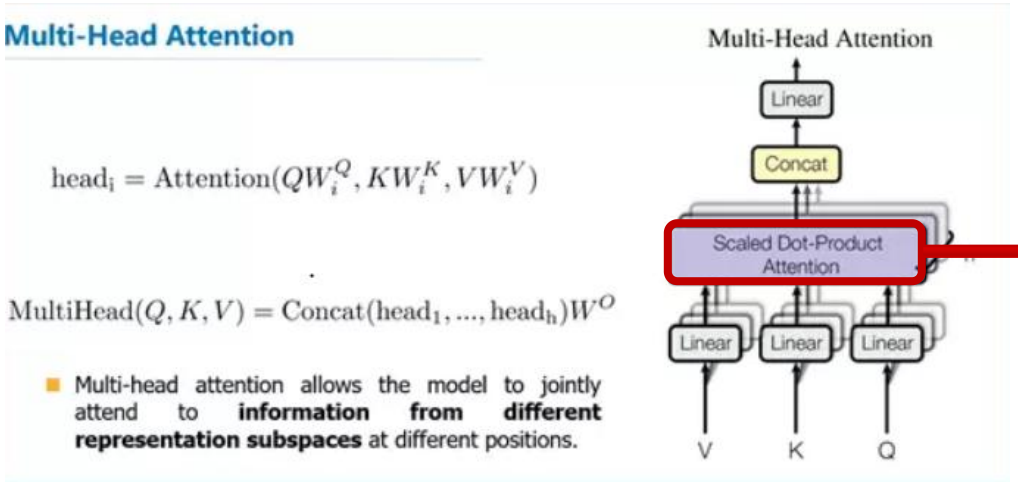
Attention机制是松散地基于人类的视觉注意机制，就是按照“高分辨率”聚焦在观察对象的某个特定区域并以“低分辨率”感知观察对象的周边区域，然后不断地调整聚焦点。这个概念最早出现在认知心理学上面，我们快读阅读或者读长篇文本的时候，我们的注意力是集中在关键词，事件或实体上。通过大量实验证明，将Attention机制应用在机器翻译，摘要生成，阅读理解等问题上，取得的成效显著。

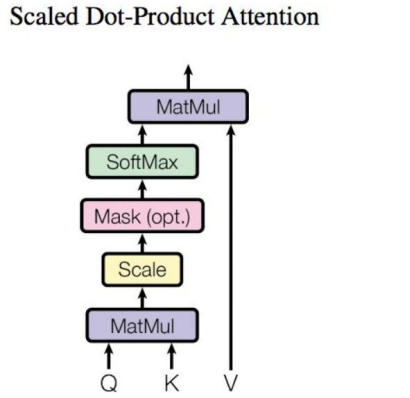


https://jalammar.github.io/illustrated-transformer/

Attention: Attention is All You Need



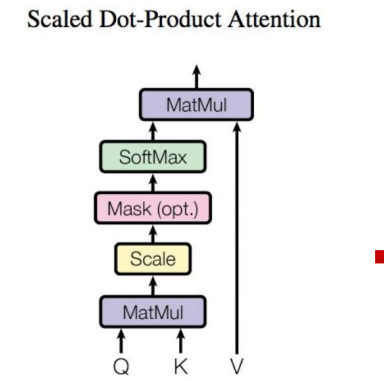




$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

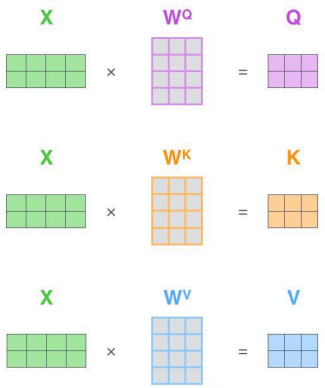
https://jalammar.github.io/illustrated-transformer/

