4 *Self-Attention*

1. *Self-Attention* Background：

- *Iuput is a set of vectors*：输入向量的**个数、长度会改变**。
  - 文本处理，*token* 的*length*、*count* 均不一定相同
    - ***One-hot Encoding For Phrase***：丢失上下文联系
    - ***Word Embedding***：为每一个词汇生成一个向量。
  - *Graph*：邻接表 $\Longrightarrow$ 向量表示
- *Output*：
  - *Each vector has a lable*：标签(预测解雇)与输入一一对应，每一个输入的独立的样本，有一个对应的输出。
  - *The whole sequence has a lable*：*A set of vectors has one Output*：对一批输入样本，仅仅拥有一个输出。
    - ***Sentiment analysis*(情感分类)**：句子、段落的情感分类。输入转换后的一批*Word*，整体长度不一(总字数不一)，，一批中的向量个数不一(句子个数不一)
    - ***Sound Classification***：不在对帧分类，转而对某一区间、某一段随机、不定长的语音分类。
  - *seq2seq(Sequence to Sequence)*：完全无从了解输入与输出的数量关系对应，由机器自行决定。
    - *Translation*：词汇翻译
    - *Voice recognition*：语音辨识
- 在复杂任务中，*Input* 往往不能孤立的被视为一个个独立的向量，而是作为一个向量序列。以此为背景，学习输入为向量序列的情况下，*Deep Learning* 的输出类型及应用场景。
-

模型类型	*seq_len*	*batch*	*features*
MLP	固定	可变	可变
RNN	可变	可变	可变
Self-Attention	可变	可变	可变

- 对任意基于神经元的模型而言，一旦规定模型结构，*features*，即 X 的列数，都不可再更改。

- *RNN*：能处理任意长度的序列。是因为，在这类处理序列的时，数据首先按如下 *Tensor* 被准备 *(batch, timesteps, features)*
  - 其中，*timesteps* 代表序列的长度，如果使用过去30天的数据来预测下一天，则 $timesteps = 30$，而序列的长度一般选取不同样本长度中最长的(最长的句子)
  - 注意到：最终工程中，输入到神经元的 *Tensor* 仍旧需要是等长的，但是，提供 *Padding*，因此，兼容任意长度的 *Sequence*
- 本质上，都是通过 *Padding* 解决序列长度不一致。而关键点不在此，关键在于序列长度不一致，代表了单一向量直接进入 *Full Connection* 的效果很差，需要上下文信息。
- 关键点在于，是 *RNN* 与 *Self-Attention* 的架构决定了，*Padding* 的数值并不会对计算产生影响。

