CIKM 2018解决方案

参赛者：杨凯州
ID：Gambler Cops
所属组织：成功大学

目录

一、队伍成员介绍

1.成员情况介绍

我是个人参赛者，队伍成员只有我一人。

我目前在成功大学研读硕士班一年级，在机器学习的领域算是新手入门，预计将自然语言理解与强化学习作为研究方向。

2.参赛原因及收获：

我参赛最主要的原因是想试试自己的水平，想找个目标来确定自己究竟已经深入到了什么程度。很幸运地今年cikm的题目是短语匹配，与目前的研究方向一致，此外，我以前也写过聊天机器人，不过那时只用了简单的特征工程与IR算法，没有引入学习功能一直是这个专案的遗憾。所以我决定趁此机会，来重新审视短语匹配与深度学习的应用。

我这场比赛的收获有三：

1. 重新审视了句编码与注意力机制，认识到了一些花式技法，像是R-Net或者Decomposable attention，用这种方式去比对文本是以前没有想过的。
2. 试验了许多迁移学习的方法，比如说ELMO、vecmap，或者是设定不同的sample rate，同时训练英文语料与西班牙文语料等等。
3. 对于模型融合有了更深刻的认识。我尝试进行模型间知识的迁移，更详细的说明将于四、集成策略中论述。

二、特征工程

我所使用的特征可考虑为5大类，共计有56个：

1. 距离特征：
   距离特征主要是计算两个两个句编码间距离，距离的衡量方式可能为余弦相似度、欧式距离、 LCS、word moving distance、以及编辑距离（模糊比对）等等。而句编码的构成方式有3种：

   - 词袋模型：采用词袋对一个句子进行编码
   - 采用TF/IDF向量进行编码，针对词频对距离做微调。
   - 词向量& TF/IDF：以TF/IDF对一个句子中的词向量进行加权平均，作为最终的句编码

2. 主题特征：
   主题特征是从聚类的角度出发，我采用了LSI与LDA这两种降维方式，将每个词袋压缩成一低维的主题向量，并以余弦相似度等方式衡量两个句子的主题相关性。
   主题特征考虑的是对共现词的聚类，这对预测有很大的帮助，因可观察到训练集中对于同类的意图，往往会使用到类似的用词与前缀，这点在后续的模型开发中会再次提及，故不于此详述。

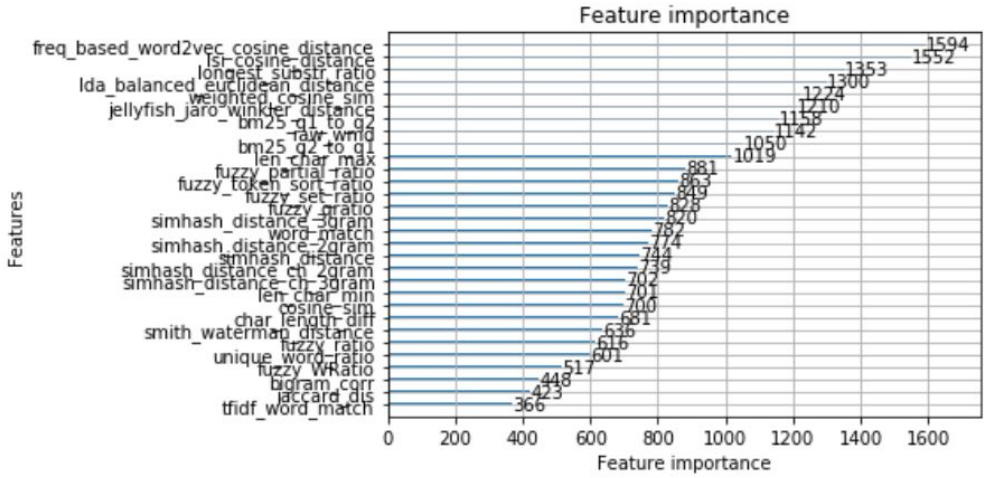
3. 共现特征：
   共现特征是计算两个句子中的词共现数，我分别考虑了1, 2, 3 gram在字元层级跟词层级上的共现统计，并针对去除停用词的情形做了额外的比对。直观上，两个句子的共现字越多，就应该越相似，但我发现资料集中存在一些特殊案例，比如说：

   - 请问争议是什么意思
   - 请问帐户冻结是什么意思

   这两句话的共现字在1,2,3 gram上都很多，但都不为关注重点，所以这类特征不适合单独使用，而该考虑为一种辅助特征。我使用特征的方式是将特征与句编码比较后的结果串接起来，故该特征可作为一信号，若其预测值与句编码的预测值呈现高落差时，就该更注意于句子相异处的比对。

4. 文本特征：
   文本特征考虑的是最原始的句子特性，比如说一个句子的长度是多少，用字的多样性、两个句子于词或字元上长度差，或去除了停用词后句子所剩的词个数等等。这些特征都具备合理的物理意义。比方说短句较可能相似，句子越长越不容易相似，这是因用字的多样性随句长增加所致。而两个句子的长度差异越多，直觉上便越不相似。去除停用词后可考虑为只关注重点的长度比较。