Attention: Scaled Dot-Product Attention



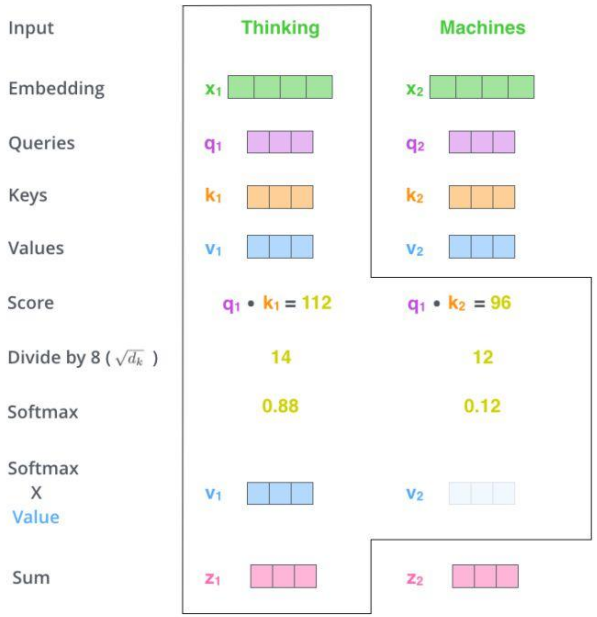
$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

