小哥哥，检索式chatbot了解一下?

原创 夕小瑶 夕小瑶的卖萌屋 2018-09-25

来自专辑
卖萌屋@自然语言处理

喵喵喵，一不小心又匿了三个月，突然诈尸害不害怕(￣▽￣)

小夕从7月份开始收到第一场面试邀请，到9月初基本结束了校招（面够了面够了T_T），深深的意识到今年的对话系统/chatbot方向是真的超级火呀。从微软主打情感计算的小冰，到百度主打智能家庭（与车联网？）的DuerOS和UNIT，到渗透在阿里许多产品的全能型智能客服小蜜，以及腾讯的小微和搜狗的汪仔，更不必说那些大佬坐镇的独角兽公司了，小夕深感以对话为主战场的NLP之风在工业界愈演愈烈，吓得小夕赶紧码了这篇文章。

> 注：本文同步到小夕的知乎专栏，可能存在的校正与更新见https://zhuanlan.zhihu.com/p/44539292

1. 扫盲

对话的概念很大，从输入形式上分为文本和语音，本文当然只考虑文本。从对话目的上分为任务型对话与非任务型/闲聊型对话。顾名思义，任务型对话就是为了解决任务而进行的对话，比如你让Siri帮你定闹钟、发短信等，而闲聊型对话当然就是human-to-human的正常聊天啦。本文就不讨论任务型对话了，有兴趣的同学可以戳这里扫扫盲，本文聚焦在非任务型对话的多轮对话问题上。

要完成对话的建模，目前主要分为检索式、生成式以及检索与生成融合的方式。顾名思义，检索式就是通过检索与匹配的方式从已有的大量candidate responses中找出最合适的那个作为response；生成式则是事先通过训练来把对话知识塞进模型中，推理的时候首先模型的encoder部分去读历史对话，然后模型中的decoder/语言模型部分直接生成相应的回复；检索与生成相结合的方法则玩法很多了，比如用生成模型来做检索模型的reranker，用生成模型来作改写，用生成模型生成的response来作为检索模型的一条response等。限于篇幅，本文只讲纯检索式的，其他的以后再说（maybe不会太久┐(￣▽￣"")┌）。

2. 检索式模型的套路

检索式对话的一般套路是首先构建一个由大量query-response pair构成的知识库（比如从豆瓣、贴吧等地方抽取），然后将对话中最后一次的回复作为query，通过经典的信息检索方式（倒排索引+TFIDF/BM25）作q-q匹配来召回若干相关的candidate responses。注意，这一步实在太粗糙了，完全没有考虑语义，所以直接使用检索分数来挑选最优response显然是太过简单粗暴不靠谱。所以我们还需要使用考虑语义的深度文本匹配模型来将历史对话与这些检索出来的candidate responses进行matching/reranking，从而挑选出一个更加合适的response。

那么怎么进行文本的深度匹配呢?

一个很简单的做法是直接把复述识别/自然语言推理/检索式问答这些相关领域的文本匹配模型直接拿来用，但是显然这样仅仅建模的是单轮对话，于是聊天机器人就变成了只有7秒记忆的金鱼┐(ﾟ▽ﾟ)┌，因此，建模多轮对话是非常有必要的。

不过了解一下文本匹配模型是很有帮助的。这方面今年COLING有一篇文章[6]总结的不错，把基于表示与基于交互的SOTA匹配模型都给详细总结对比了。

基础比较差的同学可以看这篇文章，从2013年的DSSM[9]开始入手，慢慢补。篇幅所限，加上这方面研究相对很充分了，小夕就不展开讲啦。所以话说回来，将多轮对话与候选回复进行匹配的正确方式是什么呢?

3. 论文串烧

一切还要从两年前的秋天说起，曾经，有一个少年。。。

算了算了，还是正经点吧，要不然没法写了┐(￣▽￣"")┌总之，小夕从众多鱼龙混杂的检索式多轮对话的论文里精选出如下4篇进行串烧（按时间顺序，从经典到state-of-art），包括：

- EMNLP2016 百度自然语言处理部的xiangyang大佬的**Multi-view**[1]
- ACL2017 MSRA吴侃大佬的**SMN**[2]
- COLING2018 上交的**DUA**[3]
- ACL2018 百度自然语言处理部xiangyang大佬和lilu女神的**DAM**[4]

不过不要怕，小夕的论文分享总是浅显易懂还带点萌(￣▽￣)