这四种特征具备相当强的可解释性，并很适合用在文本相似的匹配。若以这些特征训练Gradient boosting Tree，即可达约0.4的log loss，足以与Text CNN Encoder、Text RNN Encoder等经典模型并驾齐驱，并可于第一阶段挤进前百的行列。



图：LightGBM的特征重要性评估

对于无监督学习类的特征如lda,tf/idf等，我都是以全部语料（即包含了无标签语料）进行构建，且移除了重复出现中的句子，以降低来自翻译语料的噪声。除此之外，我的特征工程尽量以对称性为大原则，这是一种正规化语言模型的方法，具体细节会于五.3对称性提到。

5.数据处理

我进行了最小限度的前处理，因为发现到Stemming或去除停用词皆会使得模型精度降低，推测理由会是遗失了指称关系。比如说若我们将「你」、「我」这类停用词去除，则对于句子「你发生了汇款问题」与「我发生了汇款问题」就会变得不可分，注意这个情形是会发生的，语料中有参杂类似客服或系统的回应，此时知道这句话的主词就显得很重要。

在符号的处理上，我将逗号、句号、惊叹号等透过正则表达式与紧接着的词分开，这有助于减少oov words，因为像"hello," "dispute."等可能没有训练好的词向量(就算有，该词向量也不严谨，因为这种term出现的次数必定远少于正常的表示法)，但"hello" ","与"dispute" "."则可正确抓取到。值得一提的是，西班牙语中有一个倒过来的问号¿当作问句的开头，我并没有将¿给替换为？，因为我发现相比于?，¿更能够代表一个西班牙语的疑问语气，换句话说，可以将¿当成一类分类指标，来区分目前是陈述句或是疑问句。
此外，我发现到训练集有重复的训练句子配对，为避免过度采样这些案例，我会取出每个唯一的句子配对作为训练集。