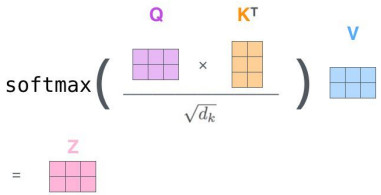
Q、K、V的计算过程：



X为输入向量；
W^Q、W^K、W^V 是最后需要求的权重矩阵，一般随机初始化得到。

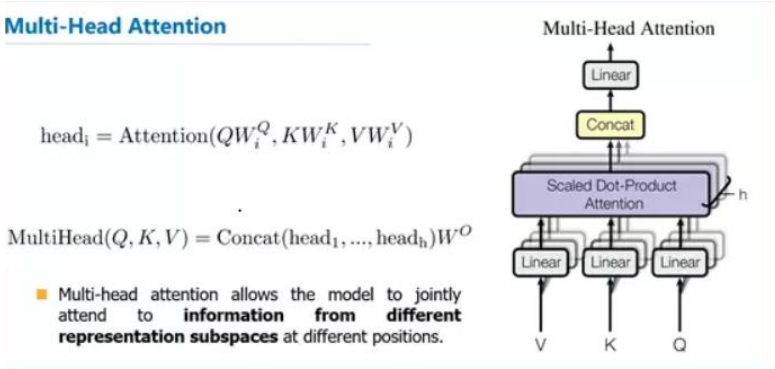
点积过程图解：



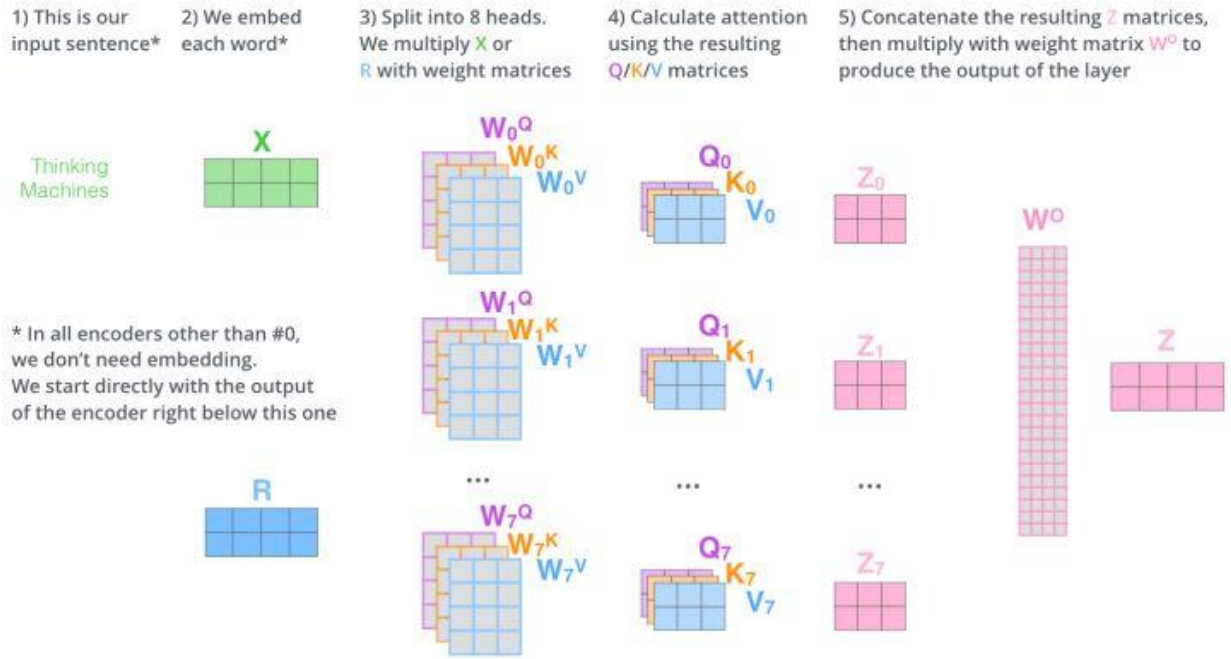


https://jalammar.github.io/illustrated-transformer/

Attention: Multi-Head Attention