必须要提的经典：Multi-view model

想一下，怎么才能从单轮q-r的匹配扩展到多轮呢？一个最最最简单的想法就是直接把多轮对话首尾连接变成一个长长的单轮┐(￣▽￣"")┌比如这种：

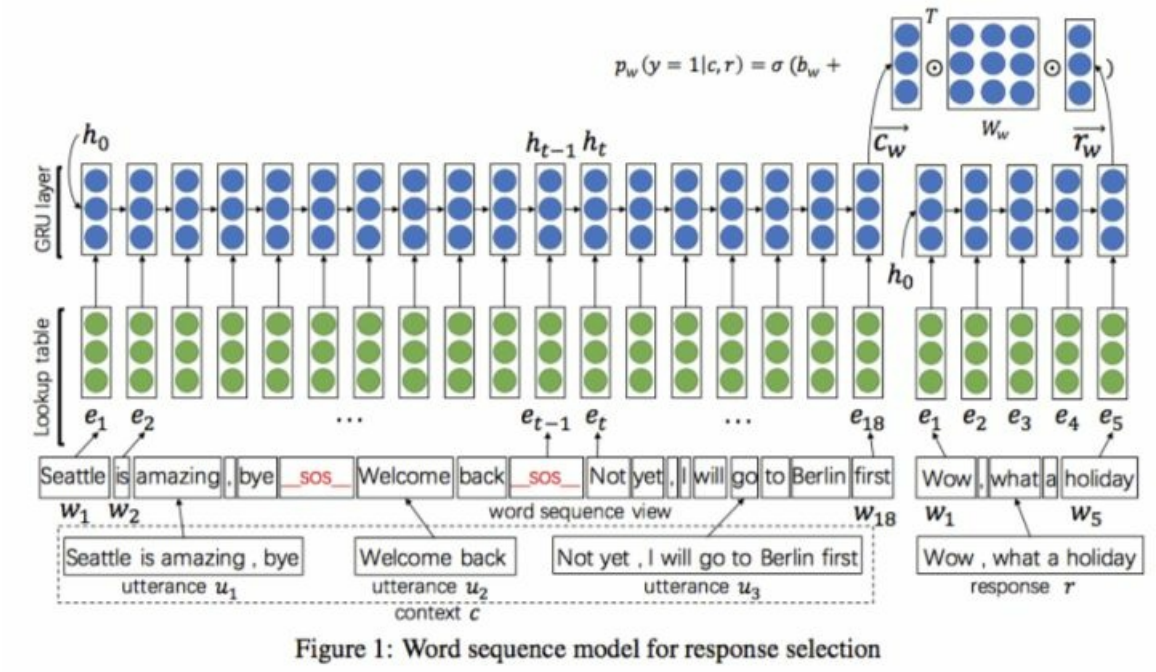


Figure 1: Word sequence model for response selection

如上图，首先将各轮的对话连接起来（这里在连接处插入一个"__SOS__"的token），然后这里用RNN系网络取最后时刻隐态的方法分别得到query和response的向量表示，进而将这俩向量通过

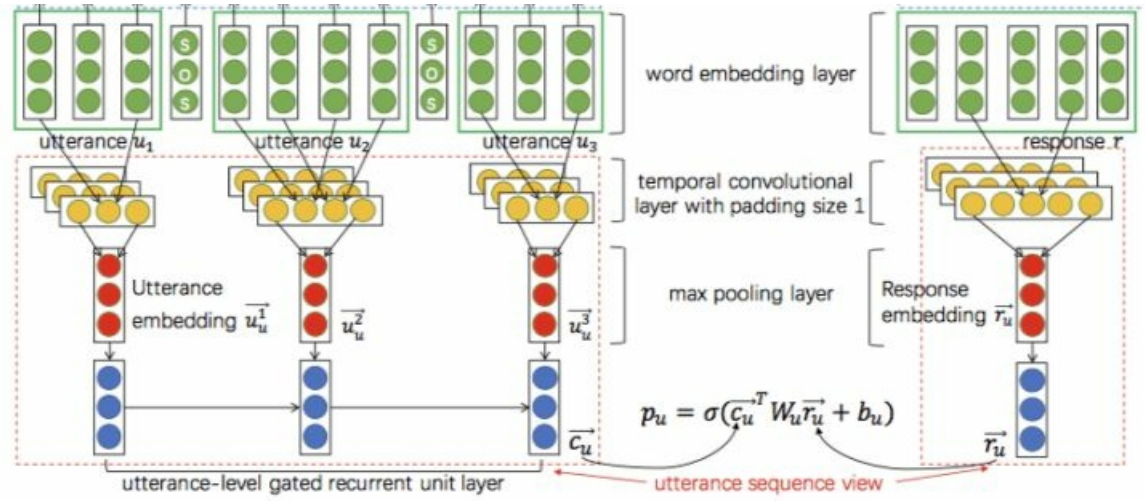
$$score(v1, v2) = v1^T \cdot M \cdot v2$$

得到匹配分值（M为网络参数），进而通过

$$p = sigmoid(s(v1, v2) + b)$$

得到匹配概率（p为参数）。当然，其实这里本质上就是一个**基于表示的文本匹配模型**，所以完全可以用更复杂的表示方法和匹配函数（如SSE模型[8]）来完成这个过程。

聪明的童鞋肯定可以想到，显然这种将长长的word embedding sequence直接塞进网络得到整个多轮对话的表示（context embedding）的做法未免太看得起神经网络对文本的表示能力了，因此作者提出，不仅要在这个word-level上进行匹配，而且还要在一个更高的level上进行匹配，这个level称为utterance-level（即**把对话中的每条文本（utterance）看作word**）。



如上图的绿色->黄色->红色的部分，首先得到对话的每条文本（utterance）的向量表示（这里用的14年Kim提出的那个经典CNN），这样历史的多轮对话就变成了一个utterance embedding sequence。之后再通过一层Gated RNN（GRU、LSTM等）把无用的utterances中的噪声滤掉，进而取最后一个时刻的隐状态得到整个多轮对话（context）的context embedding啦。

拿到context embedding后，就可以跟之前word-level中的做法一样，得到对话与candidate response的匹配概率啦。最后，将word-level得到的匹配概率与utterance-level得到的匹配概率加起来就是最终的结果。

实验结果如下

Model/Metrics	1 in 10 R@1	1 in 10 R@2	1in 10 R@5	1 in 2 R@1
Random-guess	10%	20%	50%	50%
TF-IDF	41.0%	54.5%	70.8%	65.9%
Word-seq-LSTM (Lowe et al., 2015)	60.40%	74.50%	92.60%	87.80%
Word-seq-GRU	60.85%	75.71%	93.13%	88.55%
Utter-seq-GRU	62.19%	76.56%	93.42%	88.83%
Multi-view	**66.15%**	**80.12%**	**95.09%**	**90.80%**

可以看到utterance-level确实是明显比word-level work的，而且集成一下提升效果更显著。因此从这篇论文后的大部分论文也follow了这种对每条utterance分别进行处理（表示或交互），而后对utterance embedding sequence用Gated RNN进行过滤和得到context embedding的思路。