三、模型设计

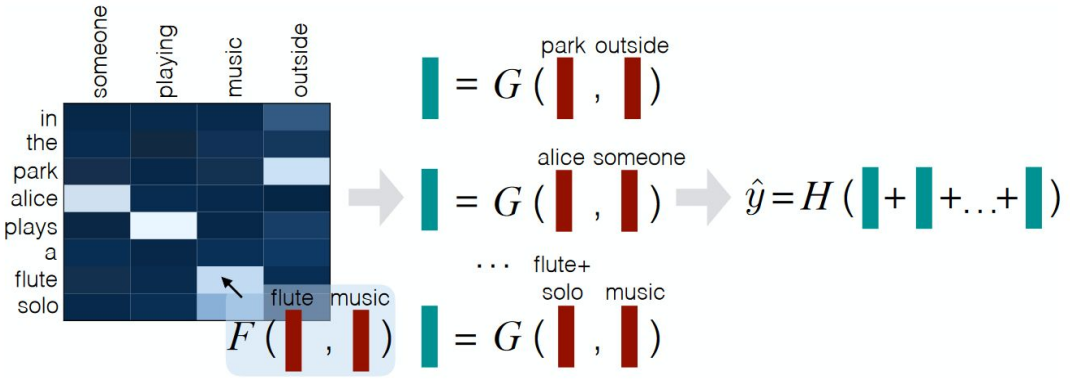
我所使用的主力模型共有三种：

- Decomposable Attention
  - Densely augmented CNN
- Cafe like RNN

这三种模型各有不同的应用情境，并于模型集成时达到了相当好的平衡，该方面的说明将于第五小节呈现。

1. Decomposable Attention：构建词的类比关系

对于一组序列A、B而言，Decomposable Attention所考虑的是如何以B的字表示A的字，或是以A的字来表示B的字。



图： Decomposable Attention用途示意

其物理意义就是将相似字对齐在一起，这在评估该资料集的相似能有极大贡献，因为训练集常有如下的句子：

- Q1: 我想询问订单
- Q2: 我想询问争议 (disputa)
- Q3: 我想联络批发商
- Q4:帮我联络卖家

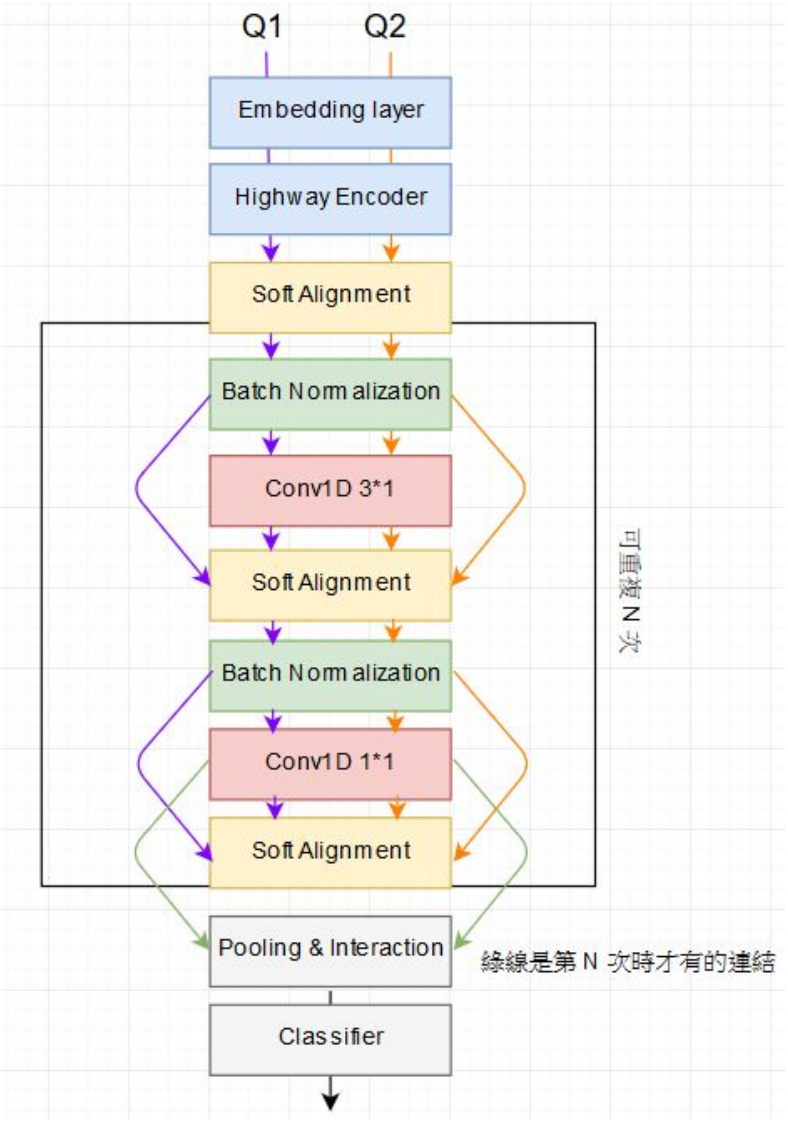
这类存在特定规则 (我想知道……、什么是……、我想请你. ……)且重复出现的句子，我们并不会希望Q1跟Q2因为都是想问什么而被归类为相似，而是因为他们想问的东西不同而被归类为不相似。

Decomposable Attention的优势在于能捕捉到那些规则，无论它们出现在句子的何处(Attention是无序的)，捕捉到的概念会由Compare layer学习它们的差异性，比方我想询问和我想询问是相同的，订单和争议是不同的，而批发商与卖家是相同的，在下一个batch的学习中，Attention就能将批发商与卖家对齐，

进一步去学其他单词语意上的不同。**可以说Decomposable Attention在这个task上，供我们能专注在句子上单词的类比关系。**

2.Densely augmented CNN：构建片语的匹配

因为Attention无序的优势也会是其劣势，这会使其对片语的关注力不足，这对于修饰字的比对效果不佳，比如说'My'，我们要关注的不应该只有'My'这个单词，而是要关注'My'后面所衔接的被修饰字，比如说'My order', 'My dispute'，换句话说，我们要关注的不会是'My'这个字，而是与在'My'周边的词组。考虑到要中心词引入周边资讯，我选择引入卷积层，并设计了一类似DenseNet的CNN架构DACNN，如下所示：



图：DACNN架构

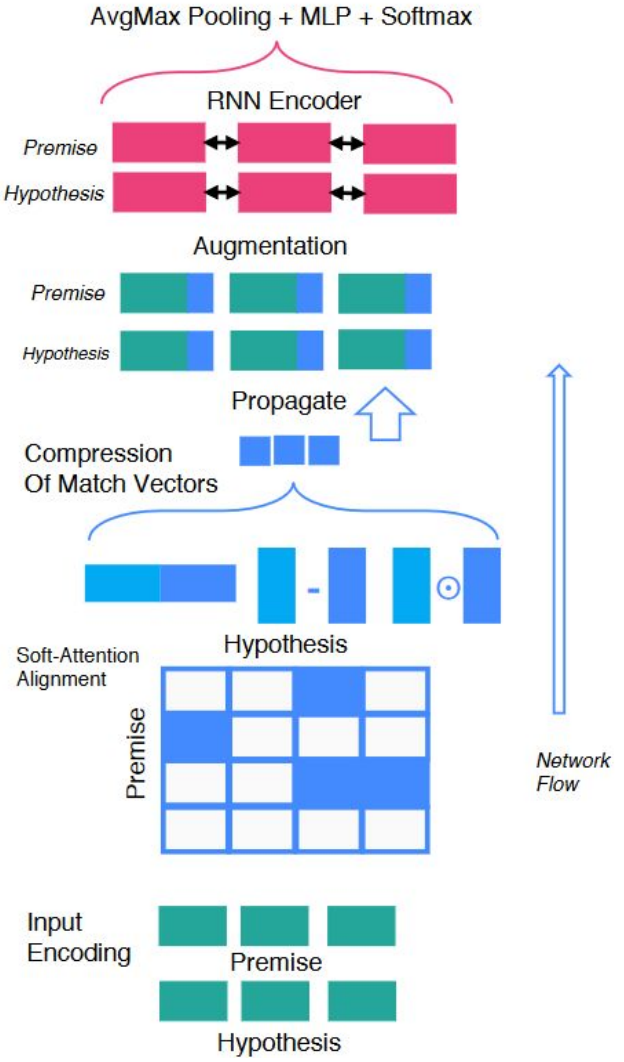
该模型可考虑为Decomposable Attention的扩充版本，我将每次卷积后Soft Alignment的结果串接回原输入，达到扩充原输入资讯的效果，这与DenseNet的设计思维相当类似。该架构是相当迅速且有效的，相比于接下来要介绍的