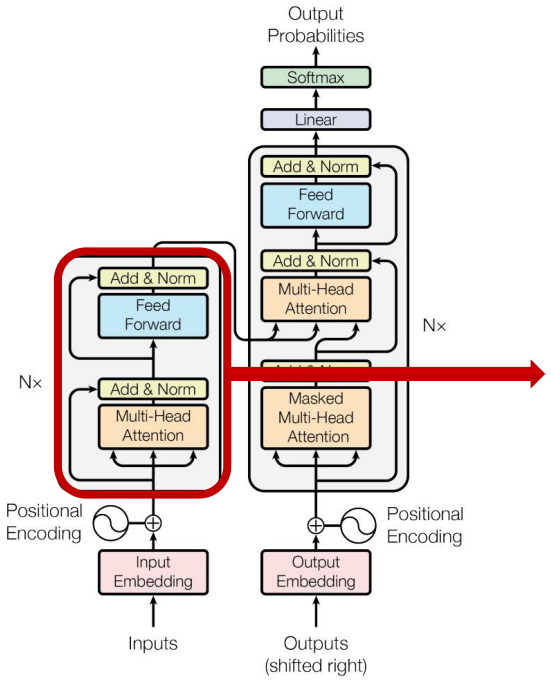


multi-head的实际意义是，将上述点积过程重复n次，n为head数量，最后将得到的结果汇总，过程如右图（n = 8）：

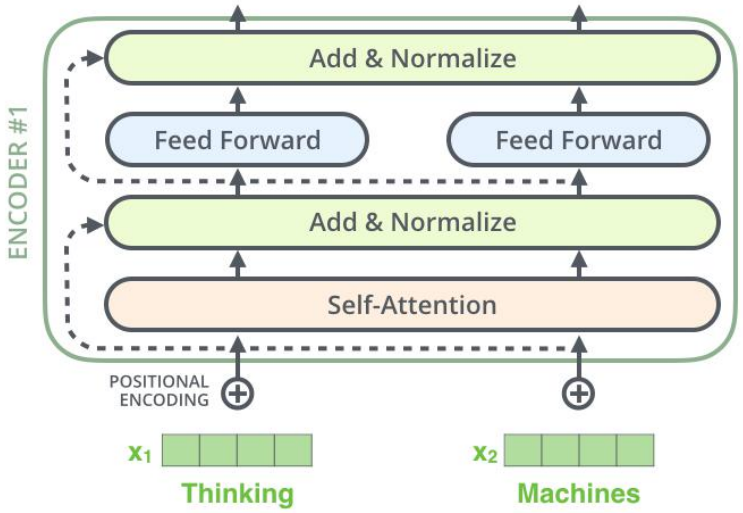


https://jalammar.github.io/illustrated-transformer/

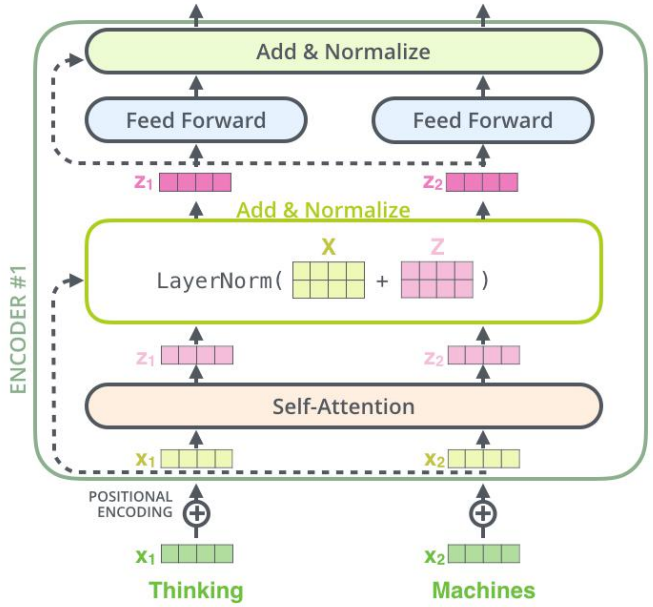
Attention: sub-layer



完整的Encoder-Decoder模型多应用于机器翻译领域，这里只截取其子模块作为应用。