2. *Sequence Labeling*：为输入序列中的每个元素分配一个标签。

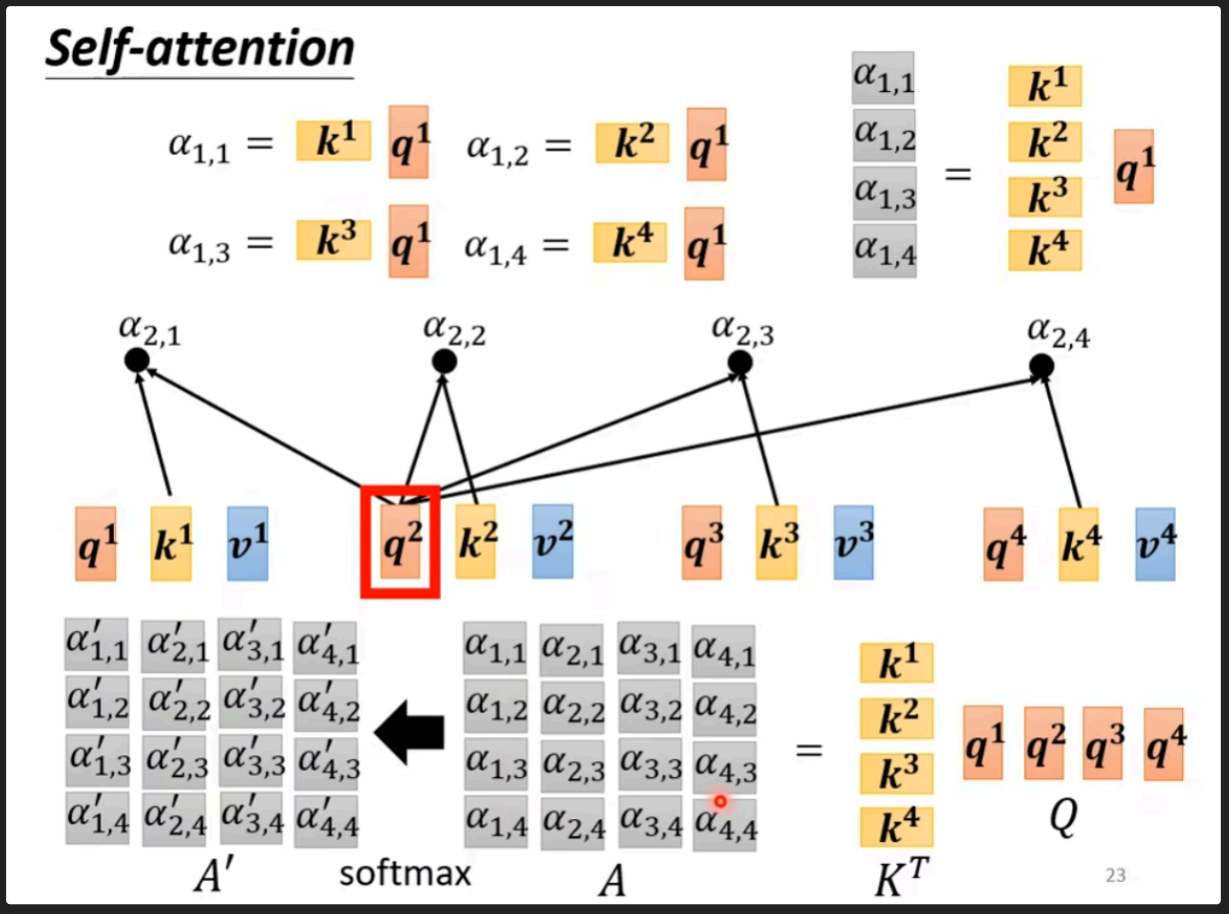
- 给定一个输入序列 $X = (x_1, x_2, \ldots, x_n)$，模型需要输出一个同长度的标签序列 $Y = (y_1, y_2, \ldots, y_n)$。
- *Method 1*：类似 *concat_nframes*，强制将上下文纳入本向量，并在测试时也这么做。
  - *Question*：*Window* 不足以囊括足够的上下文。
- *Method 2: Self-Attention*

3. *Self-Attention* 流程：

- *Input: All Network Input or Hidden Layer Output*
  - 可作为中间层、可混合使用、可搭配使用。对输入 *Sequence Vector* 的注意力处理
  - 对一个 *Sequence* 中的每个元素，称为 *Vector*
- *Background*：希望能够处理上下文信息，但不想将一个 *Sequence* 的全部信息一次性全部输入(包含在巨大的 *Window* 中)
- *Dot-product* (向量点积)：计算两个 *Vector* 的相关程度。
- *Query(Q)*：查询，以某一个 *Vector* 为基点，本向量想要查询什么？想要发现什么？
- *Key(K)*：*Vector* 携带什么信息？提供什么信息？
- $$\alpha_{1,2} = q^1 \cdot k^2 = (W^q \cdot a^1) \cdot (W^k \cdot a^2)$$

- $\alpha_{1,2}$：称为Attention Score
- 如此并行计算，再将Output 经过一个Softmax，就得到Vector 之间的注意力矩阵 $\Longrightarrow [\alpha'_{1,1},\dots,\alpha'_{4,4}] = A$
- Value(V)：$V_1 = W^v \cdot a^1$，Vector 实际携带什么内容。
- Finally Output：每个Attention 与Value 的乘积求和。

$$b_j = \sum_i \alpha'_{j,i} \cdot v^i \quad (j = 1..N)$$

  - b_1 代表 以Vector 1 为基点(Query(1) 1 发出查询)，查询其他Vector 的信息(Key(other))
- Self-Attention 矩阵表示
  - 将 $[a^1..a^4] = I$合并为矩阵，则不在分别需要每个Vector 的Q,K,V，而是统一为 W^q, W^k, W^v
  - $$(Q,K,V) = (W^q, W^k, W^v) * I$$
  - 

  - $$A' = Softmax(A) = Softmax(K^T \cdot Q)$$

- b_1 需要的是 $[\alpha'_{1,1},\dots,\alpha'_{1,4}]$ 因此，竖着摆放
- *Attention is all you need* 中写为 $A = Q \cdot K^T$ 实际上没什么区别，记忆即可。
  - 区别在于，工程中，假设 $batch = 1$，是以矩阵的每一行代表一个词、还是每一列代表一个词