RNN模型， DACNN所需的训练参数只有其六分之一、每个Epoch的耗时为其三分之一，且两者能达到相去不远的log loss。

透过反覆地soft align两个句子，片语能对应到另一个句子的视野域也将增大，比对也更加趋于抽象的意图。此外，这种堆叠形式的soft align也能考虑为跳跃性的Self attention（每叠两次就会attend到自身），在 R-Net 中即指出，该方法(self-matching)对于总结句子本身的语意也相当有帮助。后续将提到的Cafe也是启发自该思维。

3. Cafe like RNN：捕捉序列语意

前两者是专注在对单词及片语的匹配，第三个模型则是著重以RNN进行序列语意的补中，Cafe的架构为：

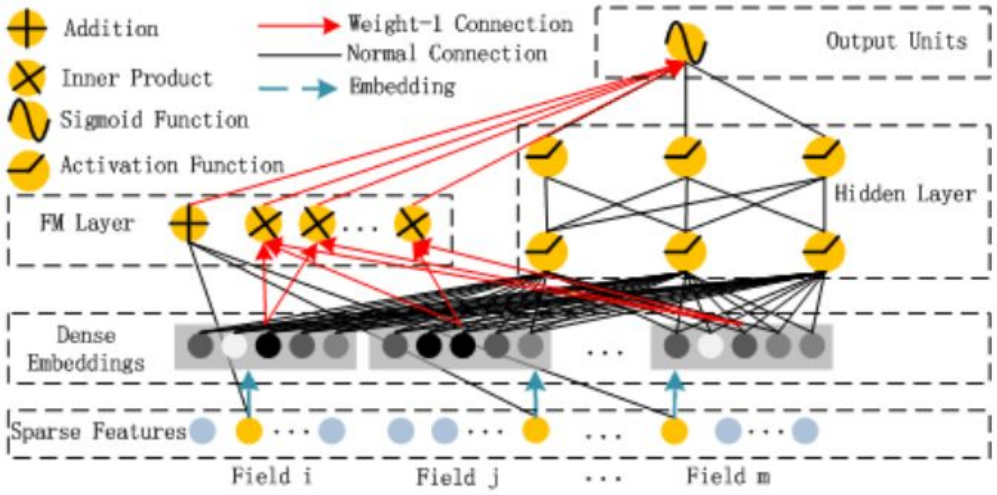


图：Cafe模型架构

Cafe的前半部同样用到Decomposble Attention的思维，其特点为后半部，将互动后的特征扩充回原本的词向量，达成传递互动性的效果(可以考虑为ESIM 的变形)。

我的Cafe与经典的Cafe模型不同点有三处，达成此资料集上更好的效能：

1. 我没使用Factorization Machine进行局部以及全局的比对，取而代之的，我使用的是多层感知机搭配ReLU。这个架构是我在尝试将FM part替换为 Deep FM Part时所发现，其优势在于感知机的输出可以为一个向量，而不像FM只为一个纯量，使感知机容易引入Dropout做正规化，此外，多层感知机于第二隐藏层开始也可建立起输入间的高阶互动关系。



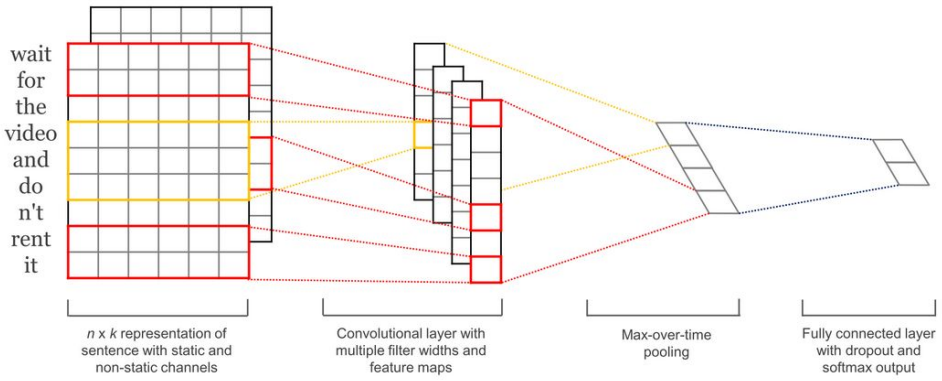
图：Deep Factorization Machine架构
左为FM Part，右为Deep NN Part

2. 假设输入为q1, q2，Cafe的互动层设计为[q1; q2], |q1 - q2|, q1 * q2，我在这之上增加了q1 + q2，相较于原架构，引入相加比对降低了约0.015的logloss，是极为惊人的提升。