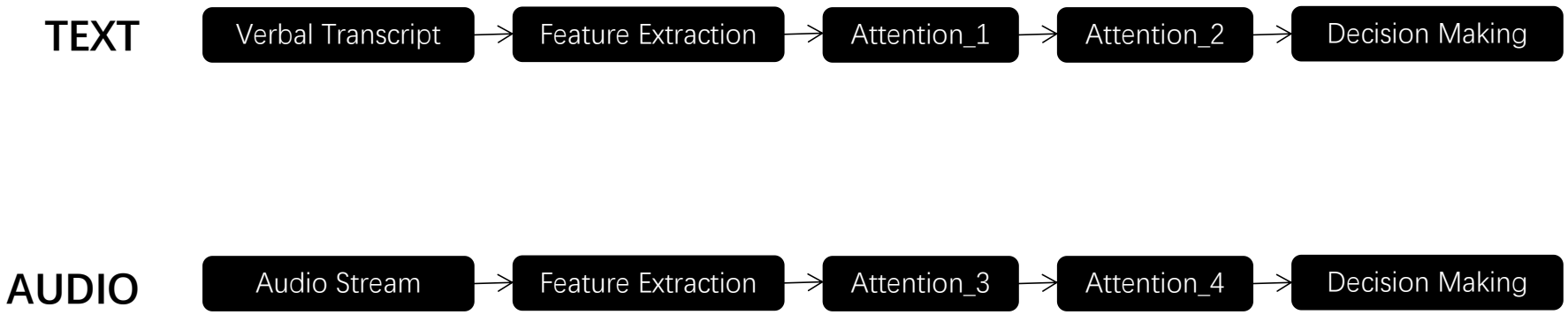


详细图解：

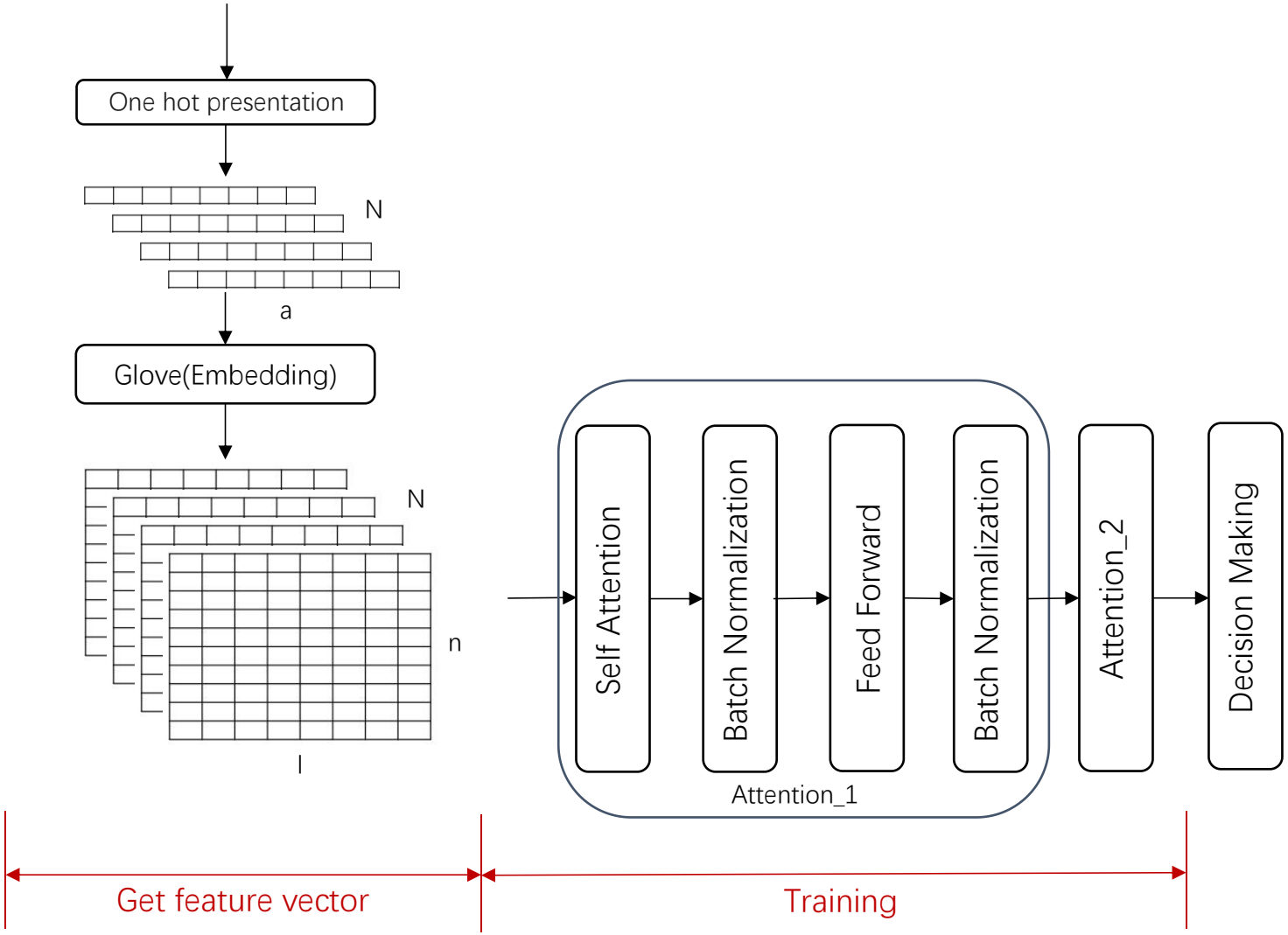


https://jalammar.github.io/illustrated-transformer/

Overall Structure

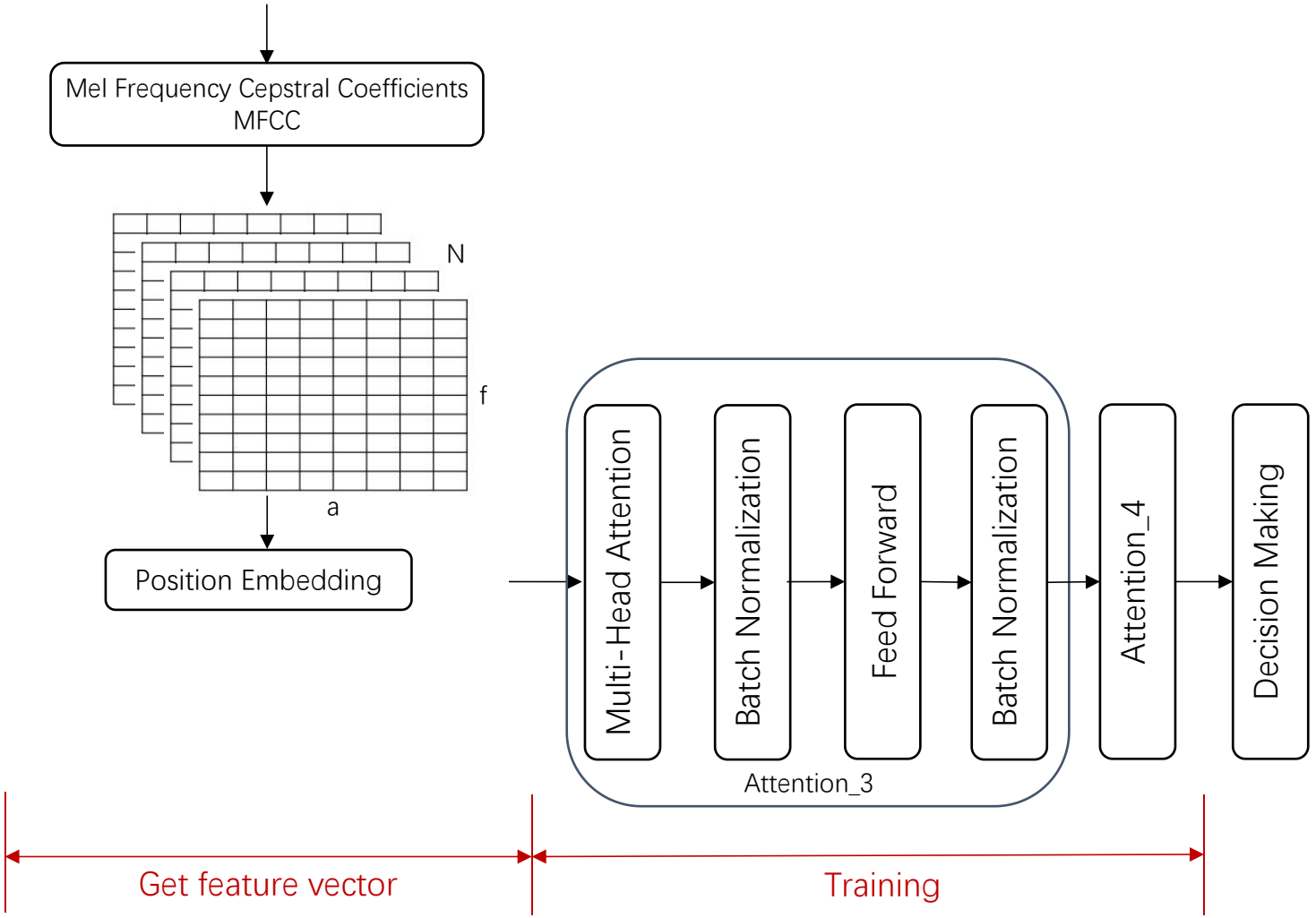


Text Branch Details



1. 寻找合适的句长定为l，句长应可包含绝大部分信息，可用正态分布来寻找。
2. 依据词频对每个单词进行独立编码，词频越高编码越大。
3. 设case数为N，经过上两步可以得到一个大小为（N * l）的Input向量，接下来需要对向量添加语义语法信息。
4. 使用Glove对词库进行训练，得到对应每个词的n维词向量
5. 使用Keras自带Embedding层对Input（N * l）向量做映射，初始权重设置为Glove获得的n维词向量矩阵，得到大小为（N * l * n）的特征向量。
6. 将特征向量放入Attention层进行训练，调整lr、batch_size等参数得到更好的训练效果。
7. 使用softmax对其进行分类，并使用test集进行验证，记录每个epoch的acc，loss值并画图分析。