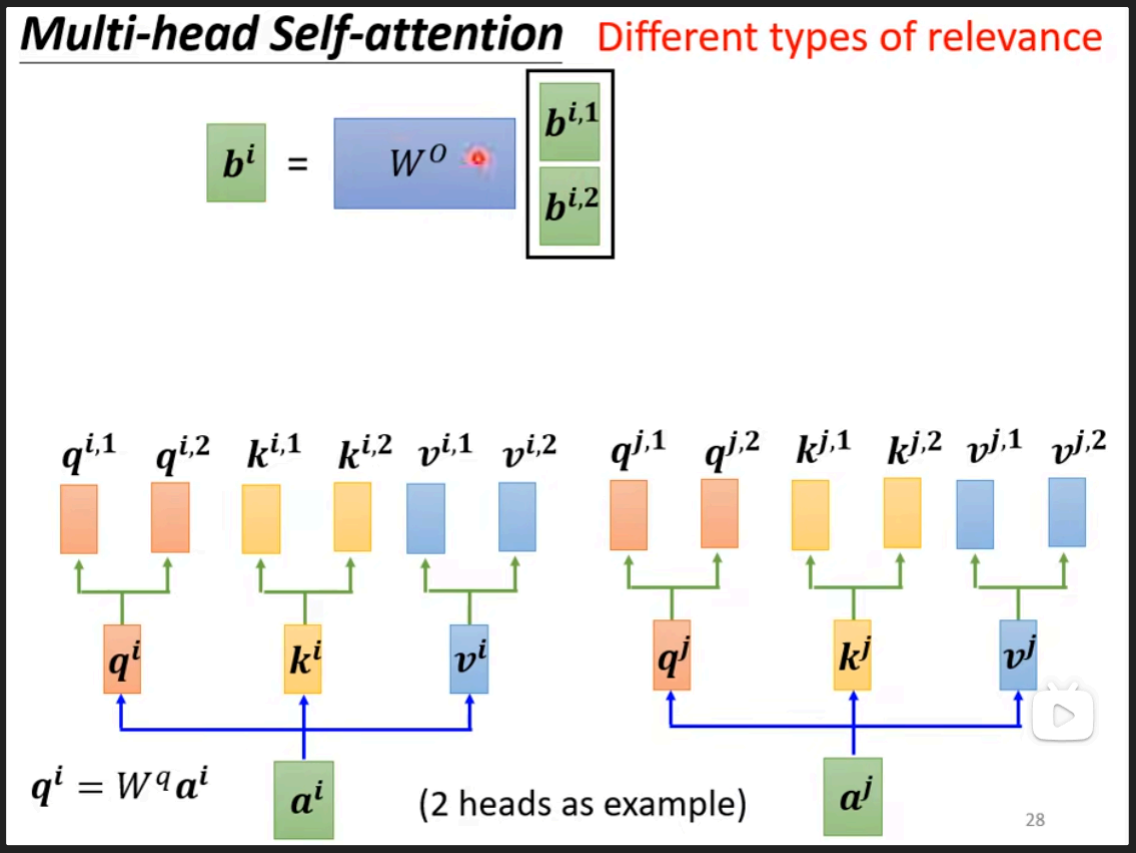
4. *Self-Attention* 细节

- *Positional Encoding*：经过上述*Self-Attention* 流程，可以获取上下文信息，但哪一部分是“上”，哪一部分是“下”？
  - *No position information*：没有位置关系。
  - 可选模块。*if* 位置信息有重要价值，如词性标注
  - 为每一个位置设计一个位置向量。*Unique positional vector* e^i，将其与原 *Input Vector* 相加。
- *Self-Attention* 代表：*Transformer*、*BERT*
- *Truncated Self-Attention*：
  - 语音辨识中，Sequence(一句话)，存在数量极其庞大的帧(*Vector*)，导致 *Attention Matrix* 也很庞大。
  - 仅仅考虑一个*Sequence* 中的部分 *Vector*
- *Self-Attention GAN*：用于图像处理。
  - 对比而言，*CNN* 卷积仅仅考虑*3×3*、*5×5* 范围，因此，可以说*Self-Attention* 是进阶版的*CNN*
- *Recurrent Neural Network(RNN)*：循环神经网络。
  - 无法并行处理、长记忆遗忘(记忆压缩)。
- *Graph Nerual Network(GNN)*：*Self-Attention for Graph*，图神经网络之一。
- *Self-Attention* 的流程，本质是动态加权求和的过程。他是一个集合处理器，输入输出为一对一的*Sequence*
  - *Input*：三维*Tensor*，形状为(*batch_size, sequence_length, feature_dim*)，例如(32, 50, 768)，一次批处理32个样本(序列)，一个序列的长度为50(50个词向量)，每个样本有768维特征。