3. 我将Cafe调整成了完全对称的结构，及限制Similarity(A, B) = Similarity(B, A)，这样更符合人类的逻辑，即A句与B句相似，那我们会预期B句与A句也相似。关于对称性的设计，我会于五.3对称性有更多描述。

4. Other Nets

除了上述模型，我另外实验了两个模型，但因其开发较晚，所以不为主力模型，但我有将其作为模型融合的素材。第一个模型为传统的Text CNN



图：Text CNN架构

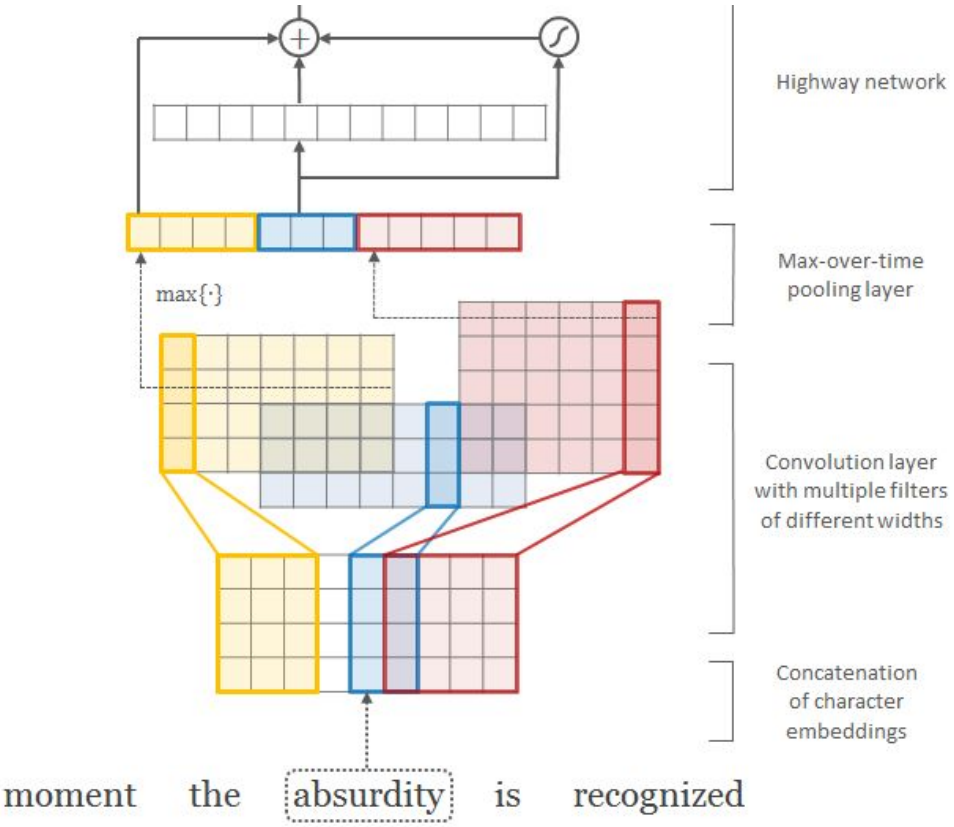
第二个是4 Way Pooling Cafe，原Cafe的Pooling方式只有Max Pooling与Average Pooling，我追加了Last Pooling (取RNN的最后一个输出)，与Attention Pooling，

Attention Pooling是使用一个带激活Linear transform，评估序列中每个向量的重要性权重，最后的Context vector为序列中所有向量的加权平均。

5. Diversity

为了引入模型的多样性，我采用了3种不同的方式训练我的模型：

1. Word Level：以词为最小单位进行模型的训练，使用官方提供的预训练词向量，并固定其权重。
2. Char Level：以字元为最小单位进行模型的训练。词向量会由该词的字向量构建而成，建模方式参考自 Character-Aware Neural Language Models，以若干个不同大小的卷积元串接MaxPooling得出。



图：Char Embedding至Word Embedding的组成方式

3. Meta Word Level：与1相同，以词为最小单位进行模型的训练。不同点在于我在特征连接层中引入了第二章提到的56个特征，让模型学习传统特征与词嵌入特征间之互动关系。

这3种训练方式皆可产出不错的模型，且彼此间的相关性低，相当适合用于后续的模型聚合。

模型	Stage 1 LB
DenseCNN	~ 0.356
DenseCharCNN	~0.40
DenseMetaCNN	~0.37
Bagging 3 Above	~0.343

表：DenseCNN的LB分数

	DenseCNN	DenseCharCNN	DenseMetaCNN
DenseCNN	1	0.89	0.95
DenseCharCNN	0.89	1	0.89
DenseMetaCNN	0.95	0.89	1

表：三种DenseCNN预测值的相关系数

模型	Stage 1 LB
SymCafe	~ 0.353
CharSymCafe	~0.40
MetaSymCafe	~0.37
Bagging 3 Above	~0.34

表：三种SymCafe的LB分数

	SymCafe	CharSymCafe	MetaSymCafe
SymCafe	1	0.87	0.96
CharSymCafe	0.87	1	0.87
MetaSymCafe	0.96	0.87	1