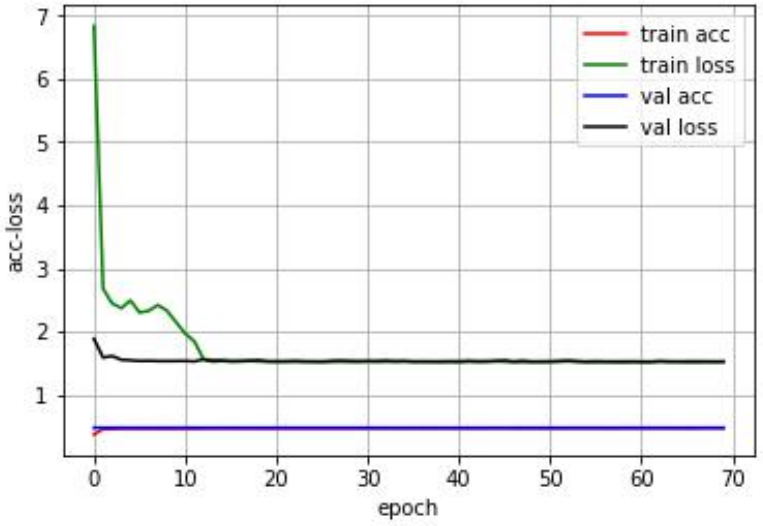
Audio Branch Details

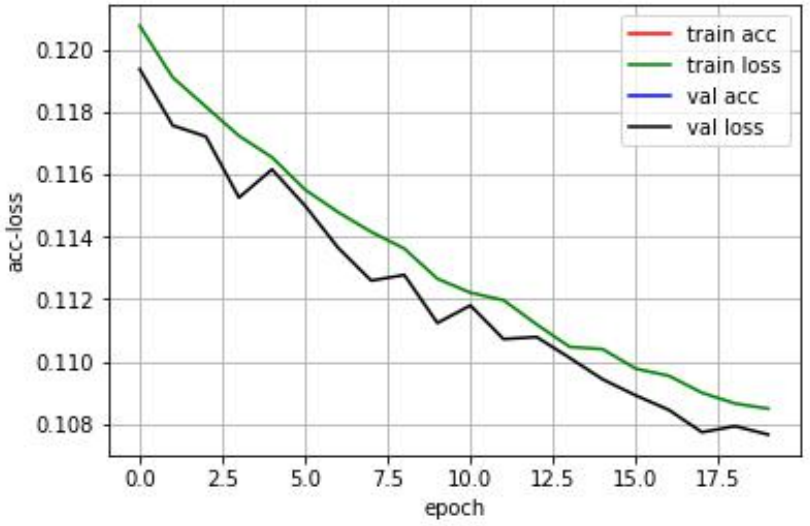


1. 将原音频通过MFCC，得到audio部分的Input向量，其大小为（N * a * f），其中N为case数量，a为帧数，f为频率
2. 进行Position Embedding，使向量包含位置信息，得到特征向量。
3. 将特征向量放入Attention层进行训练，调整lr、batch_size等参数得到更好的训练效果。
4. 使用softmax对其进行分类，并使用test集进行验证，记录每个epoch的acc，loss值并画图分析。

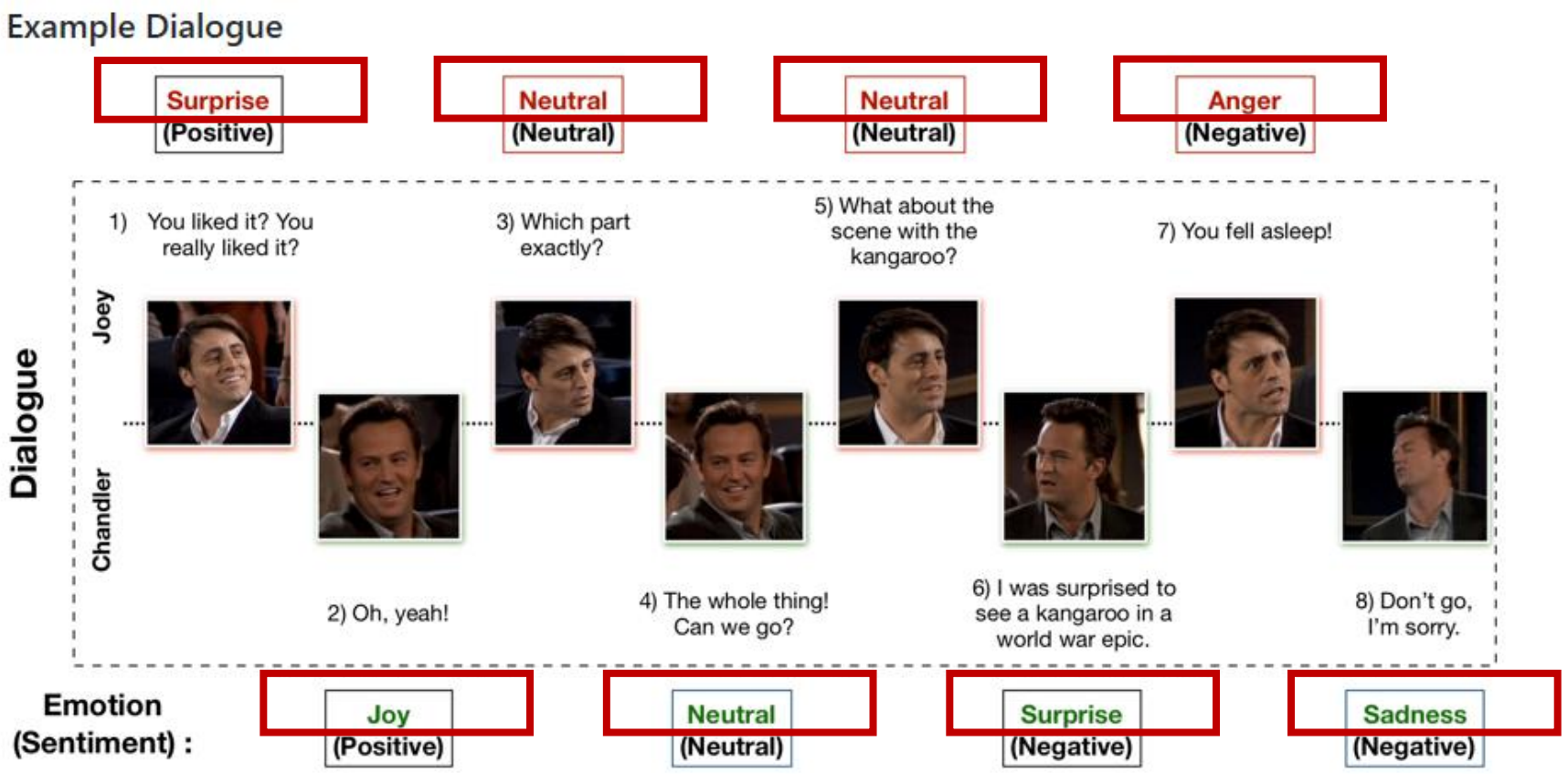
Result

Branch	Epoch	Loss	Accuracy	Val_loss	Val_acc
Text	70	1.5214	0.4753	1.5309	0.4812
Audio	20	0.1085	0.4593	0.1077	0.4812





MELD: A Multimodal Multi-Party Dataset for Emotion Recognition in Conversation



数据库：

针对情绪识别，从Friends TV提取的数据集。

共有1433段对话，包含13000语句，实验中用到的有12599句（9989+2610）。

MELD: A Multimodal Multi-Party Dataset for Emotion Recognition in Conversation

Column Specification

Column Name	Description
Sr No.	Serial numbers of the utterances mainly for referencing the utterances in case of different versions or multiple copies with different subsets
Utterance	Individual utterances from EmotionLines as a string.
Speaker	Name of the speaker associated with the utterance.
Emotion	The emotion (neutral, joy, sadness, anger, surprise, fear, disgust) expressed by the speaker in the utterance.
Sentiment	The sentiment (positive, neutral, negative) expressed by the speaker in the utterance.
Dialogue_ID	The index of the dialogue starting from 0.
Utterance_ID	The index of the particular utterance in the dialogue starting from 0.
Season	The season no. of Friends TV Show to which a particular utterance belongs.
Episode	The episode no. of Friends TV Show in a particular season to which the utterance belongs.
StartTime	The starting time of the utterance in the given episode in the format 'hh:mm:ss,ms'.
EndTime	The ending time of the utterance in the given episode in the format 'hh:mm:ss,ms'.

Text

Label

Dataset Distribution

	Train	Dev	Test
Anger	1109	153	345
Disgust	271	22	68
Fear	268	40	50
Joy	1743	163	402
Neutral	4710	470	1256
Sadness	683	111	208
Surprise	1205	150	281

9989 **2610**