而到了2017年，文本匹配的研究明显变得更加成（花）熟（哨），各种花式attention带来了匹配效果的大幅度提升，这也标志着检索式多轮对话这方面的玩法也将变得丰（麻）富（烦）。

一次大大的进化：SMN model

如果说Multi-view模型在检索式多轮对话领域开了个好头，那么SMN则是将这个大框架往前推进了一大步。虽然表面上看Multi-view模型与SMN模型相去甚远，但是熟悉文本匹配的小伙伴应该有注意到，16年左右，基于交互的匹配模型开始代替基于表示的匹配模型成为主流[6]，因此在Multi-view中内嵌的匹配模型是基于表示的，而到了17年的这个SMN模型则使用了前沿的基于交互的匹配方法。另外除了改变文本匹配的“派系”之外，SMN还有一个比较亮的操作是在做文本匹配的时候考虑了文本的不同粒度 (granularity) 之间的匹配，这个操作也成为了后续一些paper的follow的点。

对文本匹配比较熟悉的同学应该在AAAI2016看过这么一篇paper：

Text Matching as Image Recognition （参考文献[5]）

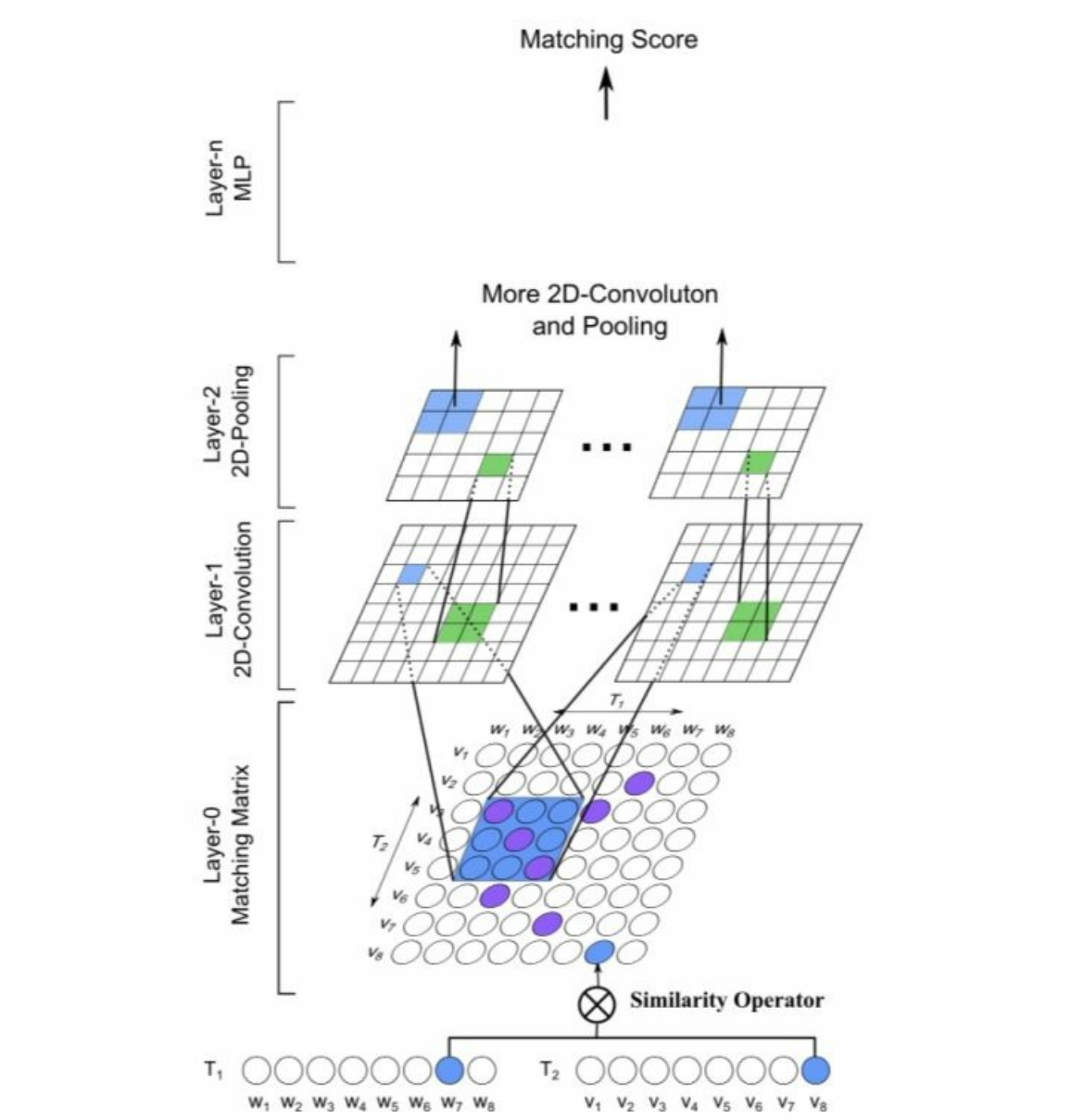


Figure 3: An overview of MatchPyramid on Text Matching.