表：三种SymCafe预测值的相关系数

此外，如前几节所述，我所设计的模型分别对应至资料集中不同的应用情境，其预测在达成接近的Log loss下仍能维持约0.95的相关系数，亦有助于模型融合。

	SymCafe	DenseCNN	Decomposable Attn
SymCafe	1	0.93	0.9
DenseCNN	0.93	1	0.92
Decomposable Attn	0.9	0.92	1

表：三种不同模型预测值的相关系数

总结而言，在同种模型间我以训练输入的不同引入多样性，不同种模型也因假说不同而存在多样性，使所有模型对后续的融合皆能有所贡献。

四、集成策略

我模型融合是两个阶段的Stacking，第一阶段为加权平均，第二阶段则是使用DART (Dropout additive regression Trees)与Pseudo labels进行预测微调。

1. Weighted bagging

第一阶的集成是基于加权平均，权重来自第一阶段各模型在Leaderboard的表现以及分析预测值的相关系数而得：

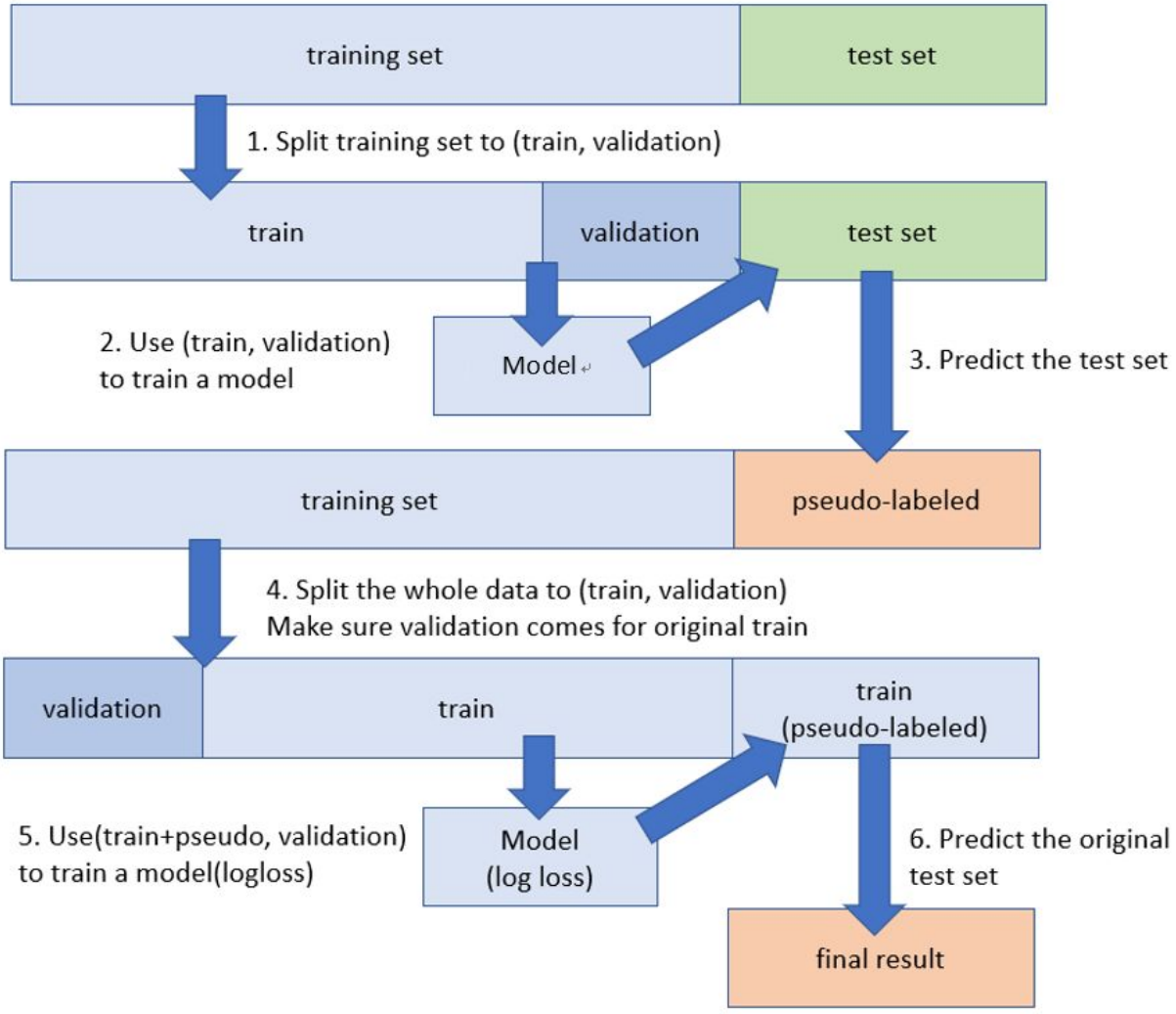
In [191]:

```
tests['ann_blending'] = (tests['oofs_pred2'] * 0.2 + tests['oofs_pred9'] * 0.4 + tests['oofs_pred1'] * 0.4)
tests['cnn_blending'] = (tests['oofs_pred5'] * 0.54 + tests['oofs_pred7'] * 0.32 + tests['oofs_pred6'] * 0.14)
tests['rnn_blending'] = (tests['oofs_pred10'] * 0.5 + tests['oofs_pred12'] * 0.33 + tests['oofs_pred11'] * 0.17)
tests['4way_blending'] = (tests['oofs_pred0'] * 0.6 + tests['oofs_pred8'] * 0.4)
tests['av_blending'] = (tests['oofs_pred3'] * 0.5 + tests['oofs_pred4'] * 0.5)

tests['all_blending'] = (tests['ann_blending'] * 0.15 + tests['av_blending'] * 0.1 + tests['4way_blending'] * 0.25 +
                         tests['cnn_blending'] * 0.25 + tests['rnn_blending'] * 0.25)

trains['ann_blending'] = (trains['oofs_2'] * 0.2 + trains['oofs_9'] * 0.4 + trains['oofs_1'] * 0.4)
trains['cnn_blending'] = (trains['oofs_5'] * 0.56 + trains['oofs_7'] * 0.36 + trains['oofs_6'] * 0.18)
trains['rnn_blending'] = (trains['oofs_10'] * 0.5 + trains['oofs_12'] * 0.33 + trains['oofs_11'] * 0.17)
trains['4way_blending'] = (trains['oofs_0'] * 0.6 + trains['oofs_8'] * 0.4)
trains['av_blending'] = (trains['oofs_3'] * 0.5 + trains['oofs_4'] * 0.5)

trains['all_blending'] = (trains['ann_blending'] * 0.15 + trains['av_blending'] * 0.1 + trains['4way_blending'] * 0.25 +
                         trains['cnn_blending'] * 0.25 + trains['rnn_blending'] * 0.25)
```