    - *Process*：并行的进行*batch_size* 次，显然，*Self* 指代的是注意内部(自己)，因此是在序列内部。
- *Q,K,V Training*： 反向传播在海量数据上自学习
    - *Transformer* 带来的革命：*BERT*、*GPT* 等，利用 *Self-Supervised Learning*（自监督学习）技术，不需要标注数据，利用海量的，未经标注的原始文本。
    - *Mask Language Model(MLM)*：为模型设计任务，任务答案可以从文本自身中找到。将经过*Mask* 的文本交给模型，强迫其预测被遮蔽的部分。
- 显然*Q,K,V* 矩阵的大小是人为设计决定的。在*Testing* 时，面对某个*Mask* 的位置或将来的输出，*Model* 会输入整个*Sequence* ，并通过注意力机制最终输出一个概率向量，表示在该位置上，某个词汇的概率更高，模型并没有学会所有的“可能”，而是学会了所有的判断抽象规则。
- *Q,K,V* 的来源：一个精妙的类比，来源于信息检索。在搜索引擎搜索到过程中，一次操作的顺序是：
    - $\Longrightarrow$ 用户发出*Query*
    - $\Longrightarrow$ *DataSet* 中提供索引，利用大量*Key* 描述文档与网页
    - $\Longrightarrow$ 搜索引擎计算*Query*、*Key* 的相关性
    - $\Longrightarrow$ 根据*Top(Query, key)* ，取出对应的*Value*
- *Self-Attention* 的想象飞跃在于：如果一个句子本身，就是它自己的数据库呢？其流程，完全复刻了搜索引擎的查询流程。*Self* $=$ 本身
- 《*Attention Is All You Need*》这篇论文的伟大之处，不是“发明”了*Q,K,V*（这个概念在它之前已在其他论文中有所雏形），而是“意识到”仅仅用这个机制，就足以构建一个强大、高效、且完全并行的模型，从而彻底抛弃了*RNN* 的“循环”结构。
- *len(Q,k,V)* $==$ *feature_dim*

5. *Multi-head Self-Attention*（多头注意力机制）

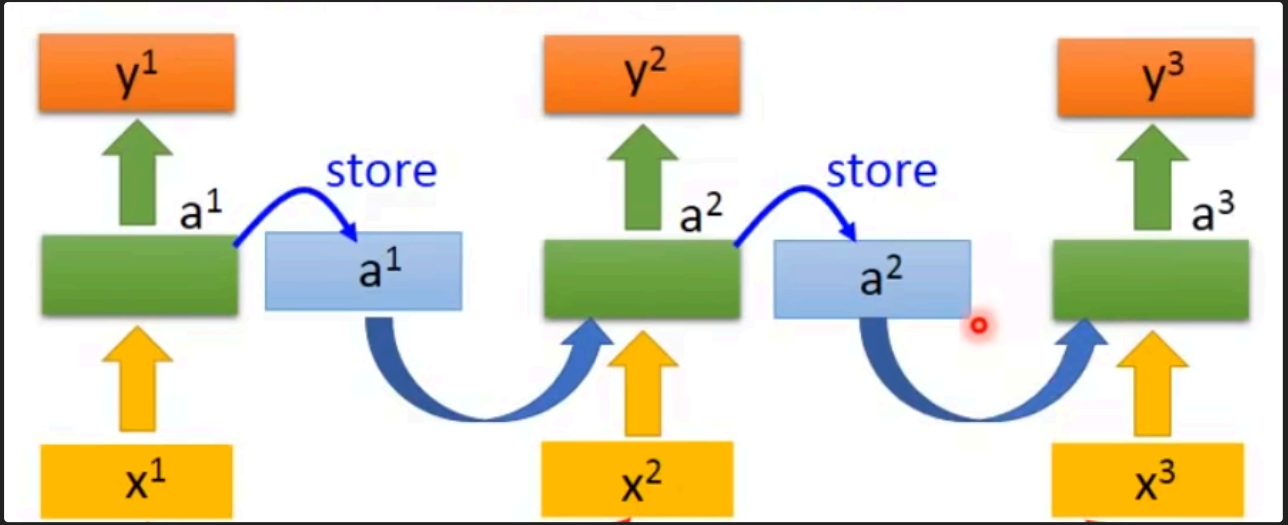
- *Q,K* 并不是以固定的算法，例如*Person*、*cos* 计算相似度，而是无监督学习训练出的矩阵 (W^q, W^k)，因此，单一模式的权重矩阵可能并不能够完全的表达所有维度上的相关性，因此，设计多组矩阵，以强化不同侧的注意力。
- (W^q, W^k, W^v) 不变，在得到 (q^i, k^i, v^i) 后，设计矩阵 $(W^{q,1}, W^{q,2})$ 以将 q^i 拆分。

- 拆分后的 $(q^{i,1}, q^{i,2})$，其中 i 为编号，同编号同组的继续进行 *Self-Attention* ，得到多组注意力矩阵，再将多组注意力 *Vector* 做矩阵 *Transform* ，得到结果，如下图
- 

- 多头注意力的多头是在算出大的（*Q,K,V*）之后拆分的
  - *Training* 阶段，拆分注意力头的意义就在于，类似 *CNN* 不同卷积核提取不同特征，多头注意力通过反向传播，迫使不同的头去学习不同的东西。
  - *Testing*：
  - 即使是在算出大的（*Q,K,V*）之后拆分，也完全不等价于直接使用（*Q,K,V*）进行计算。因为 *Softmax*
  - 在数学上而言，显然因为非线性，使得矩阵可组合性不再成立
  - 我们完全可以把 *Softmax* 就想象为 *NN* 的输出层，他是“做出选择”的层次。因此，拆分后进行 *Softmax* 等价于，负责注意不同部分的专家做出了自己独立的选择
  - 将“原始的相关性分数”转化为“百分比形式的注意力分配权重”。它是一个“软”（*Soft*）的选择机制，它不是生硬地“只选第 3 个”（*Hardmax*），而是允许模型以“*97%*的精力关注第*3*个，*2%*的精力关注第*1*个”的方式，进行平滑的、可微分的加权

6. *Recurrent Neural Network(RNN)*：循环神经网络。几乎被 *Self-Attention* 所完全替代，因此，这里只做简单了解。

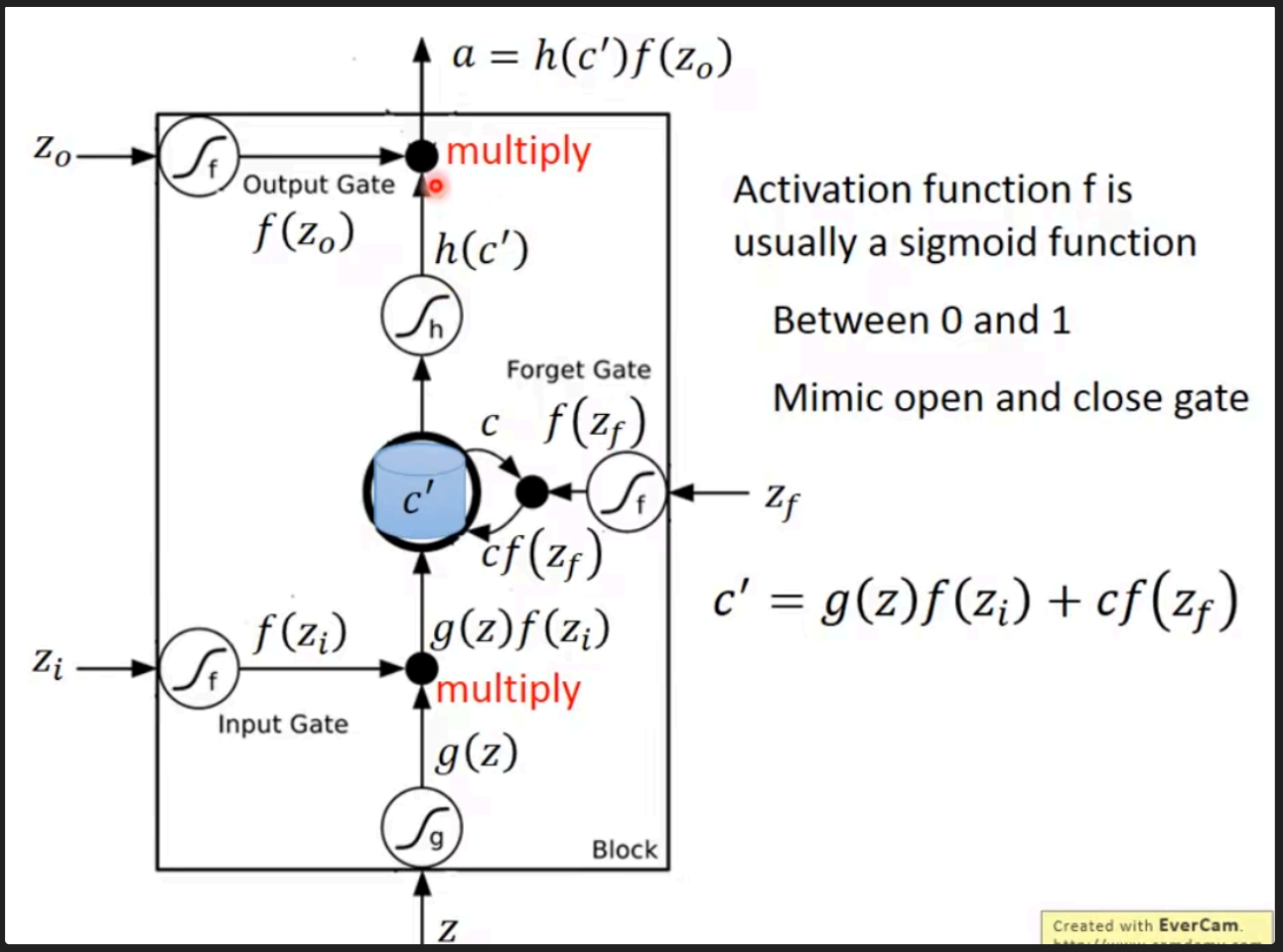
- 需要上下文信息，需要记忆力。完全相同的词汇在不同语句中的含义可能天差地别。
- 每一个 *Hidden Layer* 的 *Output* ，会被存储（*Elman Network*）
- 存储整个 *NN* 的 *Output*（*Jordan Network*）
  - 因此，随着 *Network* 的加深。所需 *Memory* 会愈发变大。
  - 或者，每新一次调用 *NN* ，旧的 *Memory* 被取出后， 直接清空该区域。因此， *LSTM* 扩写为 *LS-TM* ，是 长的“短期记忆网络”，这里的短期，就是因为会清空存储。