如图，基本思想就是，使用传统的attention来计算出两个文本的word-level对齐矩阵/相似度矩阵后，将该矩阵看成一个图像，然后使用图像分类模型（如CNN）来得到更高level的相似度特征表示（比如phrase level, segment level等），进而最终得到全局的相似度匹配特征。这也是最早的几个交互式文本匹配模型之一。

SMN这篇paper就是采用了这个思想。给定一个candidate response，在生成word-level的每个utterance的向量表示的时候，首先计算出历史上每个utterance跟该response的对齐矩阵，然后对每个对齐矩阵，均使用上面这种图像分类的思想生成high-level表征文本对相似度的特征向量作为该utterance的向量表示（utterance embedding）。

之后就是使用前面Multi-view中的做法，从这个utterance embedding sequence中得到整个对话的context embedding，最后将该context embedding和之前的word-level下得到的context embedding与response的向量去计算相似度了。

不过作者这里在**计算对齐矩阵和得到context embedding**的时候，用了更复杂一些的方法。如图

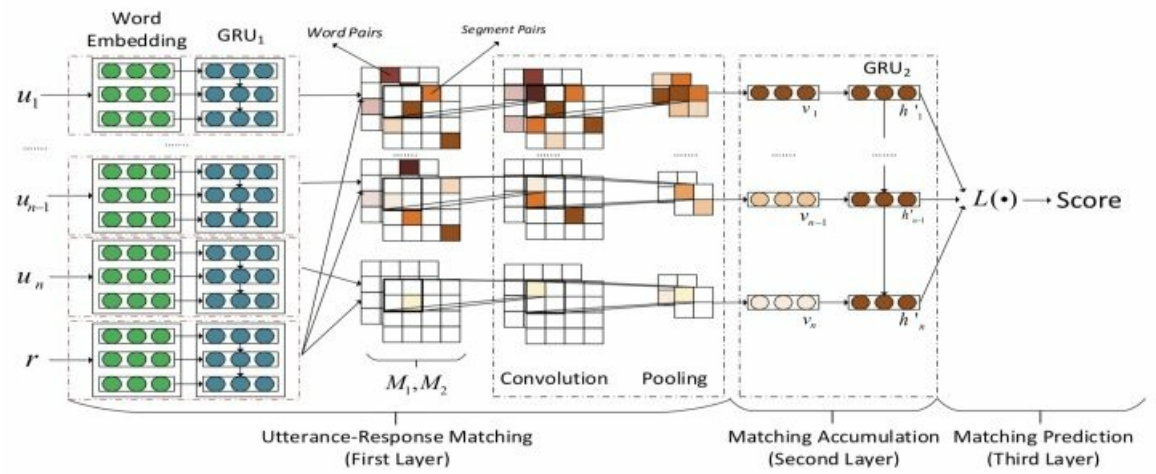


Figure 1: Architecture of SMN

在计算对齐矩阵的时候，作者不仅用了原始的word embedding，而且同时用了RNN系模型对文本encoding之后的隐状态（即编码过上下文信息的word embedding，可以看作phrase-level的"word embedding"了），这样就生成了两份对齐矩阵，然后这样将两份对齐矩阵作为两个channel丢进“图像分类模型”，从而保证了即使图像分类模型很浅，也能抽取出比较high-level的特征，得到高质量的utterance embedding。

另外，作者这里在得到最终的context embedding的时候，除了使用RNN最后一个隐状态的传统做法（记为SMN_{last}）外，作者还额外实验了对顶层各个time step的隐状态进行加权求和（权重可训练）的方式（SMN_{static}）以及更复杂的集成utterance自身表示的信息并使用self-attention的方式（$SMN_{dynamic}$），实验结果表明，总的来看$SMN_{dynamic}$的方式稍好一些（不过考虑到额外引入的计算和存储开销，一般不值得这样做）。有兴趣的同学可以去看原paper，这里就不展开讲啦。

	Ubuntu Corpus				Douban Conversation Corpus					
	$R_2@1$	$R_{10}@1$	$R_{10}@2$	$R_{10}@5$	MAP	MRR	P@1	$R_{10}@1$	$R_{10}@2$	$R_{10}@5$
TF-IDF	0.659	0.410	0.545	0.708	0.331	0.359	0.180	0.096	0.172	0.405
RNN	0.768	0.403	0.547	0.819	0.390	0.422	0.208	0.118	0.223	0.589
CNN	0.848	0.549	0.684	0.896	0.417	0.440	0.226	0.121	0.252	0.647
LSTM	0.901	0.638	0.784	0.949	0.485	0.527	0.320	0.187	0.343	0.720
BiLSTM	0.895	0.630	0.780	0.944	0.479	0.514	0.313	0.184	0.330	0.716
Multi-View	0.908	0.662	0.801	0.951	0.505	0.543	0.342	0.202	0.350	0.729
DL2R	0.899	0.626	0.783	0.944	0.488	0.527	0.330	0.193	0.342	0.705
MV-LSTM	0.906	0.653	0.804	0.946	0.498	0.538	0.348	0.202	0.351	0.710
Match-LSTM	0.904	0.653	0.799	0.944	0.500	0.537	0.345	0.202	0.348	0.720
Attentive-LSTM	0.903	0.633	0.789	0.943	0.495	0.523	0.331	0.192	0.328	0.718
Multi-Channel	0.904	0.656	0.809	0.942	0.506	0.543	0.349	0.203	0.351	0.709
Multi-Channel$_{exp}$	0.714	0.368	0.497	0.745	0.476	0.515	0.317	0.179	0.335	0.691
SMN$_{last}$	**0.923**	**0.723**	**0.842**	**0.956**	**0.526**	**0.571**	**0.393**	**0.236**	**0.387**	0.729
SMN$_{static}$	**0.927**	**0.725**	**0.838**	**0.962**	**0.523**	**0.572**	**0.387**	**0.228**	**0.387**	0.734
SMN$_{dynamic}$	**0.926**	**0.726**	**0.847**	**0.961**	**0.529**	**0.569**	**0.397**	**0.233**	**0.396**	0.724

从实验效果来看，SMN相比较之前的Multi-view有很大的提升，这也说明了：

1. 在q-r匹配上，基于交互的模型相比基于表示的模型有更大的优势，这一点与检索式问答和NLI任务中的实验表现一致；
2. 对文本进行多粒度表示是很有必要的。

utterance也要深度encoding！DUA model

虽然看似SMN已经考虑很周到了，但是如果细想一下，其实SMN的建模方式还是跟现实中人们的聊天习惯存在不小的gap的。其中一个方面就是，Multi-view和SMN都没有重视utterances之间的语义关系，仅仅是通过一层Gated RNN进行了软过滤和简单encoding。然而其实很多时候建模utterances之间的关系是很有必要的，甚至对于过滤来说也是很重要的信息，这也是DUA的motivation。我们知道，其实聊天中很少从头到尾都是一个主题，比如下面的对话：

case1:

u1-> 路人甲：小夕，中秋节你去哪里玩儿啦？

u2-> 小夕：当然是去买买买呀～

u3-> 路人甲：你之前不是想去爬百望山嘛？没去嘛？

u4-> 小夕：想去呀，然鹅她们去玩儿都不带我(。ﾍ｡)

u5-> 路人甲：你稍等下啊，我下楼取个快递

u6-> 小夕：去吧去吧，顺便帮我买个辣条！

u7-> 路人甲：好呀，要啥口味的？鸡肉味？

u8-> 小夕：这特喵的还分口味？

u9-> 路人甲：回来啦，对了，要不然下周我带你去吧？

u10-> 小夕：好呀好呀，喵喵喵～

这里如果把小夕看作是检索式chatbot，假如对话进行到第6步(u6)，这时候最后一个utterance是u5，也就是“你稍等下啊，我下楼去取个快递”。显然，这时候其实相当于对话的话题发生了剧烈偏移，如果这时候小夕去跟一堆candidate responses做匹配的时候还去考虑u1-u4这些爬山相关的utterances的话，显然就容易召回跟u5很不相关的回复。同样的道理，如果对话进行到u8，其实这时候真正有用的historical utterances是u6-u7；对话进行到u10的时候，有用的utterances又变成了u1-u4。

除此之外，对话中还容易夹杂一些类似于停用词的噪声，比如

case2:

u1-> 路人乙：小夕，明天约约约？

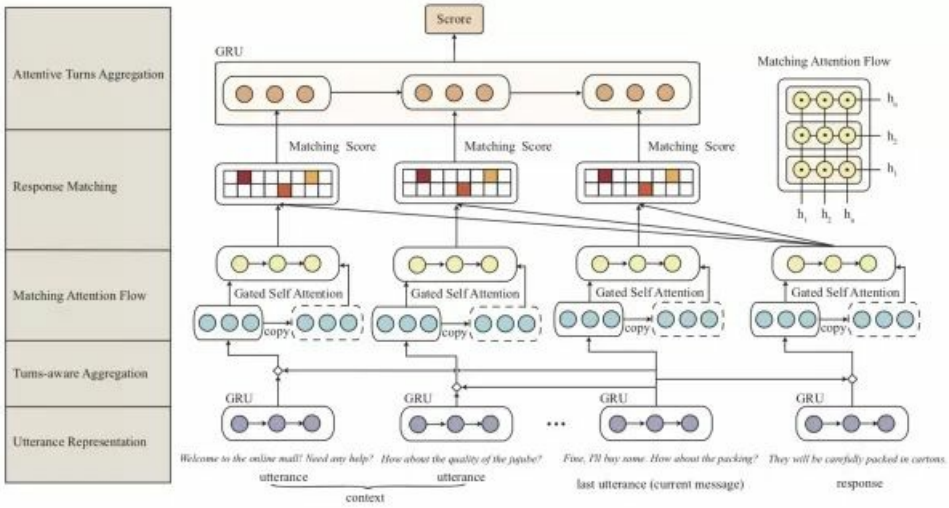
u2-> 小夕：。。。

u3-> 路人甲：哈哈

u4-> 小夕：应该木有时间

这里的u2和u3就是类似于停用词的“停用utterance”，所以对于这一类utterance，最好的办法就是忽略掉而不是让它们参与匹配。

怎么解决上述这两类问题呢？那就直接上这个让人看着灰常懵逼的DUA的模型图吧：



如图，这个图乍一看有点乱（其实画的确实很不怎么样（作者应该不会看我的文章吧2333）），论文里的公式标记也用的乱乱的（尤其第3.3节凭空冒出来的n弄得我懵逼了好久，到底是不是3.1节的n，是的话这里貌似就不对了，如果不是，这里又代表啥），一些细节也没交代清楚（比如3.1的S到底是个矩阵还是向量，如果是向量，那么怎么得到的这个向量？是矩阵的话3.2节的聚合又不对了）。

好了不吐槽了，不过其实这里的思想很直接，就是说，以前的paper呀，得到utterance embedding后就直接拿去RNN了，都没有像处理word embedding那样去好好做encoding，所以我们这里对utterance embedding也同样要做深度的encoding！

那么怎么做这个encoding呢？通过观察上面的俩cases可以发现，很多时候对话中是有hole的（比如上面case1中的u9的上一句话是u4，所以u5-u8形成了一个空洞），甚至可能很多个hole，所以这里做encoding的时候最合适的是使用self-attention而不是RNN更不是CNN。所以作者在这里先用了一层（加性）self-attention来把上下文编码进每个utterance embedding：

$$
\begin{aligned}
s_j^t &= v^T tanh(W_v f_j + W_{\tilde{v}} f_t + b_r) \\
a_i^t &= exp(s_i^t) / \sum_{j=1}^{n} exp(s_j^t) \\
c_t &= \sum_{i=1}^{n} a_i^t f_i
\end{aligned}
\tag{4}
$$

这里ft是t时刻的utterance embedding（就是前面聚合操作之后的那个向量表示），fj (j=1,2,...n) 是其上下文（即全部时刻的utterance embedding，一共n个）。 通过这个encoding操作，一下子每个时刻的utterance都能跨越时间和空洞把自己的那一群小伙伴聚在一起啦。

然鹅显然self-attention丢失了utterance的顺序信息，因此作者这里又把encoding后的utterance embedding跟encoding前的utterance embedding拼接起来又过了一层Gated RNN：

$$
p_t = GRU(p_{t-1}, [f_t, c_t]) \tag{3}
$$

Gated RNN（GRU、LSTM等）一方面可以按照时序进一步encoding，另一方面里面的输入门也起到了filter的作用，正好可以在加强encoding的同时把无用的信息过滤掉。看，这样就完成了当时的motivation，最后的这个utterance embedding可以说干净合理的多了。整个模型的其他部分则跟SMN基本没区别。

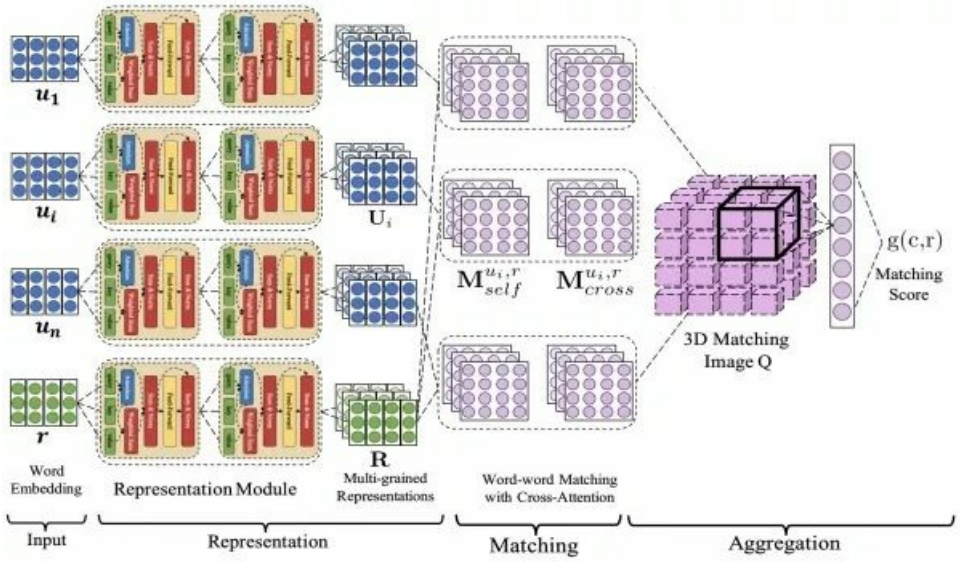
Model	Ubuntu Dialogue Corpus			Douban Conversation Corpus					
	$R_{10}@1$	$R_{10}@2$	$R_{10}@5$	MAP	MRR	P@1	$R_{10}@1$	$R_{10}@2$	$R_{10}@5$
TF-IDF	0.410	0.545	0.708	0.331	0.359	0.180	0.096	0.172	0.405
RNN	0.403	0.547	0.819	0.390	0.422	0.208	0.118	0.223	0.589
CNN	0.549	0.684	0.896	0.417	0.440	0.226	0.121	0.252	0.647
LSTM	0.638	0.784	0.949	0.485	0.537	0.320	0.187	0.343	0.720
BiLSTM	0.630	0.780	0.944	0.479	0.514	0.313	0.184	0.330	0.716
Multi-View	0.662	0.801	0.951	0.505	0.543	0.342	0.202	0.350	0.729
DL2R	0.626	0.783	0.944	0.488	0.527	0.330	0.193	0.342	0.705
MV-LSTM	0.653	0.804	0.946	0.498	0.538	0.348	0.202	0.351	0.710
Match-LSTM	0.653	0.799	0.944	0.500	0.537	0.345	0.202	0.348	0.720
Attentive-LSTM	0.633	0.789	0.943	0.495	0.523	0.331	0.192	0.328	0.718
Multi-Channel	0.656	0.809	0.942	0.506	0.543	0.349	0.203	0.351	0.709
Multi-Channel$_{exp}$	0.368	0.497	0.745	0.476	0.515	0.317	0.179	0.335	0.691
SMN	0.726	0.847	0.961	0.529	0.569	0.397	0.233	0.396	0.724
DUA	**0.752**	**0.868**	**0.962**	**0.551**	**0.599**	**0.421**	**0.243**	**0.421**	**0.780**

从实验结果来看，DUA的性能确实比SMN有了进一步明显的提升。

state-of-the-art：DAM model

这篇是多轮对话领域难得的好paper，可能xiangyang大佬太忙，都木有打打广告什么的┐(￣▽￣"")┌。作者这里抛弃了之前的建模utterance embedding sequence的思路，而是把NLP很多领域的前沿操作优雅干净的整合为一个全新的框架来建模多轮对话问题，不仅模型非常work，实验章节也对模型各个component的特点和有效性进行了充分的探索和论证，是继Multi-view和SMN以来多轮对话领域又一个不得不提的经典模型。

另外，遇到一张清晰漂亮的模型图不容易哇，就直接上图吧



> ps：这张图这么少女心，我猜是lilu女神画的。

还记得前面说的SMN的一个亮点是做了两级粒度的文本表示嘛？那么很自然的就有了一个问题：两级就够了嘛？有没有必要设置更多级呢？如果有必要的话，那么怎么去表示和学习这更多级粒度的语义表示呢？

首先答案当然是肯定的，17年的SSE文本匹配模型和今年特别火的ELMo[10]都说明了对文本的深层表示可以学习到更加高level的语义单元，然而我们知道像SSE和ELMo这种堆多层RNN的做法会极大的增加模型的推理代价，这极大的限制了它们在工业界的应用。而堆多层CNN在文本里又不容易调work，需要精细的设计网络并借助一些tricks，因此很自然的做法就是使用Transformer[11] encoder来得到文本的多级表示啦（没看过transformer那篇paper的赶紧去补啦，做NLP哪能不知道transformer）。

所以如图，DAM首先就用transformer的encoder来得到了每个utterance和response的多粒度文本表示（即图中的Representation部分），之后作者对每个utterance-response pair的每个粒度下的表示分别计算两个对齐矩阵（即图中的Matching部分）。

等下，怎么是俩对齐矩阵？除了传统的计算对齐矩阵的方式，还有新的玩法啦？

这里作者提出了一种更加深（隐）层（晦）的匹配方法，操作不难，但是为什么会work还是挺难以理解透彻的（虽然作者在5.2节已经有很努力的讲了）。总之，先来简单提一下传统的attention计算对齐矩阵的方式。

传统的方法无非就是把文本1中的word embedding sequence和文本2中的word embedding sequence进行词-词比较，这里的比较分为加性方法和乘性方法，基础差的同学可以看下面这段复习一下。

注：词-词比较的方式分为加性和乘性，加性就是将要比较的两个word embedding进行相加（相加前可以先过一个线性变换甚至MLP）然后激活后跟一个虚拟的向量做内积（其实这个虚拟向量就是个可训练的同维度向量，我理解的它存在的意义就是对每个维度的加法比较+激活后的结果进行scaling，毕竟维度不同方差也可能不同嘛），内积的结果就是对齐程度啦。乘性则容易理解一些，就是将两个word embedding直接进行相乘（准确说是内积）或中间夹一个可训练方阵（即$v1^T * M * v2$的形式），内积的结果就是对齐的程度啦。不过要记得当维度很高时，乘性方式最好对结果做个归一化以免进入softmax饱和区（参考Transformer）。

$$\mathbf{M}_{self}^{u_i,r,l} = \{\mathbf{U}_i^l[k]^T \cdot \mathbf{R}^l[t]\}_{n_{u_i} \times n_r} \qquad (7)$$

如上式，作者这里使用的是乘性的方式，这里的l就是指的第l级粒度，ui是指的第i个utterance，ui有n_{ui}个词，response有nr个词。这里就是说，对于每级语义粒度的每个utterance，都是将其中的每个词k去跟response中该粒度下的每个词t去算内积，从而得到一个$n_{ui} * n_r$的对齐矩阵。

对于传统的attention，如果两个词在semantic或syntactic上离得近，就容易得到比较大的匹配值（如run和runs, do和what）。然而对于一些比较深层和隐晦的语义关系就很难直接匹配了（我们不能强求前面的网络把各级粒度的语义单元的

embedding都学的那么完美呀对吧），所以作者这里提出了一个更加间接和隐晦的attention方式，如下

$$\widetilde{\mathbf{U}}_i^l = \mathbf{AttentiveModule}(\mathbf{U}_i^l, \mathbf{R}^l, \mathbf{R}^l) \quad (8)$$

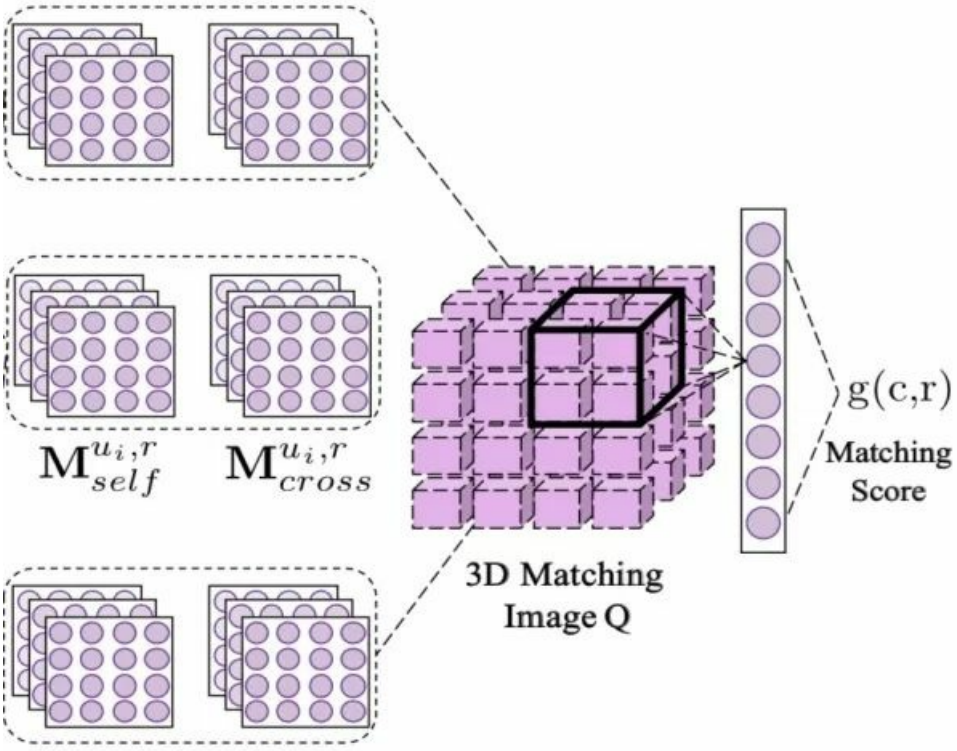
$$\widetilde{\mathbf{R}}^l = \mathbf{AttentiveModule}(\mathbf{R}^l, \mathbf{U}_i^l, \mathbf{U}_i^l) \quad (9)$$

$$\mathbf{M}_{cross}^{u_i,r,l} = \{\widetilde{\mathbf{U}}_i^l[k]^T \cdot \widetilde{\mathbf{R}}^l[t]\}_{n_{u_i} \times n_r} \quad (10)$$

这里的AttentiveModule的3个参数依次为attention的Query、Key和Value，不熟悉的同学去复习Transformer，这里就不赘述啦。首先看公式8和9，这里先通过传统的attention来把utterance和response中的每个词用对面文本的词加权表示，得到新的utterance的word embeding sequence表示和新的response的word embedding sequence表示，之后再用一层传统的attention来计算出一个对齐矩阵来作为第二个对齐矩阵。

显然这种方式将utterance中的词和response中的词之间的依赖关系（dependency information）也作为词的表示加入了对齐矩阵的计算，所以说是建模了更加深（复）层（杂）的语义关系。不过，作者在论文5.2节有提到这两种attention方式匹配文本的操作其实是互补的，并且给出了一个case解释，然而小夕功力有限，努力理解了一下还是没理解┐(￣▽￣"")┌希望有看懂的小伙伴给小夕讲讲或者贴到评论区～

经过这么深层的匹配后，每个utterance中的每个词位都包含了2(L+1)维的匹配信息（L为Transformer encoder的层数，1为原始的word embedding，2为对齐矩阵的数量），作者这里又把utterances堆叠到一起，就形成了这个漂亮的3D粉色大立方体



所以这个大立方体的三个维度分别代表对话上下文中的每个utterance、utterance中的每个词（位）、response中的每个词（位）。

之后，再通过一个两层的3D的卷积神经网络来从这个大立方体中抽取特征，得到匹配层的特征，最后的最后通过一个单层感知机得到该candidate response的匹配概率。

说了这么多，来看看实验结果吧～

	Ubuntu Corpus				Douban Conversation Corpus					
	$R_2@1$	$R_{10}@1$	$R_{10}@2$	$R_{10}@5$	MAP	MRR	P@1	$R_{10}@1$	$R_{10}@2$	$R_{10}@5$
$\text{DualEncoder}_{lstm}$	0.901	0.638	0.784	0.949	0.485	0.527	0.320	0.187	0.343	0.720
$\text{DualEncoder}_{bilstm}$	0.895	0.630	0.780	0.944	0.479	0.514	0.313	0.184	0.330	0.716
MV-LSTM	0.906	0.653	0.804	0.946	0.498	0.538	0.348	0.202	0.351	0.710
Match-LSTM	0.904	0.653	0.799	0.944	0.500	0.537	0.345	0.202	0.348	0.720
Multiview	0.908	0.662	0.801	0.951	0.505	0.543	0.342	0.202	0.350	0.729
DL2R	0.899	0.626	0.783	0.944	0.488	0.527	0.330	0.193	0.342	0.705
$\text{SMN}_{dynamic}$	0.926	0.726	0.847	0.961	0.529	0.569	0.397	0.233	0.396	0.724
DAM	**0.938**	**0.767**	**0.874**	**0.969**	**0.550**	**0.601**	**0.427**	**0.254**	**0.410**	**0.757**

可以看到实验结果非常漂亮（当前的state-of-art），尤其是$R_{10}@1$这种比较有实际意义的指标（从10个candidates里召回top1）。而且DAM没有像DUA那样对utterance embedding sequence做深层encoding（这里直接用的3D conv抽特征了），但是实验结果明显比DUA好，可以说网络设计的很棒棒啦。
另外，作者这里也给出了去掉各个component后的性能情况：

DAM	**0.938**	**0.767**	**0.874**	**0.969**	**0.550**	**0.601**	**0.427**	**0.254**	**0.410**	**0.757**
DAM_{first}	0.927	0.736	0.854	0.962	0.528	0.579	0.400	0.229	0.396	0.741
DAM_{last}	0.932	0.752	0.861	0.965	0.539	0.583	0.408	0.242	0.407	0.748
DAM_{self}	0.931	0.741	0.859	0.964	0.527	0.574	0.382	0.221	0.403	0.750
DAM_{cross}	0.932	0.749	0.863	0.966	0.535	0.585	0.400	0.234	0.411	0.733

比如对比DAM与倒数第二行可以看到，去掉那个复杂的深度注意力机制后，网络性能出现了明显的下降，说明论文中提出的这个“间接”的注意力机制确实能捕获到一些神奇的模式。

总结

最后小夕非常主观的总结一下这四个模型的亮点：

Multi-view提出了将utterance建模为一个语义单元来建模多轮对话问题；

SMN使用基于交互的匹配模型代替基于表示的匹配模型，并对文本进行多粒度表示；

DUA对utterance embedding进行深度的encoding来建模utterances之间的依赖关系；

DAM一方面对文本对进行多粒度表示并提出了一种深度attention的方法，另一方面抛弃了之前建模utterance embedding sequence的思路，提出了一种将word-level和utterance-level的信息整合到一起，构建一个多通道的3D Image（其实把utterance看成单帧的图像，那这个大方块更像是一个视频），进而通过3D Image分类器完成匹配的新思路。

参考文献

[1] Multi-view Response Selection for Human-Computer Conversation, EMNLP2016
[2] Sequential Matching Network- A New Architecture for Multi-turn Response Selection in Retrieval-Based Chatbots, ACL2017
[3] Modeling Multi-turn Conversation with Deep Utterance Aggregation, COLING2018
[4] Multi-Turn Response Selection for Chatbots with Deep Attention Matching Network, 2018ACL
[5] Text Matching as Image Recognition, AAAI2016
[6] Neural Network Models for Paraphrase Identification, Semantic Textual Similarity, Natural Language Inference, and Question Answering, COLING2018
[7] Enhanced LSTM for Natural Language Inference, ACL2017
[8] Shortcut-Stacked Sentence Encoders for Multi-Domain Inference, Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP. 2017
[9] Learning Deep Structured Semantic Models for Web Search using Clickthrough Data, CIKM2013
[10] Deep contextualized word representations, NAACL2018
[11] Attention Is All You Need, NIPS2017