图：第一阶集成的模型权重

第二阶段的集成混合了Knowledge distillation与Pseudo label的思维。

我将第一阶段的预测值当作真实标签，并把测试数据集与原始训练集串接起来，继续训练第一阶段用于集成的13个模型(code: FineTuning.ipynb)，每个folds的Early stopping仍是参照同样的验证集，具体流程可考虑如下图：



图：Level 2 Stacking

我选择采用软标签而非硬标签有两个理由：

1. 原始资料集本身即具备噪声，这点可由句子的递移性得到验证，可分为两种情形：

    - 情形一：若A与B相似，且B与C相似，则A与C当相似
    - 情形二：若A与B相似，且B与C不相似，则A与C不当相似

    经由训练集的评估可知，情形一的正确率约75%，情形二的正确率约95%，可推断出原始训练集的标签并不具有一致性，以0或1来当作训练目标并不严谨。

2. 使用软标签(soft label)本身即是非常有效的正规化手段，这点于 Learning with Noisy Labels、Regularizing Neural Networks by Penalizing Confident Output Distributions 等论文中有深刻的讨论，特别是使用集成各模型后预测的软标签，这个想法启发自 Distilling the Knowledge in a Neural Network，但我使用的不是out of fold predictions的软标签，而是选择添加测试集的预测当成额外的训练资料。

引入pseudo label的理由是因我发现测试集与训练集的分布有明显的不同，如主题及用词的种类不一致，且有若干词是只出现于测试集而不出现于训练集等等，虽然预训练的词向量能某种程度上缓解冷启动的影响，但仍不足以建立句子间词的推理或类比关系，因该层关系往往是由Highway Encoder (SymCafe)或Compare Layers (Decomposable Attention)在训练中学习而出。

在引入来自first level ensemble预测的pseudo label后，单模型的效果也有显著的提升：

Model	SysCafe
Logloss before fine-tuning	0.353
Logloss after fine-tuning	**0.348**
Improvement	0.005

如 Distilling the Knowledge in a Neural Network 所述，在该task上大模型融合后的知识(来自stage1的weighted bagging)仍旧有办法转移至小模型，因此在实务上，我们能先训练一个高精度但较为复杂的大模型，再将其知识迁移至运算快的小模型(如Decomposable attention或DACNN)上，以求兼顾准确度与运算效能。

除此之外，由于训练集大多是由英文经过Google翻译后的语料，翻译造成的资讯损失也会成为一个效能瓶颈。而使用pseudo label在原生的西班牙文测试集上训练，则可缓解冷启动以及避开翻译错误等问题。

在结合两者后，最终的预测误差有了显著的下降：

	Stage 2 Online Logloss	Local Logloss
Best Single Model	~0.40	~0.24
Level1 Stacking	0.38440	~0.20
Level2 Stacking	0.37645	~0.19
L1 to L2 Improvment	**0.00796**	~0.01

表：集成效能评估

五、正规化

我在训练与模型的设计上引入了三种正规化(regularization)的手法：

1. 软标签

   如前所述，引入软标签可有效使模型不要过分肯定自己的预测，尤其评估方式为logloss时，分布属于一个平滑而非极端的预测会有较好的成果。

2. Dropout

我主要于模型的两处引入Dropout：

- Embedding Dropout：

  属于输入层级的Dropout，可以考虑为一类合理的资料增强，因为一个类别并不会在随机删除几个字后就变成另一个类别。根据Dropout rate不同，Embedding Dropout能对原始资料做了大量的扩充，故而使较复杂的模型(如Cafe，训练参数近2M)也能在小量资料集上有不错的表现。