  - 同一个 *weight* 在不同的时间点被使用 *N* 次，因此， *RNN* 必定是串行的。
- *Bidirectional RNN*（双向循环神经网络）

7. *LSTM*（*Long Short-term Memory*）长短期记忆内存，也可称为长短期记忆递归神经网络。

- 针对 *RNN* 将一切信息全部放入 *Memory* 、 *Memory* 之间 相互孤立，仅能访问到邻近的内存，致使长记忆易缺失问题。
- *Input Gate*： 打开时，才允许 *Output* 存入存储。 *Gate* 打开与关闭的时机，是由 *NN* 自学习的。
- *Output Gate*： 同理，打开时才能被读取。
- *Forget Gate*： 哪个时刻，要“遗忘”一些记忆。

- 

8. *Graph Neural Networks(GNN)*：图神经网络
   - 以 *Graph* 图结构作为输入，以对图结构进行 *Classification*、*Regression* 。
   - 需要考虑 *Graph* 关系的情景。例如需要 *NN* 认识到人物关系网络、公交线路网络、地铁网络等。
   - *Question 1*：节点数据过大，无法利用全部数据、且无法标注全部数据。
   - *Spatial-based GNN*（基于空间的图神经网络）：
     - 类似 *CNN* ，利用某节点的邻居做“卷积”
     - $Layer_i \ =(Convolution)\Rightarrow \ Layer_{i+1} \ =(Readout)\Rightarrow$ Output
       - *Readout*：聚合所有 *Layer* 的结果的方法统称。不同的 *GNN* 有不同的实现方法。
     - 显然，任何 *Convolution* 的方法，都会将节点本身的特征平均（削弱），而随着 *NN* 的加深，所有节点的特征向量都趋向于收敛到同一个值。个体的独特性消失了。模型学不到任何有用的信息，因为每个节点看起来都一样。
     - *Graph Attention Networks(GAN)*：图注意力神经网络，令节点“注意”，而不是简单的平均。

    - 更进一步的，提出 *Graph Transformer* ，其认为在一个图里，任意两个节点都可能存在我们未知的“隐藏关系”。在 *GNN* 的基础上，完全应用了 *Self-Attention*

- *Spectral-based GNN*（基于频谱的图神经网络）：
    - 理论研究。传统信号（如音频）：它是在“时间域”上定义的。傅里叶变换（*Fourier Transform* ）能把它分解到“频率域”，让我们看到这个声音是由哪些“高频”和“低频”的波（正弦/余弦）组成的。
    - *Spectral-based GNN* 的设想是：我们是否也能对这个“图信号”做傅里叶变换？
    - 答案是可以的。这需要一个关键工具：图拉普拉斯矩阵（*Graph Laplacian* ）。
    - 这个设想在数学上无比优美，但在工程上几乎是一个灾难。
        - 需要计算拉普拉斯矩阵的特征分解，复杂度为 $O(N^3)$。
        - “滤波器”不具备“局部性”：在谱域中学习的“滤波器”是全局的。它不像 *CNN* 的 3x3卷积核那样只看局部。它是一个作用于整个图的“频率”调节器。这使得它很难捕捉到局部的、细微的模式。
        - “滤波器”不具备“迁移性”：拉普拉斯矩阵的“特征向量”（即傅里叶变换的“基”）是完全依赖于图的结构的。你训练好的图，哪怕只多加一条边，整个拉普拉斯矩阵都会改变，所有的特征向量都会改变。模型不具备泛化能力。
    - 突破：*Graph Convolutional Network* ，也就是 *Spatial-based GNN*（基于空间的图神经网络），*GCN* 的作者通过一系列精妙的数学推导，证明了：在谱域中进行滤波的复杂操作，可以在空间域中被一阶近似为一个极其简单的操作：“聚合邻居（和你自己）的信息，取平均，再过一个线性层。” $==$ *Spatial-based GNN*
    - 由此，*GNN* 的发展历程闭环了

9. *Word Embedding* ：基于 *Unsupervised Learning*

- *1-of-N Encoding* ：独热编码。*Word* 之间无关，丢失上下文信息。
- *Word Embedding* ：将每一个 *Word* 映射到高维空间中（维数远低于 *one-hot* ），机器自行学习、映射。
- *Auto-Encoder* ：设计一个沙漏型的神经网络，中间那个极其狭窄的瓶颈层（*Bottleneck* ），就是降维的发生地。强迫神经网络丢弃一切无关紧要的细节。基于 *MLP* 。
    - 非线性的、数据驱动的降维方式，能够捕捉到比 PCA 复杂得多的特征组合。
    - 更进一步的，提出 *VAE* ，变分自动编码器。

    - ⟹ *Representation Learning*（表征学习）
- *Word Embedding* 实现
    - *Count-based* ：基于频度（计数），同一文章（窗口）中，w_i, w_j 同时出现（*co-occur*）的次数。对该次数矩阵 *X* 做截断奇异值分解（SVD），用分解矩阵的 *U* 作为词向量矩阵。

$$W_{embed} = U_k \Sigma_k^{1/2}$$

        - 词向量的本质就是共现矩阵的低秩近似，寻找一个低维子空间，尽可能多地保留原始计数矩阵中的方差信息.
    - *Prediction-base* ：基于 *NN* ，*Train* 一个神经网络，使其能够根据上下文预测缺失的单词，*NN Output* 下一个单词的概率。
        - 将其中（某一层 *Hidden Layer* 作为 *Word Vector*），神经网络经过训练，为了能够得到更低的 *Loss* ，同类（后续词汇相同）的词汇的 *Vector* 必然在高维空间中接近
        - 存在变种：
            - *Continuous bag of word(CBOW)* ：左右预测中间。
            - *Skip-gram* ：中间预测左右。
        - （确立了 *Unsupervised Learning* 范式）

Homework 4

1.任务描述：*Speaker Classification*

2.作业实现：*Speaker Classification*

3.具体细节：

- （*Conformer = Convolution + Transformer*）：
    - *Transformer* 缺少局部敏感度，无法捕捉细微的、局部的频率变化。
    - *Conformer* 将 *Selt-Attention Block* 后的 *NN* 拆分，并在注意力后加入卷积层。
        - ⟹ *half Feed Forward NN* 半步前馈网络
        - ⟹ *Multi-Head Self-Attention*
        - ⟹ *Convolution Module*

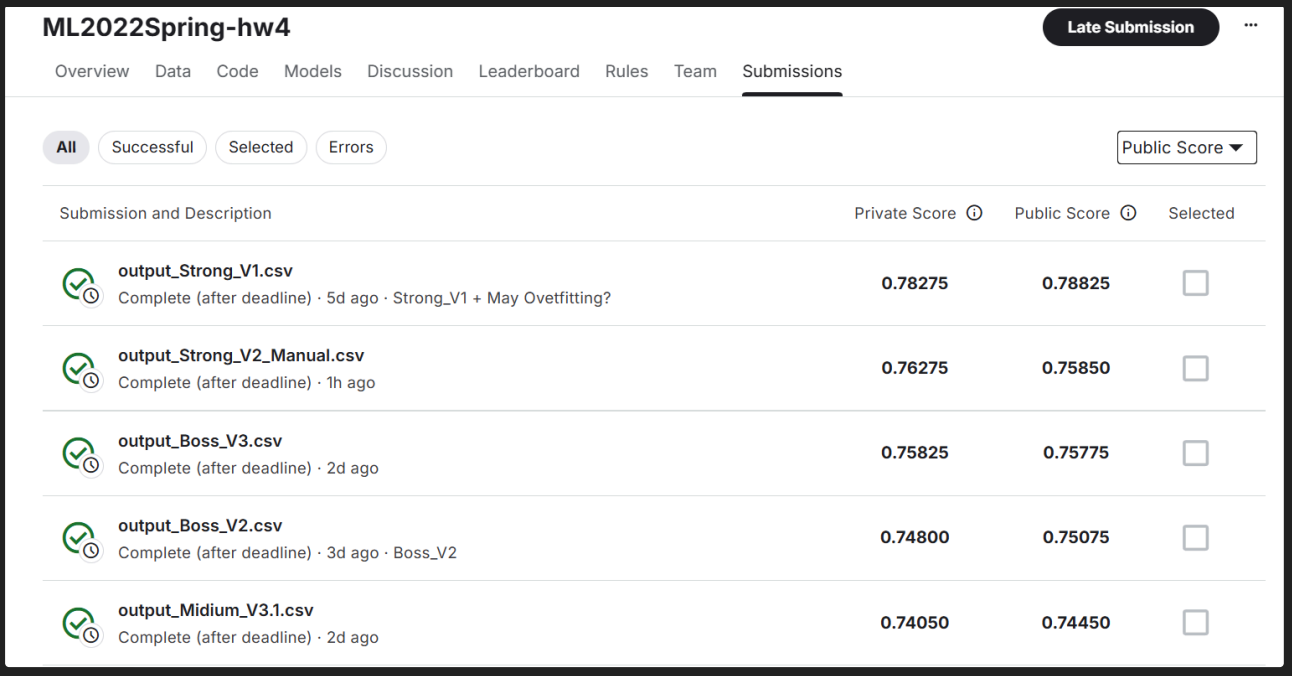
        - ⟹ *half Feed Forward NN*
        - ⟹ *layer Norm*
- *Self-Attention Pooling(SAP)*：
    - 解决*Mean Pooling*、*Statistics Pooling*“一视同仁”的缺陷。在数据(帧)中，不是所有的帧都同样重要。有点帧是静音、噪声或有效的共振峰。*Mean Pooling* 强行将噪音和有效信息平均化，稀释了特征。
    - *SAP*：让模型自行学习 *Attention Weight* ，模型将包含真实话语的帧打高分，而噪声打低分。
- *Additive Margin Softmax*：让模型从“能分清是谁”进化到“分得极其清楚，绝不认错”。
    - 传统的 *Softmax + CrossEntropy Loss* 只是为了让样本被正确分类。在特征空间里，类与类之间的边界非常模糊。只要稍微“跨过”边界一点点，模型就认为分类正确了。这在训练集上没问题，但在测试集（开放场景）上，稍微一点噪音就会让样本跳到错误的一边。
    - *AM-Softmax* 强制要求模型在特征空间里留出安全距离(*Margin*)，要求 $P(\text{正确}) > P(\text{错误}) + Margin$

4.任务指标与完成情况：

- **Sample Code**

  - Link
  - Baseline Methods
    - Simple: Run sample code & know how to use Transformer.
    - Medium: Know how to adjust parameters of Transformer.
    - Strong: Construct Conformer, which is a variety of Transformer.
    - Boss: Implement Self-Attention Pooling & Additive Margin Softmax to further boost the performance.

Class	*Score*	*Hint*
Simple	*0.60824*	*-*
Medium	*0.70375*	*Modify the parameters*
Strong	*0.77750*	*Conformer*
Boss	*0.86500*	*Self-Attention Pooling、Additive Margin Softmax*

- 

Reference

- *AM-Softmax*
- *Conformer*
- *Attention Is All You Need*
- *Self-Attention-Pooling*

5 *Transformer*

1.