Model	local logloss
TextCNN without ED	0.40
TextCNN with ED	0.31
Cafe without ED	0.27
Cafe with ED	0.24
Decomposable attention without ED	0.29
Decomposable attention with ED	0.25

表：Embedding Dropout对效能的影响

- Concatenate Dropout

  我发现与其在每个全连接层后衔接小量的Dropout，不如在特征连阶层加入大量的Dropout，这可以类比为随机森林的运作方式，有时候的预测只基于两个句向量的内积，而有的时候的预测只偏重在两个句向量间的差值，这种方式能使模型有效地聚合各种特征。

3.对称性

我的模型和特征是讲求对称的，即是说[q1, q2]与[q2, q1]的特征应当尽可能一致，比方说考虑句子长度这个特征，与其用length_q1与length_q2这两个特征，我更倾向采用max_length跟min_length，因为该表示不会因输入的顺序调换而改变。

关于模型对称性与正规化的研究，于 Extrapolation in NLP 有深入的探讨，以其中对SNLI的评估为例：

Instances	DAM	S-DAM
Whole Test Set	86.71%	85.95%
Contradictions	85.94%	85.69%
Reversed Contradictions	78.13%	85.20%

表：对称性对准确度的影响
取自Extrapolation in NLP，其中DAM为非对称模型，S-DAM为对称模型

可发现DAM能预测A,B为矛盾，却无法推理出B, A也属于矛盾，使得准确度从85.94%骤降至78.13%，而S-DAM无论输入的顺序为何，始终保持一致的效能，这对于unseen data来说会是相当好的正规化。

我认为这点同样能类比至本议题上，因为相似度的度量理当是对称的，即是说会要求Sim(A, B) = Sim(B, A)，然而现有的多数用于文本相似匹配的模型都不符合这项要件，

因它们往往如此定义最后的特征互动层(Interaction layer)为：

$$out = W([q1; q2] , |q1 - q2|, q1 * q2)$$

其中[q1; q2]表示两向量的串接，因此会因句子的输入顺序不同，而有不同的输出。然而这个特征往往是相当重要的，不适合从特征互动层中移除，可见 CAFE 中的 ablation study：

	Match	Mismatch
Original Model	79.0	78.9
(1) Replace FM with FC	77.7	77.9
(2) Remove Char Embed	78.1	78.3
(3) Remove Syn Embed	78.3	78.4
(4) Remove Inter Att	75.2	75.6
(5) Replace Highway Pred. with FC	77.7	77.9
(6) Remove Highway Enc. with FC	78.7	78.7
(7) Remove Sub Feat	77.9	78.3
(8) Remove Mul Feat	78.7	78.6
(9) Remove Concat Feat	77.9	77.6
(10) Replace LSTM with BiLSTM	78.3	78.4

表：ablation study on CAFE 。可发现Remove Concat Feature后Match/Mismatch分别从79.0 / 78.9骤降至77.9 / 77.6

因此我将互动层调整为：

$$out_1 = W([q1; q2] , |q1 - q2|, q1 * q2)$$
$$out_2 = W([q2; q1] , |q1 - q2|, q1 * q2)$$
$$out = (out_1 + out_2) / 2$$

这种方式对模型精度不会造成太大影响(本地验证集约改变 0.003)，也能兼顾模型的对称性。

六、结论

于本次竞赛，我基于共现词、文章主题以及资讯撷取的技术建立了四类传统的NLP特征，并以词单位、字元单位以及混合词与传统特征三种方式训练模型，达成单模型间有效的变异性。

在模型的使用上，我使用了经典的Decomposable Attention进行单词的对齐与比对，并提出了一快速且有效的卷积结构DACNN进行片语匹配。在序列信息的建构上，我调适了CAFE模型的比对层与互动层架构，并建立了相似度上的对称关系，达到有效的正规化。

我们使用了两阶段的模型融合，第一阶段是基于模型表现所做的加权平均，第二阶段则引入了Knowledge distillation的思维，以Pseudo Labeling搭配Soft target对原模型进行微调，最后使用Dropout Additive Regression Tree进行模型的二阶段融合。我的融合架构最终降低了近0.025的 Log loss，达成了 6%的改善，是为相当显著的提升。

七、参考资料

1. A Compare-Propagate Architecture with Alignment Factorizationfor Natural Language Inference
2. A Decomposable Attention Model for Natural Language Inference
3. Character-Aware Neural Language Models
4. Convolutional Neural Networks for Sentence Classification
5. DART: Dropouts meet Multiple Additive Regression Trees
6. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction
7. Densely Connected Convolutional Networks
8. Distilling the Knowledge in a Neural Network
9. Enhanced LSTM for Natural Language Inference
10. Extrapolation in NLP
11. From Word Embeddings To Document Distances
12. Learning with Noisy Labels
13. Regularizing and Optimizing LSTM Language Models
14. Regularizing Neural Networks by Penalizing Confident Output Distributions
15. R -net: Machine reading comprehension with self matching networks
16. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm